Practica 1: Fundamentos de la Ciencia de Datos

Daniel Lopez Moreno Alejandro Fernandez Maceira
Alvaro Maestre Santa

October 21, 2019

1 Analisis de asociacion de datos con R

En este apartado vamos a realizar un análisis de datos con ayuda del profesor. Este análisis se resolverá mediante el algoritmo apriori. Los datos utilizados son los que hemos utilizado en clase de teoría, es decir, los de la muestra. Después se deberá de obtener las asociaciones cuyo soporte **sea igual o superior al 50%** y cuya **confianza sea igual o superior al 80%**.
En primer lugar, debemos de tener la librería **arules**, si no la tenemos, tendremos que proceder a instalarla. También añadiremos la libreria **arulesViz** para mostrar las asociaciones de forma gráfica más adelante.

```
> install.packages("arules")
> install.packages("arulesViz")
> library(arules)
> library("arulesViz")
```

Nuestras cestas de la compra disponibles sobre las que vamos a actuar son las siguientes:

```
> c1 = c("pan","agua","leche","naranjas")
> c2 = c("pan","agua","café","leche")
> c3 = c("pan","agua","leche")
> c4 = c("pan","café","leche")
> c5 = c("pan","agua")
> c6 = c("leche")
```

A continuación, procedemos a generar una matriz de 0s y 1s, donde sobre cada elemento de cada cesta 1 significa que se ha comprado y 0 significa que no:

```
> muestra<-Matrix(c(1,1,0,1,1,1,1,1,1,0,1,1,0,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0,1,0),6,5,
+ byrow=T,dimnames=list(c("suceso1","suceso2","suceso3","suceso4","suceso5","suceso6"),
+ c("Pan","Agua","Cafe","Leche","Naranjas")),sparse=T)
> muestra

6 x 5 sparse Matrix of class "dgCMatrix"
        Pan Agua Cafe Leche Naranjas
suceso1   1    1    .     1        1
suceso2   1    1    1     1        .
suceso3   1    1    .     1        .
```

```
suceso4   1    .    1     1        .
suceso5   1    1    .     .        .
suceso6   .    .    .     1        .
```

Hacemos la transpuesta, para posteriormente generar las transacciones de nuestra muestra:

```
> muestraCMatriz<-as(muestra,"nsparseMatrix")
> transpuestaMatriz <-t(muestraCMatriz)
> transpuestaMatriz

5 x 6 sparse Matrix of class "ngCMatrix"
         suceso1 suceso2 suceso3 suceso4 suceso5 suceso6
Pan            |       |       |       |       .
Agua           |       |       .       |       .
Cafe           .       |       .       |       .
Leche          |       |       |       |       .
Naranjas       |       .       .       .       .
```

Ahora generamos las transacciones de nuestra muestra:

```
> transacciones<-as(transpuestaMatriz, "transactions")
> summary(transacciones)

transactions as itemMatrix in sparse format with
 6 rows (elements/itemsets/transactions) and
 5 columns (items) and a density of 0.5666667

most frequent items:
     Pan    Leche     Agua     Cafe Naranjas  (Other)
       5        5        4        2        1        0

element (itemset/transaction) length distribution:
sizes
1 2 3 4
1 1 2 2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.250   3.000   2.833   3.750   4.000

includes extended item information - examples:
  labels
1    Pan
2   Agua
3   Cafe

includes extended transaction information - examples:
  itemsetID
1   suceso1
2   suceso2
3   suceso3
```

A continuación, procedemos a utilizar el algoritmo **Apriori**, en este caso ponemos el porcentaje del soporte y la confianza

```
> asociaciones<-apriori(transacciones, parameter = list(support=0.5,confidence=0.8))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.5      1
 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: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

> inspect(asociaciones)

    lhs             rhs     support   confidence lift count
[1] {}           => {Leche} 0.8333333 0.8333333  1.00 5
[2] {}           => {Pan}   0.8333333 0.8333333  1.00 5
[3] {Agua}       => {Pan}   0.6666667 1.0000000  1.20 4
[4] {Pan}        => {Agua}  0.6666667 0.8000000  1.20 4
[5] {Leche}      => {Pan}   0.6666667 0.8000000  0.96 4
[6] {Pan}        => {Leche} 0.6666667 0.8000000  0.96 4
[7] {Agua,Leche} => {Pan}   0.5000000 1.0000000  1.20 3
```

