Extracción de reglas

Laura del Pino Díaz

23/1/2017

Índice

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 metricas 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.

```r
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
```

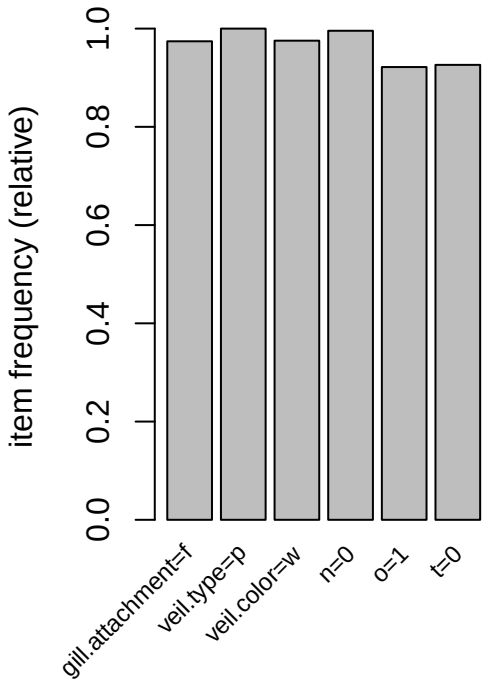
Del resumen anterior obtenermos que los elementos más repetidos son *veil.type=p*, que se da para todas las observaciones de la base de datos, seguido por *n=0* que es una de las variables generadas a partir de *ring.number* de carácter binaria, por lo que se estima que sea más normal que las otras dos 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

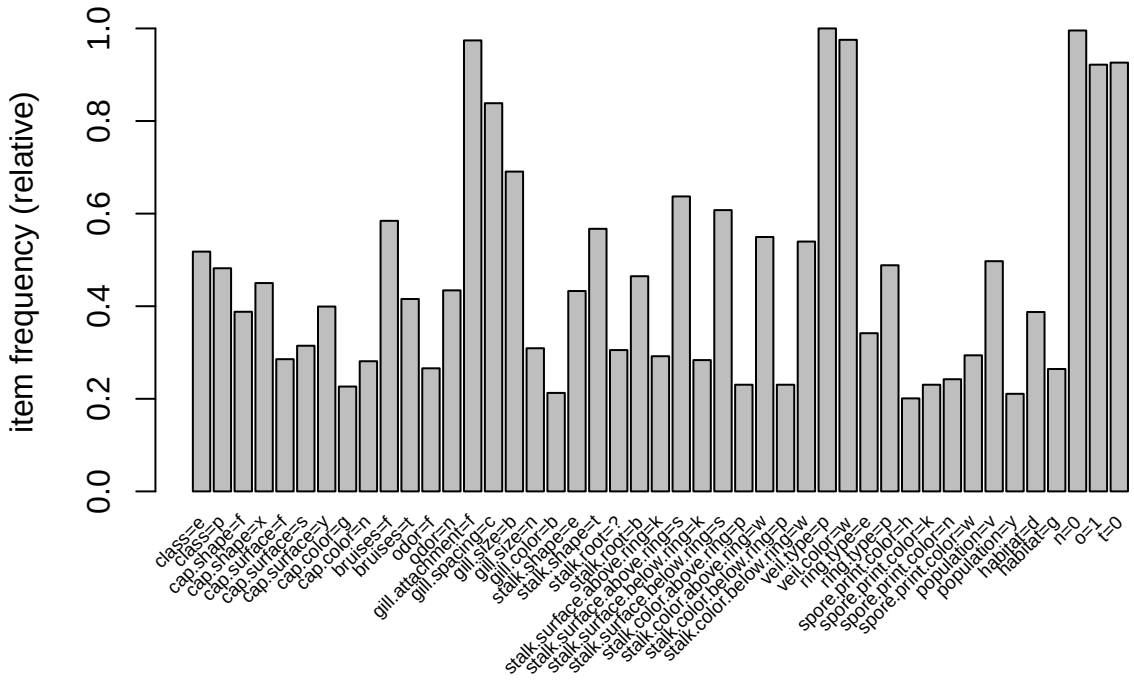
En el caso de que las variables tomasen solamente los valores *si* o *no* podríamos y que el número de variables en cada transacción fuese distinto podría ser interesante realizar una visualización de las transacciones. Pero dado que este no es el caso solamente visualizaremos los conjuntos de items 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 almenos el 90 % de los casos recogidos en la base de datos.

