0.1. Análisis de cesta de la compra para productos lácteos

0.1.1. Introducción

El objetivo principal de este apartado es aplicar un análisis de cesta de la compra para un conjunto de datos con una muestra de 7801 tickets que incluyen 4631 artículos distintos.

Aplicaremos esta técnica para descubrir los patrones de compra de los clientes y tratar así de identificar las relaciones existentes entre productos a la hora de comprar.

Para llevar a cabo este estudio haremos uso de la librería *arules*, un entorno creado para la identificación de reglas de asociación y conjuntos de items frecuentes.

0.1.2. Lectura y descripción de los datos

Los datos corresponden a una muestra de una base de datos anonimizadas empleadas en el proyecto *Advanced Promotional Engine*, dirigido por el tutor del TFG y se encuentran en formato dataFrame. Contienen información correspondiente a transacciones de una muestra de tickets de una cadena de supermercados.

Cada fila de nuestro conjunto de datos corresponde a una línea de un ticket y por tanto, una fila se corresponde con la venta de un determinado producto. Por este motivo, para una única transacción encontraremos tantas filas como productos diferentes se hayan comprado, y también encontramos registrada la cantidad de unidades comprada de cada producto.

En este conjunto de datos inicial encontramos las siguientes variables:

- **Idticket**: Variable numérica que identifica unívocamente a cada ticket
- **Linea**: Variable numérica con la línea correspondiente del ticket
- **Item**: Item concreto
- **Cantidad**: Se trata de la cantidad de unidades que se ha comprado de un determinado ítem en una transacción concreta

A continuación mostramos brevemente algunas de las filas de nuestros datos:

Idticket	linea	item	cantidad
1	1	11802	12
1	2	21662	12
1	3	27959	3

Sin embargo, para aplicar un análisis de cesta de la compra necesitamos que nuestros datos estén en *formato cesta* (formato basket). Para obtener este formado, es necesario que cada línea del nuevo conjunto de datos corresponda a una única transacción, es decir, que en cada fila estén contenidos todos los productos que se refieren a sola compra.

Por ello, únicamente necesitamos una columna en la que tengamos recogidos todos los items perteneceientes a cada transacción, por lo que tendremos así tantas filas como transacciones, es decir, tantas filas como tickets generados.

Existen un total de 4631 productos y 7801 transacciones.

	Idticket	Items
7452	7508	11865,11865,26935
7162	7217	26545,22324,14654,25509,22541
7269	7324	24147,28212,22350
1004	1013	12033,34810,33229
623	628	27942,19195
7049	7103	17216,26129,27379,27382,1134,12337,18900,19156,19155,20118,1046,19626,1084,1033
2693	2709	23346,25825,19196,61957,13433,16460
934	941	26131,26131,21713,21713,17224,17224
4496	4523	25072,18962,18960,25038,10026,28863,28863,10026,24231,20484,26229,10026
2948	2968	19200,10566

Procedemos a transformar el conjunto de datos inicial en uno nuevo para poder aplicarle funciones de la librería *arules*.

A continuación vamos a ver las seis últimas filas de nuestro nuevo conjunto de datos.

```
TicketsAgrupados<-transacciones %>% group_by(Idticket) %>% select(item,linea)

# Con el siguiente comando, conseguimos agrupar en una misma columna
#  todos los items que corresponden a una misma transacción
DatosBasket <- ddply(TicketsAgrupados, c("Idticket"),
                             function(TicketsAgrupados)
                               paste(TicketsAgrupados$item, collapse = ","))


# DatosBasket1 <- DatosBasket %>%
# rename("Transaccion" = V1) # Me daba error al compilar

DatosBasket1 <- DatosBasket
names(DatosBasket1) <- c("Idticket","Items")


set.seed(1234)
DatosBasket1[sample(nrow(DatosBasket1),size = 10),] %>%
  kable(booktabs=TRUE) %>%
  kable_styling(latex_options = c("striped","scale_down"))
```

Observación: Es necesario mencionar que en cada transacción vemos una cota inferior de los items que han sido comprados, por ejemplo, vemos que en la transacción 7796 se han comprado los productos 15457 y 26978, pero no hemos considerado cuántos items de cada producto pertenecieron a esta compra. Esto se debe a que posteriormente al hacer uso de la función *read.transacctions*, esta no va a considerar el número de items de cada producto que han sido comprados, sino que únicamente va a utilizar el que hayan sido o no comprados.

La librería arules contiene una serie de funciones para poder encontrar reglas de asociación entre productos, y por este motivo, a pesar de tener información sobre el número de items que son comprados de cada producto para una misma transacción, esta información no nos va a ser de utilidad a la hora de analizar las relaciones entre los productos.

Por último, después de haber transformado nuestros datos, guardamos la columna correspondiente a los productos en un archivo *.csv* para poder posteriormente leerlo correctamente haciendo uso de la función *read.transactions* perteneciente a la librería *arules*.

