Tarea4

Karina De Sousa

8 de mayo de 2016

```
##  [1] "stringdist" "arules"     "Matrix"     "stats"      "graphics"
##  [6] "grDevices"  "utils"      "datasets"   "methods"    "base"
```

Sistema de recomendación

Un importante periódico le entrega a usted un dataset limpio con información acerca del acceso a su portal web. El mismo contiene **131300** posibles transacciones en un tiempo determinado. Se sabe que el portal ofrece **9 tipos de contenidos y nos ofrecen solo informaci?n de 9 artículos**. Los contenidos son:

- Deportes
- Politica
- Variedades
- Internacional
- Nacionales
- Sucesos
- Comunidad
- Negocios
- Opinión

El periódico tiene sospechas de que existen bots que están ganando dinero al hacer clicks en artículos con promociones. En consecuencia, le piden a usted que realice un análisis exploratorio sobre las transacciones para determinar el número de posibles transacciones bot que tienen en su dataset (**ellos aceptan que si una persona ve un artículo más de 20 segundos entonces no es un bot**). Aunado a esto, tienen una lista de demandas que debe suplir usted, el experto.

Transacciones Bot

1. Modificar su dataset de tal manera que no se lean los identificadores de los artículos como **itemN** sino por su tipo de contenido **contenido/articuloN**. Ejemplo: {item1, item10, item81} es la transacción {deportes/articulo1, politica/articulo1, opinion/articulo9}.

```
> # Set Working directory
>
> transactions = read.csv(file = "./periodico.csv", header = TRUE, sep = ",")
>
> # transactions <- subset(transactions, select = -c(X) )
```

- Creamos la función **createItemString**, que dada una lista de items retorna el string de la forma **contenido/articuloN**,

```
> # 1-9 Deportes - 1 10-18 Politica - 2 19-27 Variedades - 3 28-36
> # Internacional - 4 37-45 Nacionales - 5 46-54 Sucesos - 6 55-63 Comunidad -
> # 7 64-72 Negocios - 8 73-81 Opinion - 9
>
```

```
> createItemString = function(items) {
+     items = unlist(items)
+     finalItem = ""
+     for (i in 1:length(items)) {
+         # Obtenemos el numero de contenido al que pertenece el item
+         content = as.integer(items[i]/9)
+         if (items[i]%%9 != 0) {
+             content = content + 1
+             # Obtenemos el numero del articulo
+             n = items[i]%%9
+         } else {
+             # Si es el ultimo articulo del contenido
+             n = 9
+         }
+
+         # Seleccionamos el contenido
+         if (content == 1) {
+             str = paste("deportes/articulo", n, sep = "")
+         } else if (content == 2) {
+             str = paste("politica/articulo", n, sep = "")
+         } else if (content == 3) {
+             str = paste("variedades/articulo", n, sep = "")
+         } else if (content == 4) {
+             str = paste("internacional/articulo", n, sep = "")
+         } else if (content == 5) {
+             str = paste("nacionales/articulo", n, sep = "")
+         } else if (content == 6) {
+             str = paste("sucesos/articulo", n, sep = "")
+         } else if (content == 7) {
+             str = paste("comunidad/articulo", n, sep = "")
+         } else if (content == 8) {
+             str = paste("negocios/articulo", n, sep = "")
+         } else if (content == 9) {
+             str = paste("opinion/articulo", n, sep = "")
+         }
+
+         if (i == 1) {
+             finalItem = paste(finalItem, str, sep = "")
+         } else {
+             finalItem = paste(finalItem, str, sep = ",")
+         }
+     }
+     finalItem
+ }
```