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



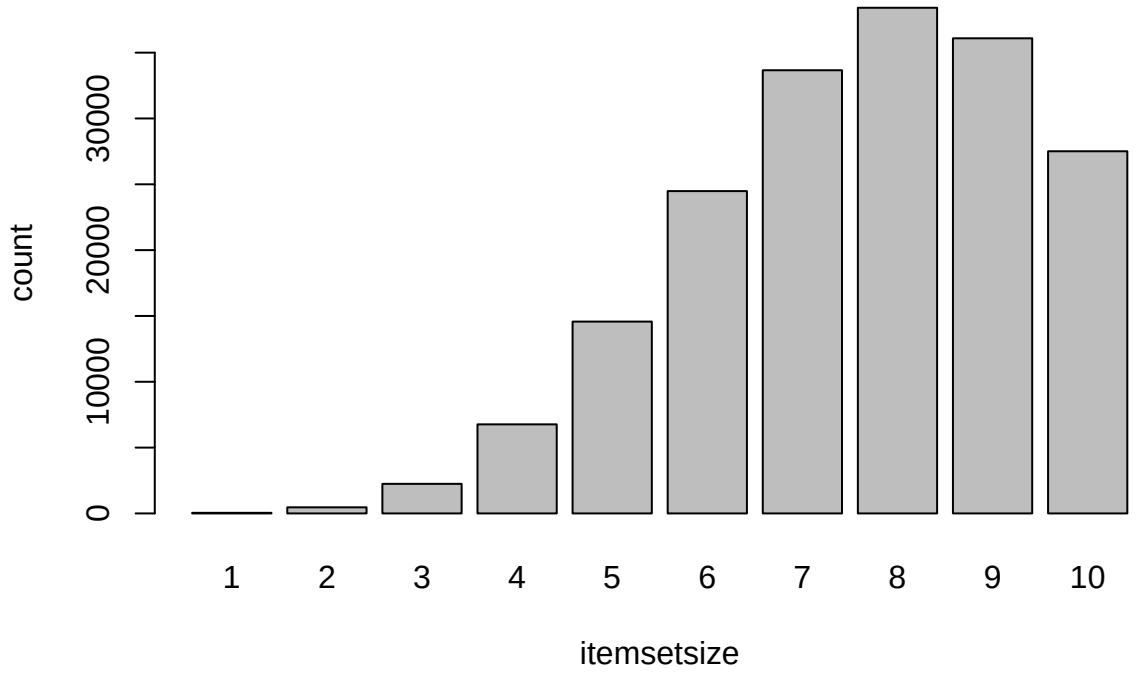
El gráfico anterior representa una población de aproximadamente 40 items que se convierten en frecuentes al determinar el umbral de soporte en 0.2 . 40 items representa una cantidad significativa de los valores que pueden tomar las variables de nuestra base de datos. Podemos dejar el soporte mínimo en 0.2, para tener un conjunto significativo de items 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 insperccionar el soporte de los itemsets de longitud 1 que hemos obtenido.