0.1.3. Análisis de ventas

Leemos los datos y vemos una primera información a modo resumen de éstos:

```
## transactions as itemMatrix in sparse format with
##  7801 rows (elements/itemsets/transactions) and
##  4631 columns (items) and a density of 0.001169421
##
## most frequent items:
##    1033   28716    1096   26785   24347 (Other)
##     507     451     358     294     266   40371
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1847 1285  952  708  552  389  336  277  221  177  153  109   98  102   70   64
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##   71   50   50   40   34   22   32   20   21   13   14   12    9    9    8    6
##   33   34   35   36   37   38   39   40   41   42   43   44   45   48   49   50
##    5    1    4    8    3    2    3    4    1    1    2    2    1    1    1    2
##   52   54   55   56   59   60   72   82
##    1    1    2    1    1    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   5.416   7.000  82.000
##
## includes extended item information - examples:
##   labels
## 1    100
## 2  10002
## 3  10004
```

Observamos que hay un total de 7801 transacciones y 4631 artículos vendidos. Las transacciones son los subconjuntos de estos 4631 artículos.

En el resumen de los datos podemos ver otra información que nos puede ser útil:

- Density: Se trata del número total de artículos que se han comprado entre el número total de artículos existentes, en nuestro caso: *densidad=0.001169*.
- Productos más frecuentes:

Tabla 1: Productos más frecuentes

ID Producto	Cota inferior de unidades vendidas
1033	507
28716	451

ID Producto	Cota inferior de unidades vendidas
1096	358
26785	294
24347	266
Otro	40371

- La media de productos por transacción es de 5 artículos. Además, cabe destacar que en el 75 % de las transacciones se han comprado 7 artículos o menos.
- Tamaño de las transacciones: un total de 1847 transacciones fueron de un único artículo y 1285 transacciones fueron de dos artículos, indicando estos resultados que la mayoría de clientes compraron entre 1 y 2 artículos. La transacción con más productos diferentes ha sido una transacción con 82 artículos.

Nota: Recordemos que el número de items es una cota inferior, es decir, para una transacción con un único artículo, sabemos que se compró al menos una vez, pero no sabemos cuantas unidades se compraron.

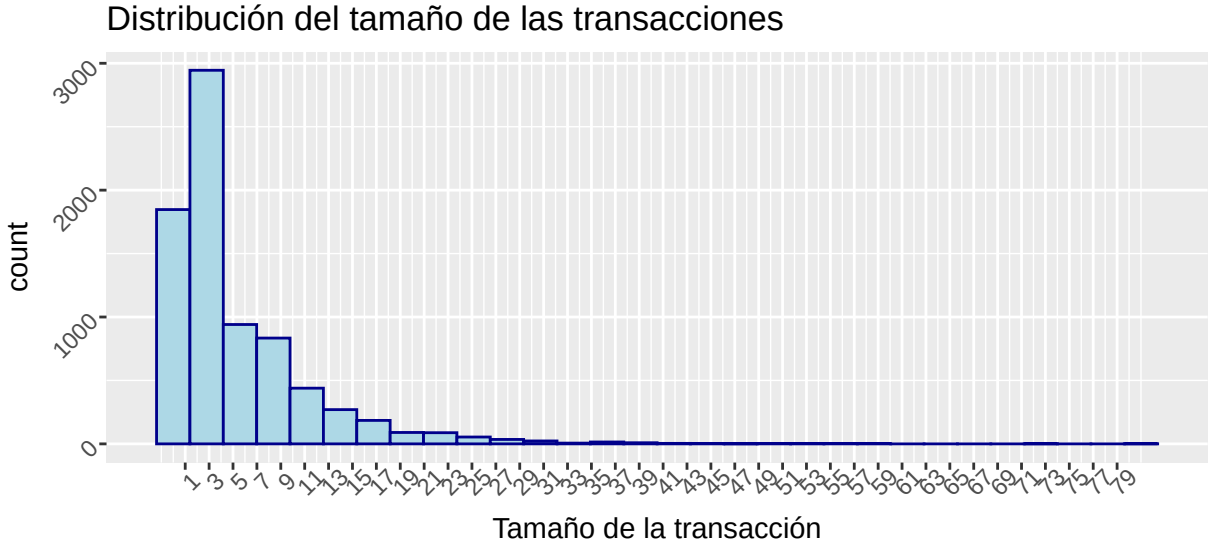
A continuación vamos a ver una lista con algunas transacciones:

```
## [1] "{16591,22852,22858,24232}" "{15005,20191,20676}"
## [3] "{24945,26737,33702,33753}" "{21134,21135}"
## [5] "{26042,26785}"             "{23570,24347}"
```

- Estudio de los cuantiles y la distribución del tamaño de las transacciones:

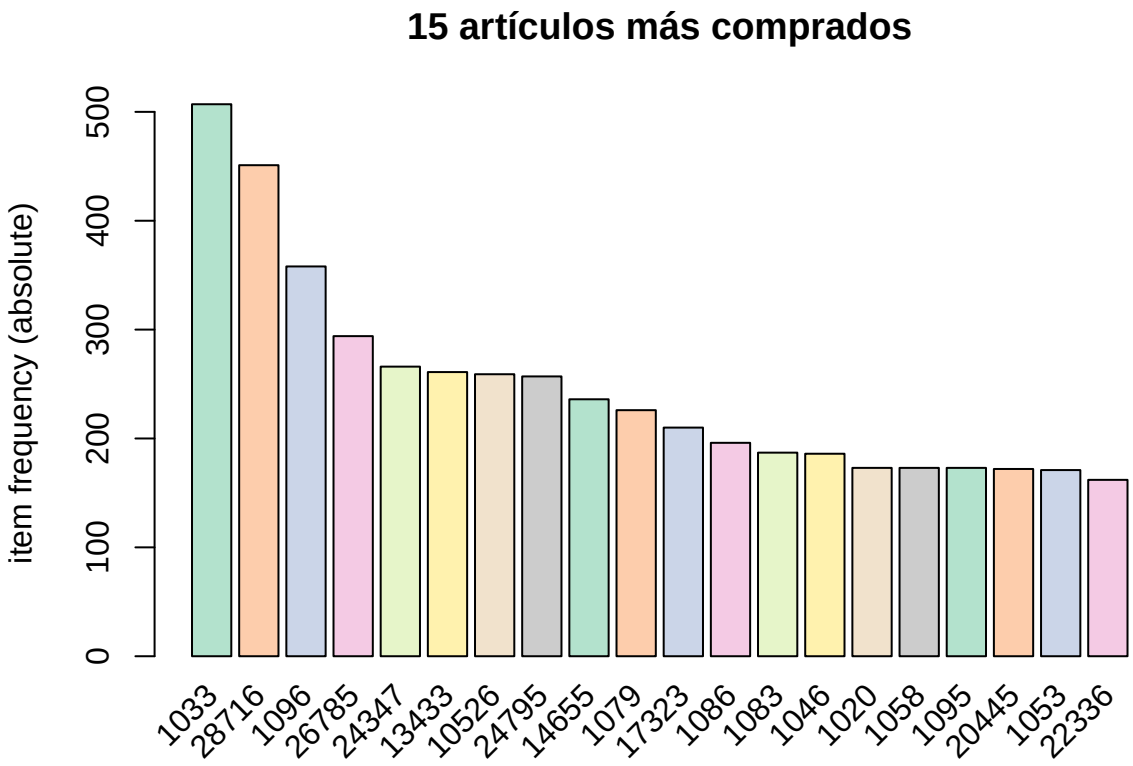
```
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
##    1    1    1    2    2    3    4    6    8   13   82
```

Vamos a mostrar gráficamente la distribución de los tamaños de las transacciones:



La mayoría de los clientes compra entre 1 y 3 productos y el 90 % de ellos compra como máximo 9 productos diferentes.

Ahora podemos ver gráficamente cuáles han sido los 15 artículos más vendidos y la frecuencia absoluta de transacciones en las que aparece ese artículo.



Vemos que los productos 1033, 28716 y 1096 son los tres artículos más vendidos de entre todos los existentes.

También es importante estudiar como se distribuye el soporte de los productos individuales, para posteriormente establecer un límite de soporte. Esto se puede calcular fácilmente con un análisis de los items más frecuentes (con mayor soporte) dentro del conjunto de transacciones.

```
frec_items <- itemFrequency(x = TransBasket, type = "relative")
frec_items %>% sort(decreasing = TRUE) %>% head(5)
```

```
##       1033      28716       1096      26785      24347
## 0.06499167 0.05781310 0.04589155 0.03768748 0.03409819
```

En el listado anterior podemos observar que el 6.5 % de las transacciones contiene al producto 1033, el 5.7 % al producto 28716 y que en el 4.58 % de éstas se ha vendido el producto 1096.

Vemos que el soporte individual de los items son bastante bajos, ya que tenemos un conjunto de datos con un gran número de transacciones y muchos productos diferentes.

Después de haber visto los aspectos más destacables de nuestros datos, procedemos a la aplicación del algoritmo a priori.

0.1.4. Aplicación del algoritmo a priori

Este algoritmo ya fué descrito en el desarrollo teórico, y nos permitirá generar una serie de reglas de asociación. Como hemos mencionado a lo largo de la descripción de este apartado práctico, el paquete *arules* también implementa el algoritmo *Apripri* para identificar itemsets frecuentes y descubre reglas de asociación con la función *apriori*, donde indicaremos una serie de parámetros: soporte, confianza, tamaño mínimo o máximo y el tipo de asociación requerida (target)

0.1.4.1. Itemsets

En primer lugar, vamos a extraer itemsets formados por al menos dos items que hayan sido comprado almenos 30 veces.