Como podemos observar en el resultado del algoritmo apriori, hemos encontrado 7 asociaciones. Por lo tanto, podemos concluir en base al mínimo de soporte del 50% y al mínimo de confianza 80lo siguiente:

- **(Agua)->(Pan)**:Cuando alguien compra **Agua**, adquirirá también **Pan**.
- **(Pan)->(Agua)**:Cuando alguien compra **Pan**, adquirirá también **Agua**.
- **(Leche)->(Pan)**:Cuando alguien compra **Leche**, adquirirá también **Pan**.
- **(Pan)->(Leche)**:Cuando alguien compra **Pan**, adquirirá también **Leche**.
- **(Agua, Leche)->(Pan)**:Cuando alguien compra **Agua y Leche**, adquirirá también **Pan**.

Los demas casos no se muestran, ya que no cumplen con los umbrales establecidos para la confianza y el soporte

2 Desarrollo por parte del grupo

2.1 Analisis de datos a través de un .txt

En este apartado vamos a realizar un análisis de datos similar al del apartado anterior, pero esta vez los datos los leeremos de un archivo .txt y cambiaremos el umbral de **soporte de 50% a 40%**, y el umbral de **confianza de 80% a 90%**.
Empezaremos asignando la matriz del archivo .txt a la variable **ventas**:

```
> ventas<-as.matrix(read.table("ventaCoches.txt"))
> ventas
```

```
  Far Ala Tec Nav Blu Con
1   1   0   0   1   1   1
2   1   0   1   0   1   1
3   1   0   0   1   0   1
4   1   0   1   1   1   0
5   1   0   0   0   1   1
6   0   0   0   1   0   0
7   1   0   0   0   1   1
8   0   1   1   0   0   0
```

A continuación, vamos a generar nuestra matriz dispersa, una muestra a partir de dicha matriz y la transpuesta de esta muestra:

```
> ventasM<-Matrix(ventas,sparse=T)
> muestraVentas<-as(ventasM,"nsparseMatrix")
> transVentas <-t(muestraVentas)
> transVentas
```

```
6 x 8 sparse Matrix of class "ngCMatrix"
    1 2 3 4 5 6 7 8
Far | | | | | . | .
Ala . . . . . . . |
Tec . | . | . . . |
Nav | . | | . | . .
Blu | | . | | . | .
Con | | | . | . | .
```

Ahora vamos a generar todas las transacciones posibles que existan en nuestra muestra:

```
> transacciones<-as(transVentas, "transactions")
> summary(transacciones)
```

```
transactions as itemMatrix in sparse format with
 8 rows (elements/itemsets/transactions) and
 6 columns (items) and a density of 0.5

most frequent items:
    Far     Blu     Con     Nav     Tec (Other)
```

```
      6       5       5       4       3       1

element (itemset/transaction) length distribution:
sizes
1 2 3 4
1 1 3 3

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    2.75    3.00    3.00    4.00    4.00

includes extended item information - examples:
  labels
1    Far
2    Ala
3    Tec

includes extended transaction information - examples:
  itemsetID
1         1
2         2
3         3
```

Y por último, procedemos a utilizar el algoritmo **Apriori** para filtrar solo aquellas asociaciones con un mínimo de soporte del 40% y un mínimo de confianza del 90%:

```
> asociaciones<-apriori(transacciones, parameter = list(support=0.4,confidence=0.9))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.9    0.1    1 none FALSE            TRUE       5     0.4      1
 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: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 8 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(asociaciones)
```

```
    lhs          rhs   support confidence lift     count
[1] {Con}     => {Far} 0.625   1          1.333333 5
[2] {Blu}     => {Far} 0.625   1          1.333333 5
[3] {Blu,Con} => {Far} 0.500   1          1.333333 4
```

Como se puede comprobar por la salida del algoritmo Apriori, hemos encontrado 3 asociaciones. Hemos determinado con un mínimo de soporte del 40% y de confianza del 90% que:

- **(Con)->(Far)**: Cuando alguien compra **Control de velocidad**, adquirirá también **Faros Xenon**.
- **(Blu)->(Far)**: Cuando alguien compra **Bluetooth**, adquirirá también **Faros Xenon**.
- **(Blu,Con)->(Far)**:Cuando alguien compra **Bluetooth y Control de velocidad**, adquirirá también **Faros Xenon**.

Entre el resto de los extras disponibles en los coches no hemos encontrado ninguna asociación. Es probable que si se rebajasen los umbrales de soporte y confianza apareciese alguna asociación más.