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

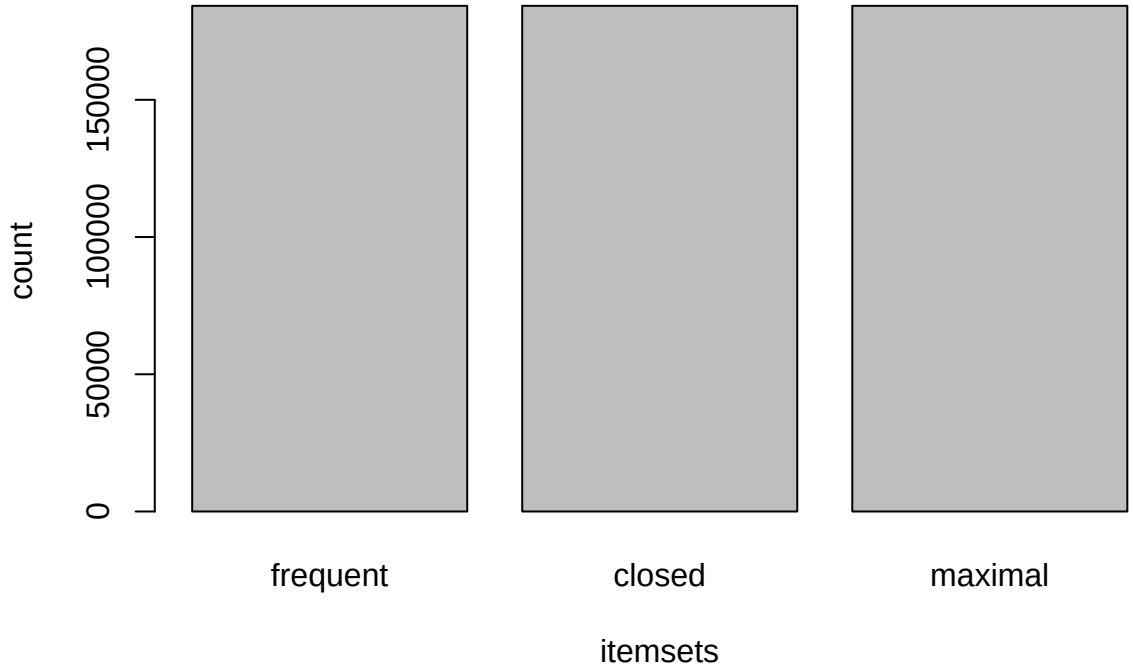
```
     items                          support
[1]  {veil.type=p}                  1.0000000
[2]  {n=0}                          0.9955687
[3]  {veil.color=w}                 0.9753816
[4]  {gill.attachment=f}            0.9741507
[5]  {t=0}                          0.9261448
[6]  {o=1}                          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.

```
icloAPrioriTransactionData <- 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 presindir 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
```

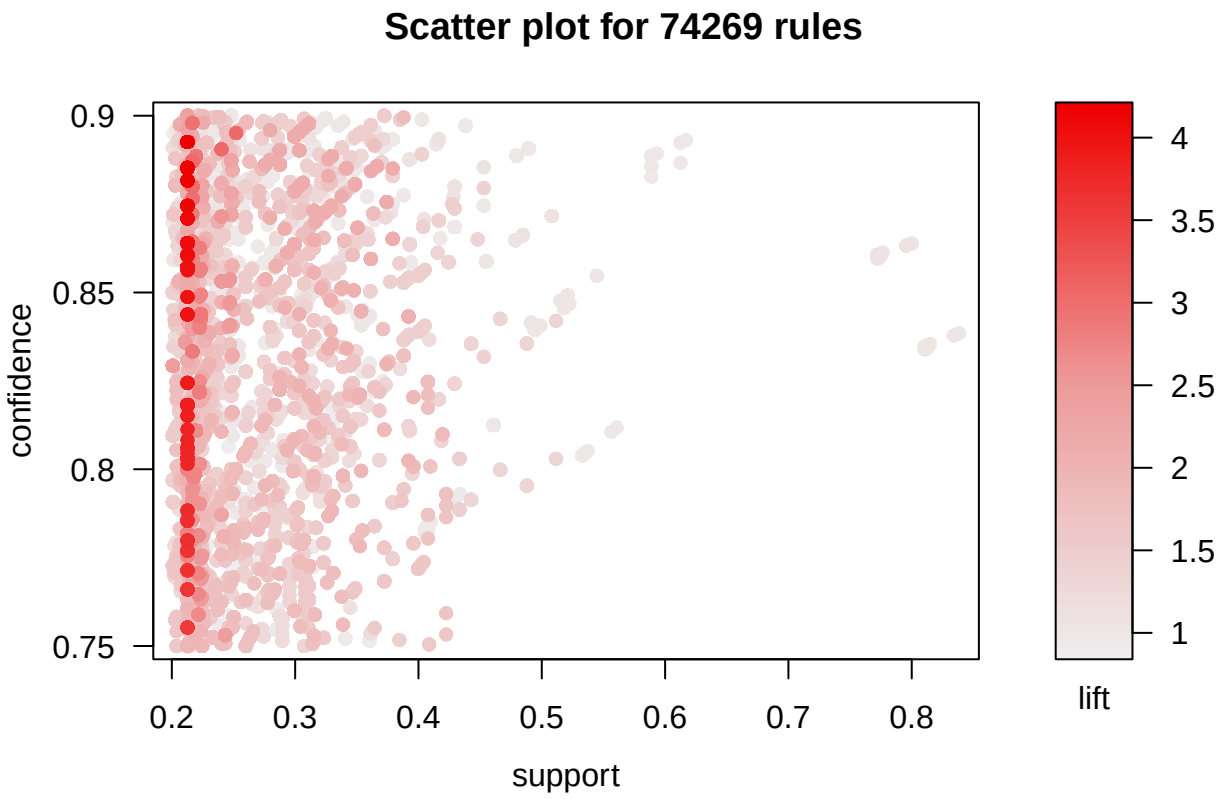
Como podemos ver, aparecen a la cabeza aquellas reglas que tienen un itemset con un soporte alto en el consecuente, por lo que da igual lo que se ponga en el antecedente, la regla tendrá una confianza alta. Este tipo de reglas las eliminamos con el siguiente código.