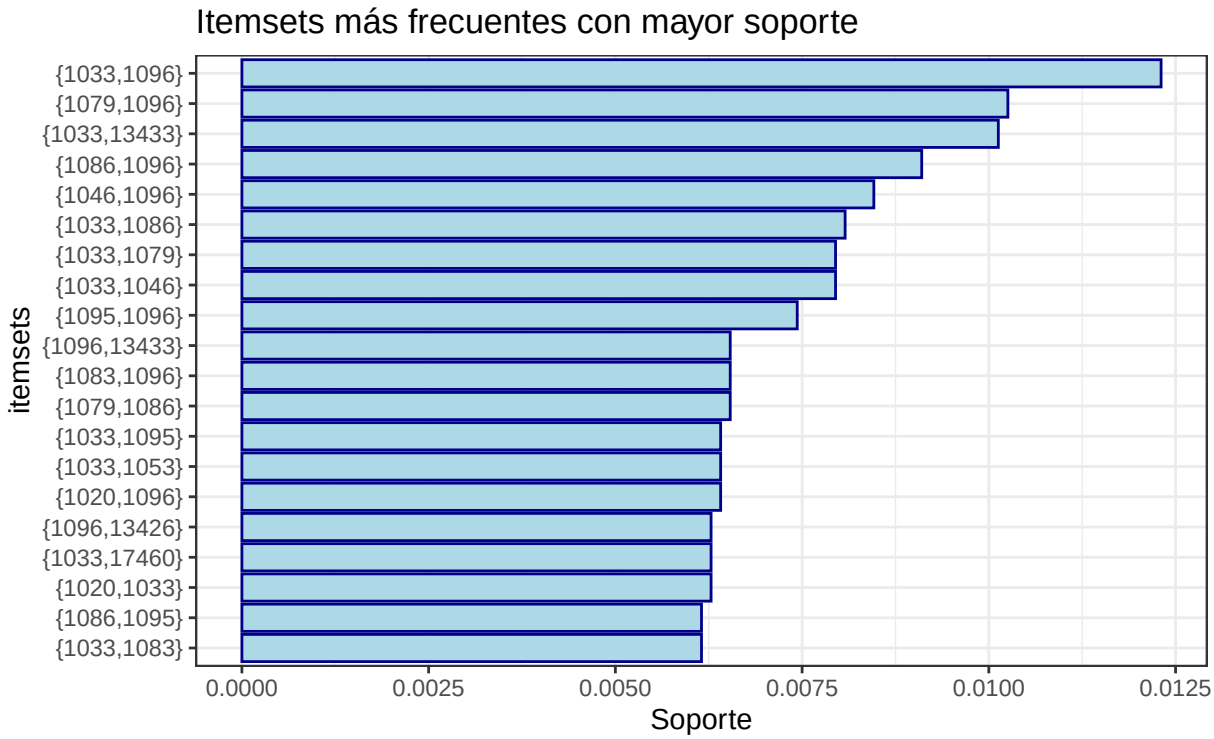
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime     support minlen
##          NA    0.1    1 none FALSE            TRUE       5 0.003845661      2
##  maxlen            target  ext
##      80 frequent itemsets TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[4631 item(s), 7801 transaction(s)] done [0.06s].
## sorting and recoding items ... [253 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## sorting transactions ... done [0.00s].
## writing ... [64 set(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
## set of 64 itemsets
##
## most frequent items:
##    1033    1096    1079    1086    1083 (Other)
##      25      15       9       9       8      63
##
## element (itemset/transaction) length distribution:sizes
##  2  3
## 63  1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.016   2.000   3.000
##
## summary of quality measures:
##     support         transIdenticalToItemsets     count
##  Min.   :0.003846   Min.   :0.0000000        Min.   :30.00
##  1st Qu.:0.004198   1st Qu.:0.0000000        1st Qu.:32.75
##  Median :0.004871   Median :0.0000000        Median :38.00
##  Mean   :0.005482   Mean   :0.0001182        Mean   :42.77
##  3rd Qu.:0.006281   3rd Qu.:0.0001282        3rd Qu.:49.00
##  Max.   :0.012306   Max.   :0.0010255        Max.   :96.00
##
```

```
## includes transaction ID lists: FALSE
##
## mining info:
##         data ntransactions     support confidence
##  TransBasket          7801 0.003845661          1
```

Hemos encontrado un total de 64 itemsets con un soporte mayor al soporte mínimo indicado de 0.0038457. La mayoría de estos conjuntos de items están formados por dos items.

De estos itemsets encontrados, vamos a proceder a mostrar aquellos con mayor soporte:



Del gráfico anterior podemos observar que la dupla $\{1033, 1096\}$ tiene el mayor soporte, es decir, que se trata del itemset que más veces ha sido vendido. El soporte, 0.012306115 indica que estos productos han sido vendidos un 1.23 % del total de transacciones.

Vamos a filtrar los itemsets para seleccionar aquellos que contienen los productos 1033 y 1096.

```
##     items             support     transIdenticalToItemsets count
## [1] {1033,1096}       0.012306115 0.0003845661             96
## [2] {1033,1086,1096}  0.003845661 0.0000000000             30
```

La mayoría de veces, un total de 96, estos dos items han sido comprados exclusivamente, mientras que han sido comprado con otro producto, el 1086 hasta en 30 ocasiones.

0.1.4.2. Reglas de asociación

A continuación vamos a crear reglas de asociación de la misma forma que hemos identificado los itemsets, pero indicando un valor mínimo para el parámetro *confianza*, en este caso, del 50 %. Vamos a imponer la obtención de reglas cuyos itemsets hayan sido comprado almenos 20 veces.