- Creamos la nueva columna del dataset,

```
> # Obtenemos los numeros de cada item
> itemsList = lapply(transactions$articles, function(article) unique(na.omit(as.numeric(unlist(strsplit
+     "[^0-9]+"))))))
>
> final = c(1:length(itemsList))
>
```

```
> final = sapply(itemsList, createItemString)
>
> head(final)
```

```
## [1] "deportes/articulo1,deportes/articulo9,comunidad/articulo9"
## [2] "deportes/articulo1,deportes/articulo2,deportes/articulo3"
## [3] "deportes/articulo9,nacionales/articulo7,comunidad/articulo3"
## [4] "deportes/articulo2,politica/articulo5,negocios/articulo9"
## [5] "politica/articulo2"
## [6] "deportes/articulo6,sucesos/articulo8"
```

```
> # Agregamos la nueva columna
> transactions = data.frame(transactions, final)
>
> # Cambiamos el nombre de la cabecera
> names(transactions) = c("X", "ID", "entry", "exit", "items", "articles")
>
> head(transactions)
```

```
##   X     ID               entry                exit                  items
## 1 1 trans1 2016-05-02 22:39:52 2016-05-02 22:49:08  {item1,item9,item63}
## 2 2 trans2 2016-05-02 17:38:55 2016-05-02 17:53:29   {item1,item2,item3}
## 3 3 trans3 2016-05-01 06:57:57 2016-05-01 07:00:44 {item9,item43,item57}
## 4 4 trans4 2016-05-01 09:10:07 2016-05-01 09:15:16 {item2,item14,item72}
## 5 5 trans5 2016-05-01 00:28:29 2016-05-01 01:01:16              {item11}
## 6 6 trans6 2016-04-30 02:18:59 2016-04-30 02:24:49        {item6,item53}
##                                                  articles
## 1   deportes/articulo1,deportes/articulo9,comunidad/articulo9
## 2    deportes/articulo1,deportes/articulo2,deportes/articulo3
## 3 deportes/articulo9,nacionales/articulo7,comunidad/articulo3
## 4    deportes/articulo2,politica/articulo5,negocios/articulo9
## 5                                        politica/articulo2
## 6                       deportes/articulo6,sucesos/articulo8
```

- Ahora, veamos cuanto tarda cada transacción. Sabemos que aquellas que duren menos de 20 seg son consideradas bots.

```
> diffTimes = c(1:length(transactions$entry))
> diff = 0
>
> for (i in 1:length(transactions$entry)) {
+     # Convertir a Date Obtener seg entre entry y exit
+     diff = strptime(transactions$exit[i], "%Y-%m-%d %H:%M:%S", tz = "GMT") -
+         strptime(transactions$entry[i], "%Y-%m-%d %H:%M:%S", tz = "GMT")
+
+     # Convertir a numeric
+     diff = as.numeric(diff, units = "secs")
+     diffTimes[i] = diff
+ }
>
> times = data.frame(transactions, diffTimes)
> names(times) = c("X", "ID", "entry", "exit", "items", "articles", "diff")
```

```
>
> head(times)
```

```
##   X     ID               entry                exit                   items
## 1 1 trans1 2016-05-02 22:39:52 2016-05-02 22:49:08  {item1,item9,item63}
## 2 2 trans2 2016-05-02 17:38:55 2016-05-02 17:53:29   {item1,item2,item3}
## 3 3 trans3 2016-05-01 06:57:57 2016-05-01 07:00:44 {item9,item43,item57}
## 4 4 trans4 2016-05-01 09:10:07 2016-05-01 09:15:16 {item2,item14,item72}
## 5 5 trans5 2016-05-01 00:28:29 2016-05-01 01:01:16              {item11}
## 6 6 trans6 2016-04-30 02:18:59 2016-04-30 02:24:49        {item6,item53}
##                                                    articles diff
## 1   deportes/articulo1,deportes/articulo9,comunidad/articulo9  556
## 2    deportes/articulo1,deportes/articulo2,deportes/articulo3  874
## 3 deportes/articulo9,nacionales/articulo7,comunidad/articulo3  167
## 4    deportes/articulo2,politica/articulo5,negocios/articulo9  309
## 5                                        politica/articulo2 1967
## 6                        deportes/articulo6,sucesos/articulo8  350
```

El dataframe **times** contiene el tiempo que dura cada transacción y una fila **bot** donde se indica si la misma es un bot o no. De esta forma, la cantidad de transacciones bot es,

```
> bots = diffTimes[diffTimes <= 20]
> length(bots)
```

```
## [1] 2302
```

```
> nobots = times[times$diff > 20, ]
> head(nobots)
```

```
##   X     ID               entry                exit                   items
## 1 1 trans1 2016-05-02 22:39:52 2016-05-02 22:49:08  {item1,item9,item63}
## 2 2 trans2 2016-05-02 17:38:55 2016-05-02 17:53:29   {item1,item2,item3}
## 3 3 trans3 2016-05-01 06:57:57 2016-05-01 07:00:44 {item9,item43,item57}
## 4 4 trans4 2016-05-01 09:10:07 2016-05-01 09:15:16 {item2,item14,item72}
## 5 5 trans5 2016-05-01 00:28:29 2016-05-01 01:01:16              {item11}
## 6 6 trans6 2016-04-30 02:18:59 2016-04-30 02:24:49        {item6,item53}
##                                                    articles diff
## 1   deportes/articulo1,deportes/articulo9,comunidad/articulo9  556
## 2    deportes/articulo1,deportes/articulo2,deportes/articulo3  874
## 3 deportes/articulo9,nacionales/articulo7,comunidad/articulo3  167
## 4    deportes/articulo2,politica/articulo5,negocios/articulo9  309
## 5                                        politica/articulo2 1967
## 6                        deportes/articulo6,sucesos/articulo8  350
```

Tipos de usuarios

Sabemos que,

- El valor de **soporte** de X (un set de elementos - antecedente) con respecto a T (un conjunto de transacciones) se define como la proporción de transacciones en el dataset que con contienen a X.