```
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
```

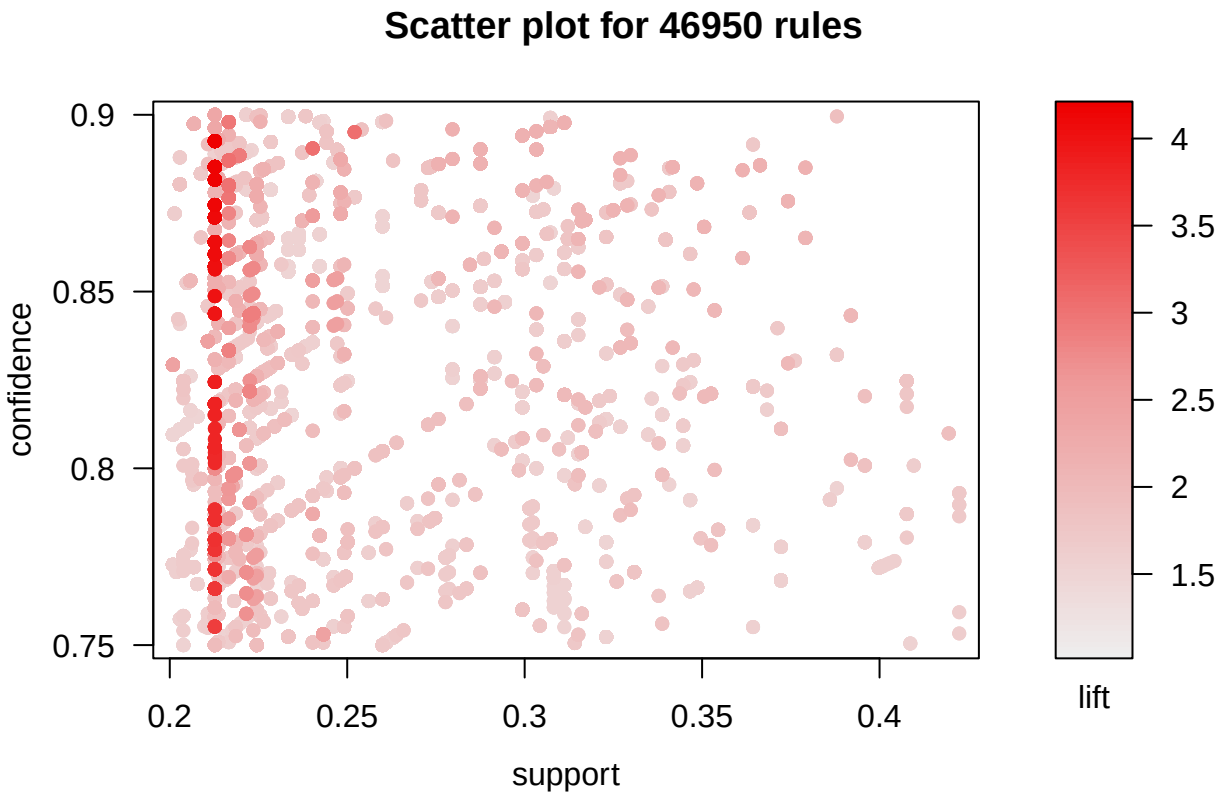
Para extraer información de valor vamos a focalizar el estudio de las reglas en unas pocas, para seleccionar las más interesantes lo primero que haremos es mostrar las reglas en una comparación entre su soporte y su confianza.