```
##      lhs              rhs     support     confidence coverage    lift      count
## [1]  {1084}       => {1083}  0.002563774 0.5000000  0.005127548 20.858289 20
## [2]  {14240}      => {14655} 0.002820151 0.5000000  0.005640303 16.527542 22
## [3]  {16037}      => {1096}  0.003076529 0.5000000  0.006153057 10.895251 24
## [4]  {1033,13426} => {1096}  0.002563774 0.5128205  0.004999359 11.174617 20
## [5]  {1020,1046}  => {1096}  0.002691963 0.6774194  0.003973850 14.761308 21
## [6]  {1020,1096}  => {1033}  0.003332906 0.5200000  0.006409435  8.001026 26
## [7]  {1020,1033}  => {1096}  0.003332906 0.5306122  0.006281246 11.562308 26
## [8]  {1046,1086}  => {1096}  0.002563774 0.6896552  0.003717472 15.027933 20
## [9]  {1046,1079}  => {1096}  0.002820151 0.6285714  0.004486604 13.696887 22
## [10] {1086,1095}  => {1096}  0.003076529 0.5000000  0.006153057 10.895251 24
## [11] {1079,1095}  => {1096}  0.002820151 0.5789474  0.004871170 12.615554 22
## [12] {1079,1086}  => {1096}  0.003461095 0.5294118  0.006537623 11.536149 27
## [13] {1079,1083}  => {1096}  0.002563774 0.5405405  0.004742982 11.778650 20
## set of 13 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
##  3 10
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   2.769   3.000   3.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.002564   Min.   :0.5000   Min.   :0.003717   Min.   : 8.001
##  1st Qu.:0.002564   1st Qu.:0.5000   1st Qu.:0.004743   1st Qu.:11.175
##  Median :0.002820   Median :0.5294   Median :0.005128   Median :11.779
##  Mean   :0.002899   Mean   :0.5545   Mean   :0.005315   Mean   :13.025
##  3rd Qu.:0.003077   3rd Qu.:0.5789   3rd Qu.:0.006153   3rd Qu.:14.761
##  Max.   :0.003461   Max.   :0.6897   Max.   :0.006538   Max.   :20.858
##      count
##  Min.   :20.00
##  1st Qu.:20.00
##  Median :22.00
##  Mean   :22.62
##  3rd Qu.:24.00
##  Max.   :27.00
##
## mining info:
##         data ntransactions     support confidence
##  TransBasket          7801 0.002563774        0.5
```

Se han encontrado un total de 13 reglas. Sin embargo, el algoritmo nos está recomendando comprar los productos 1096 y 1033 en la mayoría de las reglas. Se trata de algunos de los productos más vendidos, por lo que no tiene sentido. Estos productos no tienen problemas para su venta, por lo que nuestro objetivo es buscar reglas que recomienden productos que se han vendido en menor volumen.

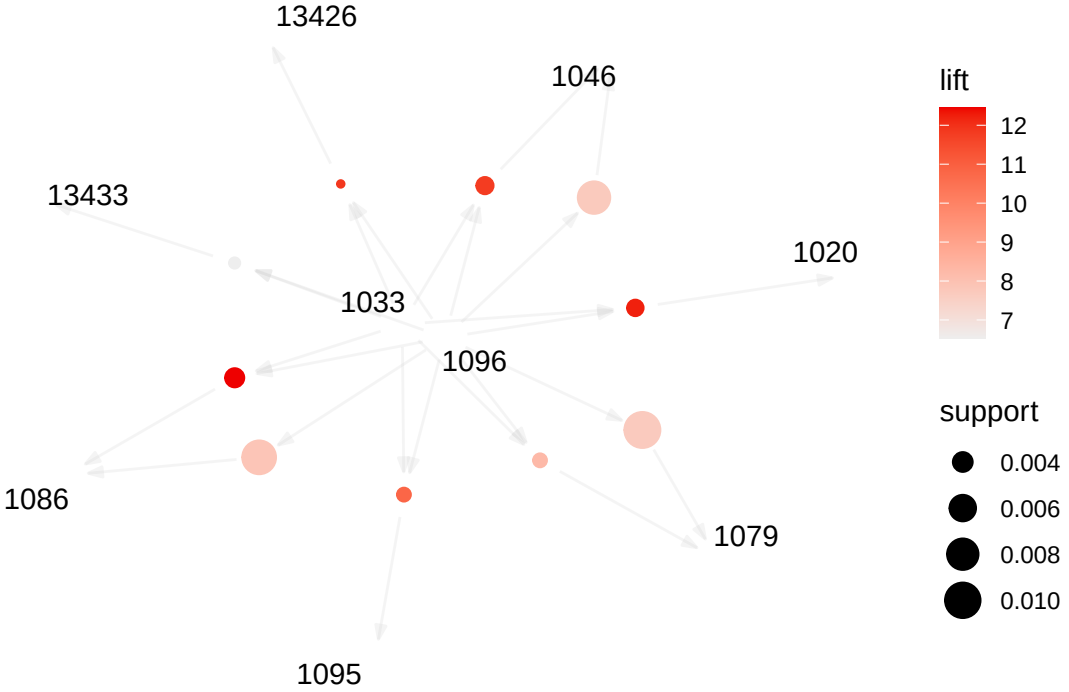
Para ello, vamos a modificar las reglas, bajando el valor de la confianza y obligando al algoritmo a tener los productos más frecuentes a la izquierda, en la parte de *lhs*.

Veamos las nuevas reglas de asociación:

```
##      lhs             rhs     support     confidence coverage   lift      count
## [1]  {1033,1096} => {1086}  0.003845661 0.3125000  0.01230611 12.437819 30
## [2]  {1033,1096} => {1046}  0.003461095 0.2812500  0.01230611 11.795867 27
## [3]  {1033,1096} => {1020}  0.003332906 0.2708333  0.01230611 12.212548 26
## [4]  {1033,1096} => {1095}  0.002948340 0.2395833  0.01230611 10.803408 23
## [5]  {1033,1096} => {1079}  0.002948340 0.2395833  0.01230611  8.269865 23
## [6]  {1096}      => {1079}  0.010255096 0.2234637  0.04589155  7.713452 80
## [7]  {1033,1096} => {13433} 0.002691963 0.2187500  0.01230611  6.538194 21
## [8]  {1033,1096} => {13426} 0.002563774 0.2083333  0.01230611 11.862835 20
## [9]  {1096}      => {1086}  0.009101397 0.1983240  0.04589155  7.893498 71
## [10] {1096}      => {1046}  0.008460454 0.1843575  0.04589155  7.732114 66
```

Vamos a ver una representación gráfica a modo de grafo de las nuevas reglas de asociación encontradas:

```
## Available control parameters (with default values):
## layout    =  stress
## circular  =  FALSE
## ggraphdots    =  NULL
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
## verbose   =  FALSE
```



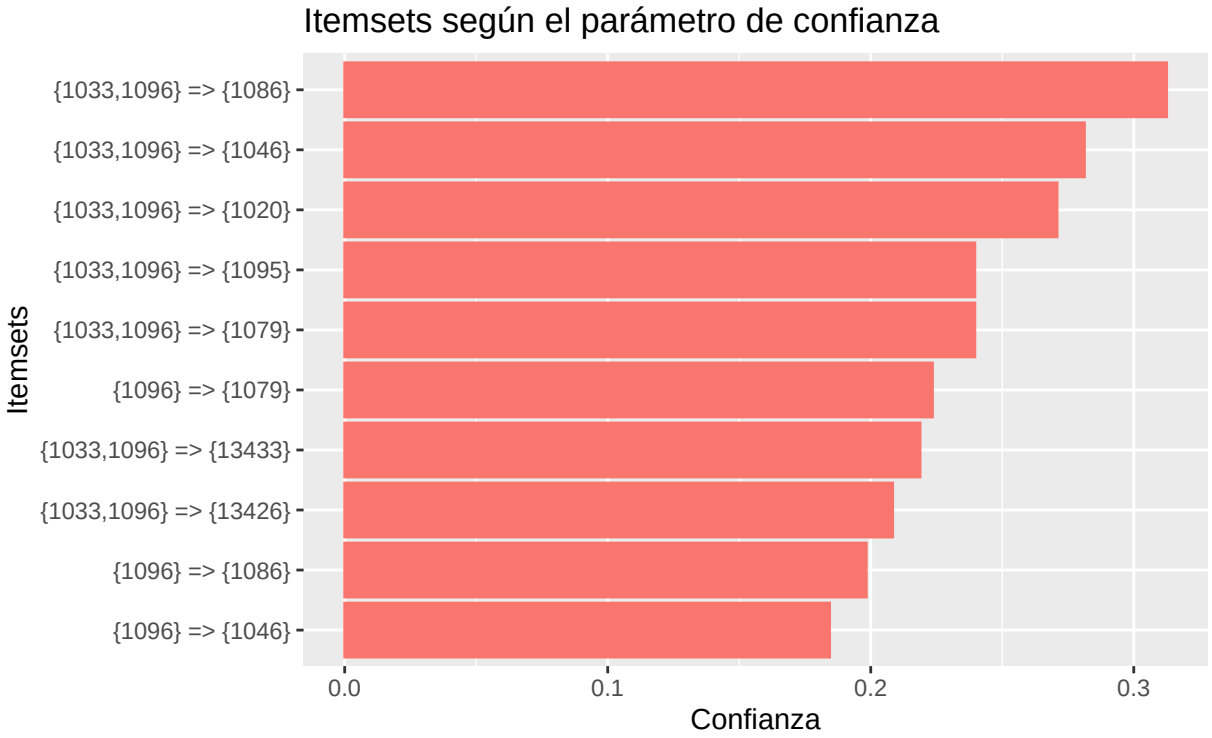
Finalmente, se han obtenido 10 reglas, y como vemos en el grafo, la mayoría de ellas

consisten en dos productos en el antecedente, los productos 1033 y 1096, por tanto, como habíamos observado en el estudio de itemsets frecuentes, la venta conjunta de estos dos productos se hace de manera frecuente.

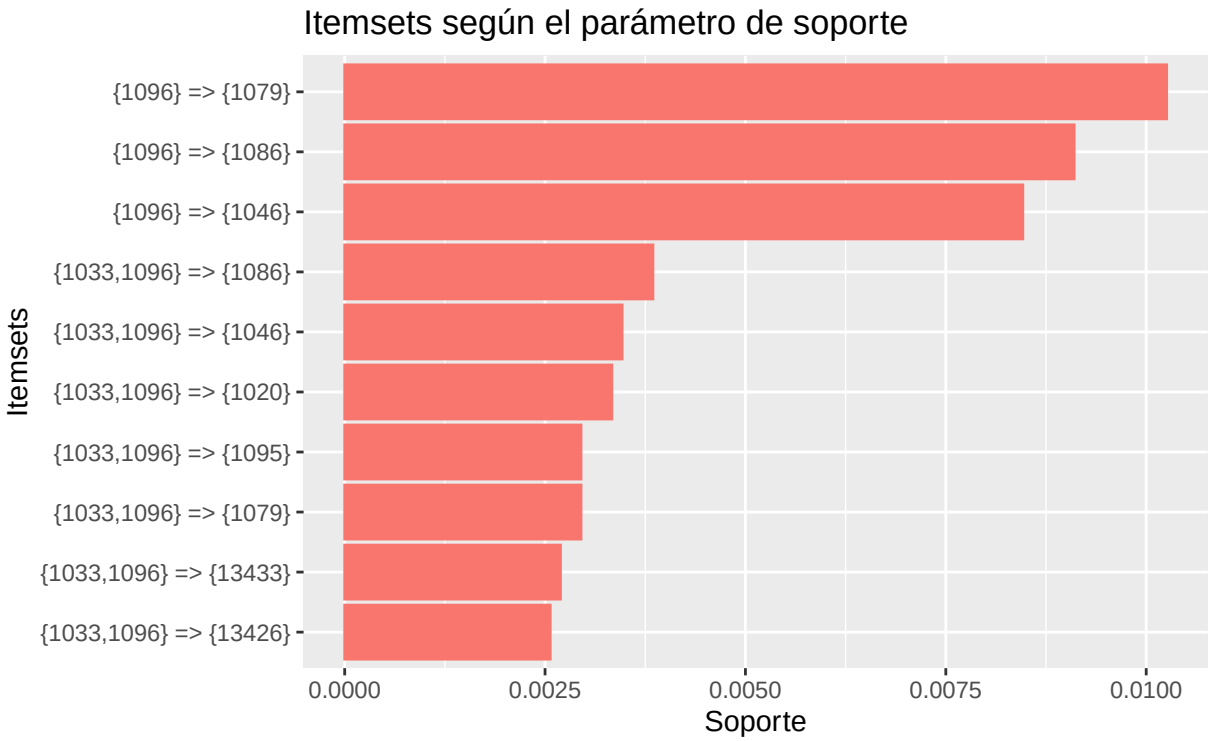
Podemos observar con un valor más alto de soporte, la regla que consiste en la venta de estos dos productos que lleva a la venta del producto 1079. Además, las reglas en las que el consecuente son los productos 1086, 1020 y 1046, tienen unos valores de lift bastante altos, indicando así que se trata de reglas robustas.

Vamos a ordenar las reglas en función de los distintos parámetros:

- Ordenación según confianza:



- Ordenación según soporte (frecuencia con que los objetos son comprados juntos):



Con esta segunda ordenación observamos los siguiente: los artículos que se han vendido juntos con mayor frecuencia son el 1096 y el 1079, con una frecuencia del 1.0 % del total de transacciones. También destacar que la compra de los productos 1096 y 1086 en la misma transacción ha tenido lugar en un 0.91 % de las transacciones. Además, la venta de los productos 1096 y 1046 se ha producido con una frecuencia del 0.84 %. Para el resto de itemsets no tienen una frecuencia ni del 0.35 %.

Los valores del soporte son tan bajos debido al gran número de transacciones, por lo tanto, para que una transacción tenga un valor de soporte del 1 % se ha tenido que producir un total de 79 veces.

0.1.4.3. Evaluación de las reglas

Veamos un resumen de las reglas encontradas, con las siguientes métricas:

- Support: 0.0019228
- Confidence: 0.18