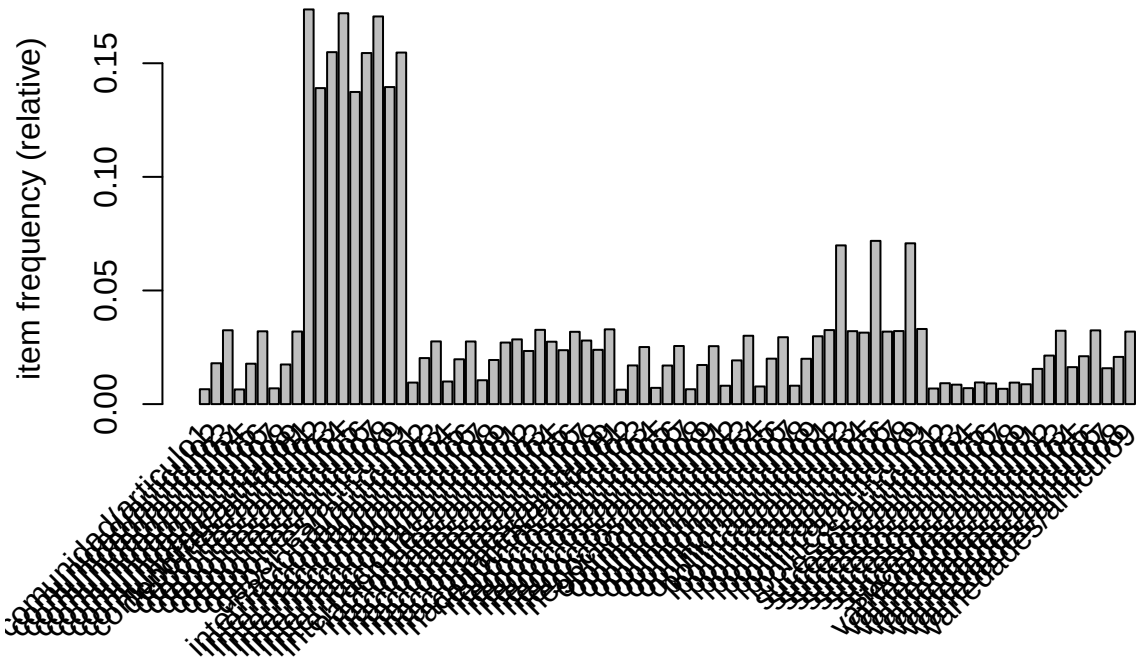
- La confianza de una regla X => Y, con respecto a T, es la proporción de transacciones que contienen X y además contienen a Y.

Partiendo de estos dos puntos, sabemos que a medida que aumente la confianza, la cantidad de reglas será menor. Por eso, elegimos un valor medio de confianza. Con respecto al soporte, vemos que debe ser bajo para obtener reglas y el mínimo valor permitido por **apriori** sin obtener alguún warning es 0.00002.

```
> # final es la lista con los articulos
> write(final, file = "transactions")
>
> # Leemos las transacciones
> transactionsRules = read.transactions("transactions", format = "basket", sep = ",")
>
> # Borrar archivo generado por write
> unlink("transactions")
>
> summary(transactionsRules)
```

```
## transactions as itemMatrix in sparse format with
##  131300 rows (elements/itemsets/transactions) and
##  81 columns (items) and a density of 0.03701259
##
## most frequent items:
## deportes/articulo1 deportes/articulo4 deportes/articulo7
##              22805              22584              22400
## deportes/articulo3 deportes/articulo9            (Other)
##              20337              20313             285201
##
## element (itemset/transaction) length distribution:
## sizes
##     1     2     3     4     5     6     7     8     9    10    11
## 17931 35545 35395 23721 11705  4770  1633   465   105    21     9
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.998   4.000  11.000
##
## includes extended item information - examples:
##                labels
## 1 comunidad/articulo1
## 2 comunidad/articulo2
## 3 comunidad/articulo3
```

```
> itemFrequencyPlot(transactionsRules)
```




```
> rules <- apriori(transactionsRules, parameter = list(supp = 2e-05, conf = 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE   2e-05      1     10
##  target   ext
##   rules FALSE
##
## 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 ...[0 item(s)] done [0.00s].
## set transactions ...[81 item(s), 131300 transaction(s)] done [0.04s].
## sorting and recoding items ... [81 item(s)] done [0.01s].
## creating transaction tree ... done [0.10s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.14s].
## writing ... [6232 rule(s)] done [0.01s].
## creating S4 object  ... done [0.03s].
```