```
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)
```



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
```

```
sortRules <- sort(rulesInteresting, by = "interestingMeasure")
```

Estudio de las reglas por grupos.

Estudio de los tipos de seta

Estudio de las setas comestibles

Puesto que hemos determinado que el consecuente todas las reglas solamente tenga un item por lo que podemos realizar una selección de las reglas por el consecuente. En el caso particular de la base de datos *mushrooms* con la que estamos trabajando podemos tratar de resolver el problema de qué setas son comestibles y cuales no.

```
eatableMushrooms <- subset(sortRules, subset = rhs %in% "class=e")
inspect(head(eatableMushrooms))
```

```
    lhs                              rhs         support confidence    lift interestingMeasure
[1] {gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.below.ring=w}   => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
[2] {gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.above.ring=w}   => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
[3] {gill.attachment=f,
     gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.below.ring=w}   => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
[4] {gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.below.ring=w,
     veil.color=w}               => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
[5] {gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.below.ring=w,
     n=0}                        => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
[6] {gill.size=b,
     stalk.surface.above.ring=s,
     stalk.color.below.ring=w,
     veil.type=p}                => {class=e} 0.2383063  0.8996283 1.73683          0.7992565
```

Si miramos el conjunto de las reglas vemos que todas ellas nos tienen el mismo valor para todas las medidas de interés, y todas las reglas comparten dos items en el antecedente la superficie encima del anillo con el valor suave (*smooth* en inglés) y el color del tallo debajo del anillo con el valor blanco. Estos dos valores tendrán un soporte muy alto para que se añada cualquier item y siga manteniendo el valor de soporte. Por lo que se puede afirmar con un 0.799 de confianza confirmada de los casos las setas que tienen una superficie lisa por encima del anillo del tallo y el color del tallo por debajo del anillo sea blanca es comestible.

Estudio de las setas venenosas

Si el 20 por ciento de los casos que tienen la superficie encima del anillo con el valor suave (*smooth* en inglés) y el color del tallo debajo del anillo con el valor blanco la seta es venenosa podemos interesarnos en cuáles son las características que tienen las setas venenosas.

```
poisinousMushrooms <- subset(sortRules, subset = rhs %in% "class=p")
inspect(head(poisinousMushrooms))
```

```
    lhs                    rhs         support confidence     lift interestingMeasure
[1] {bruises=f,
     gill.spacing=c,
     n=0}               => {class=p} 0.3879862  0.8995434 1.866162          0.7990868
[2] {bruises=f,
     gill.spacing=c,
     veil.type=p,
     n=0}               => {class=p} 0.3879862  0.8995434 1.866162          0.7990868
[3] {gill.size=n}       => {class=p} 0.2737568  0.8853503 1.836718          0.7707006
[4] {gill.size=n,
     o=1}               => {class=p} 0.2737568  0.8853503 1.836718          0.7707006
[5] {gill.size=n,
     t=0}               => {class=p} 0.2737568  0.8853503 1.836718          0.7707006
[6] {gill.attachment=f,
     gill.size=n}       => {class=p} 0.2737568  0.8853503 1.836718          0.7707006
```

Como en el estudio de las setas comestibles, en las reglas obtenidas podemos ver que en aquellas reglas que mantienen el mismo valor en las medidas de interés tienen un itemset común. En las primeras dos reglas tienen en común los items de existencia de moretones, espacio pequeño entre branquias y la variable "número de anillo igual no" falso para determinar que la seta es venenosa, con una confianza confirmada del 0.79. Las restantes 4 reglas tienen como denominador común la variable de el espacio entre agallas con el valor estrecho.

Estudio de la localización de las setas.

Ahora que sabemos que características tienen las setas comestibles y las setas no comestibles, queremos saber en donde se encuentran cada una.