```
## set of 10 rules
##
## rule length distribution (lhs + rhs):sizes
## 2 3
## 3 7
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.25    3.00    2.70    3.00    3.00
##
## summary of quality measures:
##     support           confidence        coverage            lift
##  Min.   :0.002564   Min.   :0.1844   Min.   :0.01231   Min.   : 6.538
```

```
##  1st Qu.:0.002948   1st Qu.:0.2109   1st Qu.:0.01231   1st Qu.: 7.772
##  Median :0.003397   Median :0.2315   Median :0.01231   Median : 9.537
##  Mean   :0.004961   Mean   :0.2377   Mean   :0.02238   Mean   : 9.726
##  3rd Qu.:0.007307   3rd Qu.:0.2630   3rd Qu.:0.03750   3rd Qu.:11.846
##  Max.   :0.010255   Max.   :0.3125   Max.   :0.04589   Max.   :12.438
##      count
##  Min.   :20.0
##  1st Qu.:23.0
##  Median :26.5
##  Mean   :38.7
##  3rd Qu.:57.0
##  Max.   :80.0
##
## mining info:
##         data ntransactions    support confidence
##  TransBasket          7801 0.00192283       0.18
```

Si los valores de support y confidence están próximos a los ajustados, revisar.

Métricas:

REPASAR ESTO

- Soporte:
  - Valor medio: 0.005335
  - Valor mínimo: 0.002179
  - Valor máximo: 0.010255
- Confianza:
  - Valor medio: 0.2209
  - Valor máximo: 0.3125
  - Valor mínimo: 0.1558

El valor de lift mide la importancia y robustez de una regla:

- Valor medio: 8.970
- Valor máximo: 12.438
- Valor mínimo: 4.657

Hemos obtenido unos valores de lift bastante altos, lo que indica que nuestras reglas son improtantes y robustas. Como ya estudiamos en la descripción teórica, este parámetro indica la fuerza de la asociación entre los productos de la parte de la izquierda (antecedentes) y los de la derecha. Cuanto mayor sea el valor de lift, mayor evidencia tendremos de que la regla no se deba a la aleatoriedad, sino que se trata de un patrón de comportamiento existente.