2. Conocer los tipos de usuarios que ingresan a su página (ellos creen que son 8 tipos de usuarios) y tratar de determinar la proporción de cada tipo de usuario.

```
> rhs = inspect(unique(rules@rhs))
```

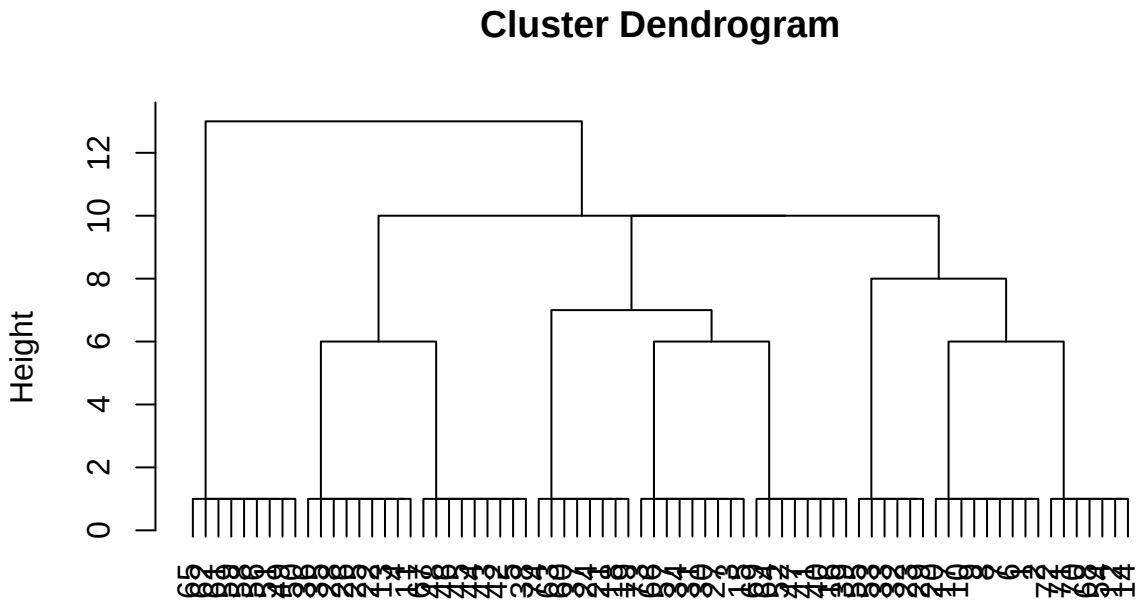
```
##    items
## 1  {deportes/articulo6}
## 2  {deportes/articulo4}
## 3  {politica/articulo5}
## 4  {nacionales/articulo4}
## 5  {variedades/articulo5}
## 6  {deportes/articulo3}
## 7  {deportes/articulo9}
## 8  {deportes/articulo7}
## 9  {deportes/articulo2}
## 10 {deportes/articulo5}
## 11 {nacionales/articulo6}
## 12 {negocios/articulo6}
## 13 {nacionales/articulo8}
## 14 {negocios/articulo1}
## 15 {politica/articulo2}
## 16 {opinion/articulo9}
## 17 {deportes/articulo1}
## 18 {comunidad/articulo6}
## 19 {comunidad/articulo9}
## 20 {deportes/articulo8}
## 21 {comunidad/articulo1}
## 22 {nacionales/articulo1}
## 23 {sucesos/articulo3}
## 24 {comunidad/articulo8}
## 25 {nacionales/articulo3}
## 26 {nacionales/articulo2}
## 27 {politica/articulo8}
## 28 {nacionales/articulo7}
## 29 {sucesos/articulo7}
## 30 {politica/articulo6}
## 31 {politica/articulo7}
## 32 {sucesos/articulo9}
## 33 {sucesos/articulo4}
## 34 {politica/articulo9}
## 35 {nacionales/articulo5}
## 36 {nacionales/articulo9}
## 37 {comunidad/articulo3}
## 38 {variedades/articulo1}
## 39 {opinion/articulo5}
## 40 {opinion/articulo2}
## 41 {opinion/articulo4}
## 42 {variedades/articulo6}
## 43 {variedades/articulo8}
## 44 {variedades/articulo4}
## 45 {variedades/articulo3}
## 46 {variedades/articulo9}
## 47 {opinion/articulo3}
## 48 {variedades/articulo7}
## 49 {internacional/articulo9}
## 50 {internacional/articulo2}
```

```
## 51 {internacional/articulo7}
## 52 {opinion/articulo1}
## 53 {sucesos/articulo6}
## 54 {negocios/articulo3}
## 55 {sucesos/articulo8}
## 56 {internacional/articulo5}
## 57 {politica/articulo4}
## 58 {internacional/articulo1}
## 59 {internacional/articulo8}
## 60 {comunidad/articulo2}
## 61 {internacional/articulo4}
## 62 {internacional/articulo6}
## 63 {comunidad/articulo5}
## 64 {opinion/articulo6}
## 65 {internacional/articulo3}
## 66 {politica/articulo1}
## 67 {variedades/articulo2}
## 68 {negocios/articulo2}
## 69 {opinion/articulo8}
## 70 {negocios/articulo9}
## 71 {negocios/articulo8}
## 72 {negocios/articulo5}
## 73 {politica/articulo3}
## 74 {comunidad/articulo7}
```