2.2 Análisis con modificaciones

En esta parte de la práctica se analizará una base de datos sobre dónde viven las personas. Para ello se obtendrán el soporte y confianza mediante el algoritmo apriori y así saber qué grupos son más probables que vivan en ciudad o en campo. La librería utilizada para todo el apartado es **arules**, la cual ya se ha instalado previamente en otros ejercicios. Lo primero que hay que hacer es leer el archivo de bases de datos. En este caso es un fichero .csv llamado data.csv. Como siempre, se utiliza la instrucción **read.csv**.

```
> data = read.csv("basic_income.csv")
```

Sin embargo, esta base de datos contiene columnas que no nos interesan para el análisis. Para saber qué columnas nos interesan, la instrucción **colnames** nos dirá el nombre de las columnas de la base de datos.

```
> colnames(data)
```

```
 [1] "country_code"
 [2] "uuid"
 [3] "age"
 [4] "gender"
 [5] "rural"
 [6] "dem_education_level"
 [7] "dem_full_time_job"
 [8] "dem_has_children"
 [9] "question_bbi_2016wave4_basicincome_awareness"
[10] "question_bbi_2016wave4_basicincome_vote"
[11] "question_bbi_2016wave4_basicincome_effect"
[12] "question_bbi_2016wave4_basicincome_argumentsfor"
```

```
[13] "question_bbi_2016wave4_basicincome_argumentsagainst"
[14] "age_group"
[15] "weight"
```

Ahora, se pueden eliminar las columnas que no necesitamos. Esto se consigue añadiendo lo siguiente en la instrucción **read.csv**.

```
> data = read.csv("basic_income.csv")[,c('gender','rural','dem_education_level',
+ 'dem_has_children','age_group')]
```

Las columnas con las que nos quedamos son las siguientes:

- **gender**: El género de la persona, que puede ser **male** (hombre) o **female** (mujer).
- **rural**: El lugar de residencia, la ciudad **(urban)** o el campo **(rural)**.
- **dem-education-level**: El nivel de estudios de la persona, que puede ser **low** (bajo), **medium** (medio) o **high** (alto).
- **dem-has-children**: Si la persona tiene hijos, **yes** o **no**.
- **age-group**: El rango de edad de la persona, que puede ser **14-25**, **26-39**,**40-65**.

Una vez que tenemos el fichero con las columnas que nos interesan, debemos guardarlo en un archivo .csv para poder utilizarlo posteriormente en una instrucción especial. Esto se consigue con la **write.csv**. Esta instrucción toma como parámetros la variable con el fichero modificado (data), el nombre del fichero que queremos y un parámetro que indica si la primera fila contendrá los nombres de las columnas o no.

```
> write.csv(data,"basic_income2.csv",row.names=T)
```

Ahora, ya podemos generar las transacciones necesarias para pasárselas al algoritmo apriori. Las transacciones se realizan con la instrucción **read.transactions** de la librería **arules**.

```
> txn = read.transactions(file='basic_income2.csv',format='basket',sep=',',cols=1,rm.dupli

distribution of transactions with duplicates:
  1
194
```

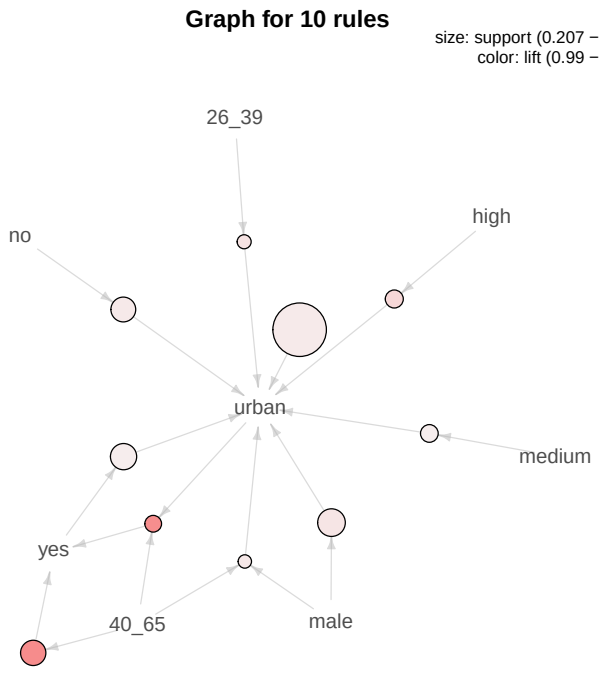
Los parámetros son la ruta del archivo recién creado en el paso anterior; la forma del archivo, que puede ser single o basket, en este caso basket ya que cada línea es una transacción separada por comas; el método de separación; y si la primera columna está formada por un id o número de transacción, no la tiene en cuenta para realizar las transacciones. Después de que R genere las transacciones que se hayan encontrado en la base de datos, hay que aplicar el algoritmo apriori para obtener el soporte y confianza de las asociaciones. En este caso, el soporte umbral que queremos será 0.2, y la confianza de 0.7, ya que con más soporte no hay muchas asociaciones que nos interesen.

```
> tn = apriori(txn,parameter=list(sup=0.2,conf=0.7))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.7    0.1    1 none FALSE            TRUE       5     0.2      1
 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: 1930

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[16 item(s), 9650 transaction(s)] done [0.00s].
sorting and recoding items ... [11 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].
```