Localización de las setas comestibles.

```
locEatableMushrooms <- subset(sortRules, subset = lhs %in% "class=e" &
    rhs %pin% "habitat=")
inspect(head(locEatableMushrooms))
```

```
    lhs                           rhs           support confidence     lift interestingMeasure
[1] {class=e,
     odor=n,
     gill.size=b,
     ring.type=p,
     o=1}                      => {habitat=d} 0.2127031        0.9 2.322618               0.8
[2] {class=e,
     odor=n,
     gill.size=b,
     ring.type=p,
     t=0}                      => {habitat=d} 0.2127031        0.9 2.322618               0.8
```

```
[3] {class=e,
     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] {class=e,
     odor=n,
     gill.spacing=c,
     gill.size=b,
     o=1}                        => {habitat=d} 0.2127031        0.9 2.322618               0.8
[5] {class=e,
     odor=n,
     gill.spacing=c,
     gill.size=b,
     t=0}                        => {habitat=d} 0.2127031        0.9 2.322618               0.8
[6] {class=e,
     gill.size=b,
     stalk.shape=t,
     stalk.surface.above.ring=s,
     stalk.surface.below.ring=s} => {habitat=d} 0.2127031        0.9 2.322618               0.8
```

Las setas comestibles se encuentran en los bosques con una confianza confirmada del 0.8.

Localización de las setas venenosas

```
locPMushrooms <- subset(sortRules, subset = lhs %in% "class=p" &
    rhs %pin% "habitat=")
```

No se puede determinar ningún habitat concreto para las setas venenosas puesto que ninguna regla tiene soporte suficiente.

Estudio de la población de las setas

Otro factor que puede ayudar a determinar si una seta es venenosa o no, es si vive solitaria o si vive en población dentro del hábitat.

```
popEMushrooms <- subset(sortRules, subset = lhs %in% "class=e" &
    rhs %pin% "population=")
popPMushrooms <- subset(sortRules, subset = lhs %in% "class=p" &
    rhs %pin% "population=")

# size(popEMushrooms) #Devuelve integer 0 size(popPMushrooms)
# #Distinto de 0
```

No podemos determinar que haya algún tipo población frecuente para las setas comestibles ya que no tienen ninguna regla frecuente.

```
inspect(head(popPMushrooms))
```

```
    lhs                    rhs              support confidence     lift interestingMeasure
[1] {class=p,
     gill.size=n,
     veil.color=w}      => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
[2] {class=p,
```

```
     gill.size=n,
     veil.color=w,
     o=1}               => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
[3] {class=p,
     gill.size=n,
     veil.color=w,
     t=0}               => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
[4] {class=p,
     gill.attachment=f,
     gill.size=n,
     veil.color=w}      => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
[5] {class=p,
     gill.size=n,
     veil.color=w,
     n=0}               => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
[6] {class=p,
     gill.size=n,
     veil.type=p,
     veil.color=w}      => {population=v} 0.2442147  0.8953069 1.800365          0.7906137
```

Las reglas anteriores podemos afirmar con una confianza confirmada de 0.79 que las setas venenosas se encuentran juntas en conjuntos reducidos.

Conclusión

En este trabajo se ha examinado la base de datos de *mushrooms* en la que hemos revisado los itemset más frecuentes, los itemset cerrados y los maximales para saber que reglas eran a priori superfluas y no aportaban información.

Tras ello obtuvimos las reglas mediante el método a priori y descartamos aquellas que eran potencialmente superfluas y aquellas que son estadísticamente independientes.

Con el conjunto resultante se hizo un análisis por grupos de reglas en el que nos interesamos por como se puede definir las setas comestibles y las venenosas, donde se sitúan las mismas y en que tipo de poblaciones se suelen encontrar obteniendo las siguientes conclusiones: * Las setas comestibles se definen por la superficie encima del anillo con el valor suave y el color del tallo debajo del anillo con el valor blanco. * Las setas venenosas se definen o bien por existencia de moretones, espacio pequeño entre branquias y la variable “número de anillo igual no” falso o bien por el espacio entre agallas con el valor estrecho. * Las setas comestibles se encuentran sobretodo en bosques. * Las setas venenosas no se puede definir un hábitad frecuente. * Las setas comestibles no tienen un tipo de población definida. * Las setas venenosas se suelen agurpar en grupos pequeños.