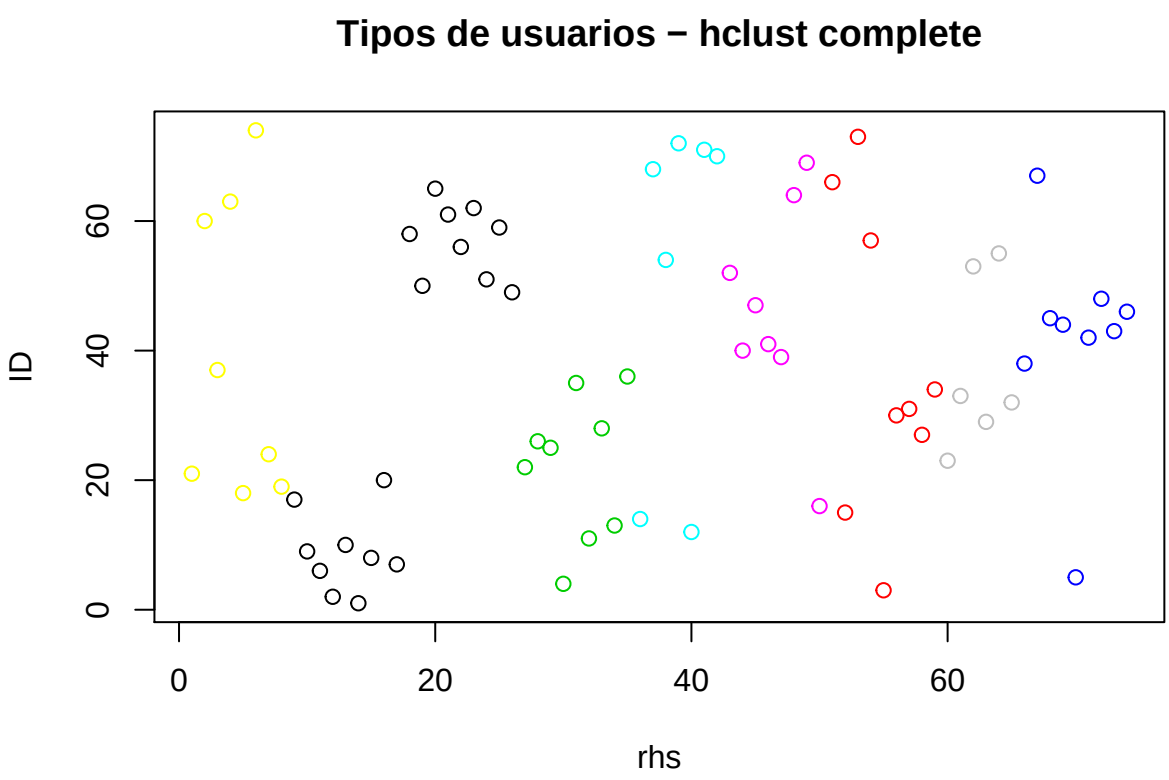
```
> class(rhs$items)
```

```
## [1] "factor"
```

```
> d <- stringdistmatrix(rhs$items)
> usersClusters = hclust(d, method = "complete")
> plot(usersClusters)
```




```
> nclases = 9
> # Cortamos
> corte = cutree(usersClusters, k = nclases)
>
> plot(as.numeric(rhs$items), c(1:74), col = corte, xlab = "rhs", ylab = "ID",
+      main = "Tipos de usuarios - hclust complete")
```




```
> typesUsers = data.frame(as.character(rhs$items), as.numeric(rhs$items), corte)
> names(typesUsers) = c("items", "ID", "class")
> head(typesUsers)
```

```
##                     items ID class
## 1   {deportes/articulo6} 14     1
## 2   {deportes/articulo4} 12     1
## 3   {politica/articulo5} 55     2
## 4 {nacionales/articulo4} 30     3
## 5 {variedades/articulo5} 70     4
## 6   {deportes/articulo3} 11     1
```

3. Dado un usuario nuevo que haya ingresado a n artículos (n variable), poder recomendar un artículo n+1 y así aumentar el compromiso del cliente con su portal web. Como usted sabe, para poder calcular las reglas necesita como entrada **MinSupport** y **MinConfianza**. Sin embargo, el cliente desconoce cuáles son estos valores en consecuencia es tarea de usted determinar y justificar los mismos de acuerdo a su criterio.