Ahora solo falta mostrar estos resultados con la instrucción **plot** e **inspect**.

```
> plot(tn, method = "graph")
```



Con el método plot podemos mostrar gráficamente todas las asociaciones que hemos creado con el algoritmo Apriori, utilizamos el método "graph" para

crear un grafo de nodos. Como se puede observar, la mayoria de las asociaciones marcan el campo **urban**, algunas de ellas combinadas en un nodo intermedio común que posteriormente apunta al campo final.

```
> inspect(tn)
```

```
     lhs               rhs     support   confidence lift      count
[1]  {}             => {urban} 0.7127461 0.7127461  1.0000000 6878
[2]  {26_39}        => {urban} 0.2156477 0.7397796  1.0379286 2081
[3]  {high}         => {urban} 0.2654922 0.7834862  1.0992501 2562
[4]  {medium}       => {urban} 0.2615544 0.7054220  0.9897241 2524
[5]  {no}           => {urban} 0.3513990 0.7170649  1.0060594 3391
[6]  {40_65}        => {yes}   0.3573057 0.7042484  1.3460085 3448
[7]  {yes}          => {urban} 0.3691192 0.7054862  0.9898142 3562
[8]  {male}         => {urban} 0.3868394 0.7328229  1.0281683 3733
[9]  {40_65,urban} => {yes}   0.2492228 0.7023949  1.3424659 2405
[10] {40_65,male}  => {urban} 0.2071503 0.7248006  1.0169127 1999
```

Las asociaciones obtenidas son las siguientes

- **(26-39)->(urban)**: Si alguien está en el grupo de edad **26-39**, vivirá en **ciudad**.
- **(high)->(urban)**: Si el nivel de estudios es **alto**, vivirá en **ciudad**.
- **(medium)->(urban)**: Si el nivel de estudios es **medio**, vivirá en **ciudad**.
- **(no)->(urban)**: Si no tienen **hijos**, vivirá en **ciudad**.
- **(40-65)->(yes)**: Si alguien está en el grupo de edad **40-65**, tendrá **hijos**.
- **(yes)->(urban)**: Si alguien tiene **hijos**, vivirá en **ciudad**.
- **(male)->(urban)**: Si es **hombre**, vivirá en **ciudad**.
- **(40-65,urban)->(yes)**: Si está entre **40-65** y vive en **ciudad**, tendrá **hijos**.
- **(40-65,male)->(urban)**: Si está entre **40-65** y es **hombre**„ vivirá en **ciudad**.

A la vista de estos resultados, podemos llegar a varias conclusiones según la regla que analicemos. Por lo general, las asociaciones con mayor soporte y confianza pertenecen a los hombres y la ciudad, siendo la asociación hombre -> ciudad la que más soporte tiene entre todas las analizadas. Específicamente, podemos decir que si alguien está entre 40-65 años, tendrá hijos, que es otra asociación con bastante soporte. Por último, si establecemos el umbral de soporte y confianza en valores de ejercicios previos(0.5 y 0.8), obtenemos lo siguiente.

```
> tn = apriori(txn,parameter=list(sup=0.5,conf=0.8))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.5      1
 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: 4825

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[16 item(s), 9650 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 done [0.00s].
writing ... [0 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> inspect(tn)
```

No hay asociaciones con soporte mayor a 0.5, lo que indica que no se puede estar muy seguro de la probabilidad de ocurrencia de un evento tras otro en esta base de datos.