```
##     lhs             rhs    support     confidence coverage   lift     count
## [1] {1033,1096} => {1086} 0.003845661 0.3125     0.01230611 12.43782 30
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{1033,1096}	=>	{1086}	0.0038457	0.3125	0.0123061	12.43782	30

La regla más robusta que hemos encontrado es que al comprar los productos 1033 y 1096, se comprará también el 1086. Esta transacción ha ocurrido un total de 30 veces (frecuencia del 0.38 %). Su valor de lift es de 12.43, que es un valor bastante alto, indicando así que el producto 1086 (consecuente) está bastante vinculado a la compra conjunta de los productos 1033 y 1096 (antecedentes).

La transacción que más veces se ha repetido ha sido:

```
##     lhs       rhs     support   confidence coverage   lift     count
## [1] {1096} => {1079} 0.0102551 0.2234637  0.04589155 7.713452 80
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{1096}	=>	{1079}	0.0102551	0.2234637	0.0458916	7.713452	80

Comprar el producto 1079 al comprar el 1096, en un total de 80 ocasiones y con un valor de lift de 7.7, confirmando así la robustez de esta regla.

Otra métrica interesante para cuantificar la calidad de las reglas de asociación obtenidas es el *Test exacto de Fisher*, que permite contrastar las siguientes hipótesis:

$$\begin{cases} H_0: & \text{La regla obtenida se debe al azar, es un resultado aleatorio} \\ H_1: & \text{La regla obtenida se debe a un patrón de comportamiento real} \end{cases}$$

```
## # A tibble: 6 x 7
##   rules                 support confidence coverage  lift count Testfisher
##   <fct>                   <dbl>      <dbl>    <dbl> <dbl> <int>      <dbl>
## 1 {1033,1096} => {1086} 0.00385      0.312   0.0123 12.4     30   1.77e-25
## 2 {1033,1096} => {1046} 0.00346      0.281   0.0123 11.8     27   2.98e-22
## 3 {1033,1096} => {1020} 0.00333      0.271   0.0123 12.2     26   8.02e-22
## 4 {1033,1096} => {1095} 0.00295      0.240   0.0123 10.8     23   4.42e-18
## 5 {1033,1096} => {1079} 0.00295      0.240   0.0123  8.27    23   1.84e-15
## 6 {1096} => {1079}      0.0103       0.223   0.0459  7.71    80   1.49e-51
```

Conclusión: Los *p-valores* son todos menores que $\alpha = 0.05$, por lo que no existen evidencias significativas a favor de que las reglas obtenidas son fruto del azar, por lo que podemos afirmar que significativamente, las reglas de asociación obtenidas se deben a un comportamiento real de ventas.

0.1.4.4. Reglas maximales

Un itemset es *maximal* si no existe otro itemset que sea su superset. Se dice *regla maximal* a aquella que es generada con un itemset maximal. Vamos a estudiar la presencia de este tipo de reglas en las obtenidas haciendo uso de la función *is.maximal()*

```
## set of 7 rules
```

Existen 7 reglas maximales:

```
##     lhs             rhs     support     confidence coverage   lift      count
## [1] {1033,1096} => {13426} 0.002563774 0.2083333  0.01230611 11.862835 20
```

	lhs		rhs
[1]	{1033,1096}	=>	{13426}
[2]	{1033,1096}	=>	{1020}
[3]	{1033,1096}	=>	{1046}
[4]	{1033,1096}	=>	{1095}
[5]	{1033,1096}	=>	{1086}
[6]	{1033,1096}	=>	{1079}
[7]	{1033,1096}	=>	{13433}

```
## [2] {1033,1096} => {1020}  0.003332906 0.2708333  0.01230611 12.212548 26
## [3] {1033,1096} => {1046}  0.003461095 0.2812500  0.01230611 11.795867 27
## [4] {1033,1096} => {1095}  0.002948340 0.2395833  0.01230611 10.803408 23
## [5] {1033,1096} => {1086}  0.003845661 0.3125000  0.01230611 12.437819 30
## [6] {1033,1096} => {1079}  0.002948340 0.2395833  0.01230611  8.269865 23
## [7] {1033,1096} => {13433} 0.002691963 0.2187500  0.01230611  6.538194 21
##     Testfisher
## [1] 1.336665e-16
## [2] 8.017800e-22
## [3] 2.978559e-22
## [4] 4.418948e-18
## [5] 1.767117e-25
## [6] 1.837553e-15
## [7] 3.844979e-12
```

0.1.4.5. Reglas redundantes

Se dice que dos reglas son *redundantes* si tienen el mismo antecedente y mismo consecuente. Se trata de reglas que son subconjuntos de otras reglas más grandes.

Por ejemplo, dadas dos reglas de asociación:

$$1. \quad \{A, B\} \rightarrow \{C\} 2. \quad \{A\} \rightarrow \{C\}$$

Se tiene que la regla *1* es redundante.

Vamos a estudiar la presencia de este tipo de reglas en las obtenidas haciendo uso de la función *is.redundant()*.

```
## set of 0 rules
```

No existen reglas redundantes.

0.1.5. Conclusiones