Vemos la matriz de transacciones y las reglas creadas usando **apriori**,

```
> summary(rules)
```

```
## set of 6232 rules
##
```

```
## rule length distribution (lhs + rhs):sizes
##    3    4    5    6    7
##    6 2631 2736  806   53
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   5.000   4.722   5.000   7.000
##
## summary of quality measures:
##     support            confidence         lift
##  Min.   :2.285e-05   Min.   :0.500   Min.   :  2.879
##  1st Qu.:2.285e-05   1st Qu.:0.500   1st Qu.:  3.236
##  Median :2.285e-05   Median :0.600   Median :  3.876
##  Mean   :2.720e-05   Mean   :0.628   Mean   :  6.503
##  3rd Qu.:3.046e-05   3rd Qu.:0.750   3rd Qu.:  4.854
##  Max.   :9.901e-05   Max.   :1.000   Max.   :109.966
##
## mining info:
##               data ntransactions support confidence
##  transactionsRules        131300   2e-05        0.5
```

Para hacer la recomendación al usuario se ha creado una función, que dada una lista de n art´los visitados retorna las recomendaciones por orden de confianza y el artílo que posee la mayor confianza.

```
> testFunction = c("deportes/articulo1", "deportes/articulo3", "deportes/articulo4",
+     "deportes/articulo7", "deportes/articulo8", "deportes/articulo9")
>
> recomendation = function(articlesVisited) {
+     subRules = subset(rules, subset = lhs %ain% articlesVisited)
+     sortedConf = sort(subRules, by = "confidence", decreasing = TRUE)
+     inspect(head(sortedConf))
+     recom = inspect(unique(sortedConf@rhs))
+     recom[1, 1]
+ }
>
> result = recomendation(testFunction)
```

```
##   lhs                     rhs                        support confidence     lift
## 1 {deportes/articulo1,
##    deportes/articulo3,
##    deportes/articulo4,
##    deportes/articulo7,
##    deportes/articulo8,
##    deportes/articulo9} => {deportes/articulo2} 2.284844e-05        0.5 3.595487
## 2 {deportes/articulo1,
##    deportes/articulo3,
##    deportes/articulo4,
##    deportes/articulo7,
##    deportes/articulo8,
##    deportes/articulo9} => {deportes/articulo6} 2.284844e-05        0.5 3.236222
##   items
## 1 {deportes/articulo2}
## 2 {deportes/articulo6}
```

```
> result
```

```
## [1] {deportes/articulo2}
## Levels: {deportes/articulo2} {deportes/articulo6}
```

4. Conocer las 10 visitas con mayor tiempo de estadía en la página y las 10 visitas con menor tiempo de estadía en la página.

El dataframe **times** contiene el tiempo que dura cada transacción, hacemos un **sort** y obtenemos las transacciones con mayor tiempo de estadía en la página,

```
> sortTimes = nobots[with(nobots, order(-nobots$diff)), ]
> sortTimes[1:10, 1:7]
```

```
##             X         ID               entry                exit
## 93676   93676 trans93676 2016-04-29 14:52:09 2016-04-29 18:16:33
## 7511     7511  trans7511 2016-05-03 06:50:17 2016-05-03 10:14:11
## 80516   80516 trans80516 2016-04-29 17:33:26 2016-04-29 20:49:09
## 122879 122879  trans4579 2016-05-02 02:12:09 2016-05-02 05:25:26
## 130641 130641  trans7341 2016-05-01 00:27:26 2016-05-01 03:39:14
## 66995   66995 trans66995 2016-05-02 21:14:51 2016-05-03 00:26:12
## 111953 111953  trans1653 2016-05-01 16:23:18 2016-05-01 19:33:39
## 23099   23099 trans23099 2016-05-02 06:55:59 2016-05-02 10:06:18
## 55628   55628 trans55628 2016-04-30 03:37:12 2016-04-30 06:46:53
## 48007   48007 trans48007 2016-04-30 01:26:07 2016-04-30 04:35:39
##                                        items
## 93676                          {item16,item70}
## 7511                           {item14,item63}
## 80516                  {item14,item37,item45}
## 122879                         {item57,item78}
## 130641                 {item35,item71,item81}
## 66995                    {item8,item9,item66}
## 111953                         {item39,item51}
## 23099              {item3,item5,item63,item71}
## 55628  {item4,item5,item8,item9,item11,item71}
## 48007                           {item4,item17}
##
## 93676                                                                  politica/articulo
## 7511                                                                  politica/articulo5
## 80516                                                politica/articulo5,nacionales/articulo1,
## 122879                                                                  comunidad/articul
## 130641                                               internacional/articulo8,negocios/articul
## 66995                                                    deportes/articulo8,deportes/articulo
## 111953                                                                nacionales/articul
## 23099                                        deportes/articulo3,deportes/articulo5,comunidad/articulo
## 55628  deportes/articulo4,deportes/articulo5,deportes/articulo8,deportes/articulo9,politica/articulo
## 48007                                                                   deportes/articulo
##         diff
## 93676  12264
## 7511   12234
## 80516  11743
## 122879 11597
```

```
## 130641 11508
## 66995  11481
## 111953 11421
## 23099  11419
## 55628  11381
## 48007  11372
```

Ahora, obtenemos las transacciones con menor tiempo de estadía en la página

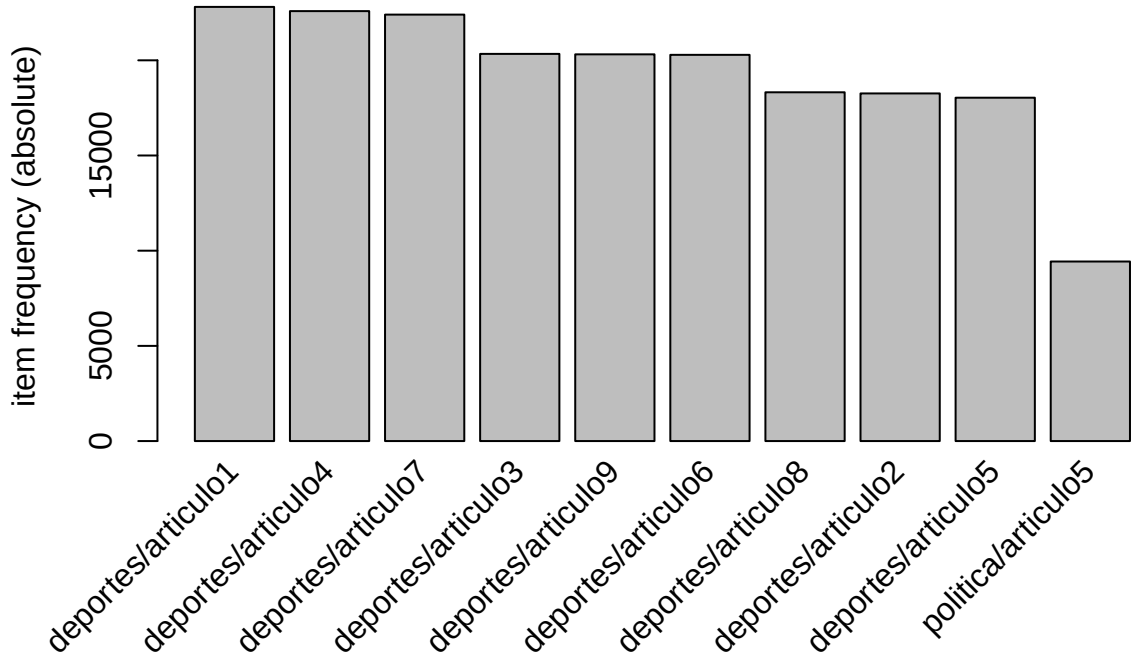
```
> lengthDF = length(nobots$ID)
> sortTimes[(lengthDF - 10):lengthDF, 1:7]
```

```
##             X        ID               entry                exit
## 120108 120108 trans1808 2016-04-29 15:46:26 2016-04-29 15:46:47
## 123367 123367   trans67 2016-05-03 06:54:36 2016-05-03 06:54:57
## 124564 124564 trans1264 2016-05-04 18:24:29 2016-05-04 18:24:50
## 125322 125322 trans2022 2016-05-01 04:20:00 2016-05-01 04:20:21
## 125695 125695 trans2395 2016-05-01 21:21:03 2016-05-01 21:21:24
## 126050 126050 trans2750 2016-04-29 10:55:05 2016-04-29 10:55:26
## 127187 127187 trans3887 2016-04-30 09:56:41 2016-04-30 09:57:02
## 129780 129780 trans6480 2016-05-02 19:32:53 2016-05-02 19:33:14
## 130053 130053 trans6753 2016-04-30 06:10:00 2016-04-30 06:10:21
## 130941 130941 trans7641 2016-05-01 08:24:00 2016-05-01 08:24:21
## 131037 131037 trans7737 2016-05-02 03:49:33 2016-05-02 03:49:54
##                                                 items
## 120108                                {item17,item45}
## 123367                  {item45,item57,item68,item80}
## 124564           {item42,item50,item62,item77,item81}
## 125322                                       {item45}
## 125695                         {item27,item64,item68}
## 126050                         {item18,item32,item35}
## 127187 {item30,item31,item34,item39,item72,item74}
## 129780                                {item10,item75}
## 130053                                {item46,item63}
## 130941                                {item18,item62}
## 131037                         {item30,item42,item61}
##
## 120108                                                                                  poli
## 123367                                                      nacionales/articulo9,comunidad/articulo3,n
## 124564                                         nacionales/articulo6,sucesos/articulo5,comunidad/articulo8
## 125322
## 125695                                                                      variedades/articulo9,na
## 126050                                                                politica/articulo9,internaciona
## 127187 internacional/articulo3,internacional/articulo4,internacional/articulo7,nacionales/articulo3,n
## 129780                                                                                           p
## 130053                                                                                          su
## 130941                                                                                         pol
## 131037                                                                 internacional/articulo3,nacio
##        diff
## 120108   21
## 123367   21
## 124564   21
## 125322   21
```

```
## 125695    21
## 126050    21
## 127187    21
## 129780    21
## 130053    21
## 130941    21
## 131037    21
```

5. Conocer las 10 transacciones con mayor número de apariciones en el dataset.

```
> # Create an item frequency plot for the top 10 items
> itemFrequencyPlot(transactionsRules, topN = 10, type = "absolute")
```




```
> topTenItems = c(sort(itemFrequency(transactionsRules), decreasing = T)[1:10])
> topTenItems
```

```
## deportes/articulo1 deportes/articulo4 deportes/articulo7
##         0.17368621         0.17200305         0.17060168
## deportes/articulo3 deportes/articulo9 deportes/articulo6
##         0.15488957         0.15470678         0.15450114
## deportes/articulo8 deportes/articulo2 deportes/articulo5
##         0.13952780         0.13906321         0.13736481
## politica/articulo5
##         0.07182788
```

Curvas ROC

1. Los scores por instancia (no necesariamente ordenados).
2. La verdadera clase de las instancias.
3. La clase target. En el caso de que nclass > 2 entonces haga un enfoque **1 vs all**.

```
> generate_ROC = function(scores, real, target) {
+   labels = c(Inf, scores, 1) #For plot
+   scores = sort(scores, decreasing = FALSE)
+   Fprev = Inf #previous score
+   FP = 0 #Number of False Positives
+   TP = 0 #Number of True Positives
+   realP = real[real == target] #Target class in real array
+   P = length(realP) #Number of positives
+   N = length(real) - P #Number of negatives
+   j = 1 #aux index
+   arr_FP = c() #Array of False Positives - X axis of ROC curve
+   arr_TP = c() #Array of True Positives - Y axis of ROC curve
+
+   for(i in 1:length(scores)) {
+     if (scores[i] != Fprev){ #If it's a different score
+       arr_FP[j] = FP/N
+       arr_TP[j] = TP/P
+       Fprev = scores[i]
+       j = j + 1
+     }
+
+     if (real[i] == target) { #i is a positive example
+       TP = TP + 1
+     }else { #i is a negative example
+       FP = FP + 1
+     }
+   }
+
+   #Last point in ROC curve - This is (1,1)
+   arr_FP[j] = FP/N
+   arr_TP[j] = TP/P
+
+   plot(arr_FP, arr_TP,
+        type = "b", #points joined by lines
+        main = "ROC Curve",
+        xlab = "FP-Rate",
+        ylab = "TP-Rate",
+        col = "blue")
+   abline(0, 1)
+   points(arr_FP, arr_TP, col = 2, pch = 20)
+
+   labels = labels[!duplicated(labels)] #Remove duplicated scores
+   text(arr_FP, arr_TP, labels, cex= 0.7, pos=4)
+ }
>
> #TESTING FUNCTION
```

```
> scores = c(0.9, 0.8, 0.7, 0.6, 0.55, 0.54, 0.53, 0.52, 0.5, 0.5, 0.5, 0.5, 0.38, 0.37, 0.36, 0.35, 0.3
>
> real = c(2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1)
>
> target = 2
>
> generate_ROC(scores, real, target)
```