Minería de datos: PEC2 - Métodos no supervisados

Autor: Gabriel Patricio Bonilla Sanchez

Noviembre 2020

Contents

Introducción

Presentación

Esta Prueba de Evaluación Continuada cubre principalmente los módulos 5 y 6 (Métodos de agregación y Algoritmos de asociación) del programa de la asignatura.

Competencias

Las competencias que se trabajan en esta prueba son:

- Uso y aplicación de las TIC en el ámbito académico y profesional.
- Capacidad para innovar y generar nuevas ideas.
- Capacidad para evaluar soluciones tecnológicas y elaborar propuestas de proyectos teniendo en cuenta los recursos, las alternativas disponibles y las condiciones de mercado.
- Conocer las tecnologías de comunicaciones actuales y emergentes así como saberlas aplicar convenientemente para diseñar y desarrollar soluciones basadas en sistemas y tecnologías de la información.
- Aplicación de las técnicas específicas de ingeniería del software en las diferentes etapas del ciclo de vida de un proyecto.
- Capacidad para aplicar las técnicas específicas de tratamiento, almacenamiento y administración de datos.
- Capacidad para proponer y evaluar diferentes alternativas tecnológicas para resolver un problema concreto.

Objetivos

En esta PEC trabajaremos la generación, interpretación y evaluación de un modelo de agregación y de un modelo donde generaremos reglas de asociación con el software de practicas. No perderemos de vista las fases de preparación de los datos, calidad del modelo y extracción inicial del conocimiento.

Descripción de la PEC a realizar

Recursos Básicos

Material docente proporcionado por la UOC.

Módulo 5 y 6 del material didáctico.

Criterios de valoración

Ejercicios teóricos

Todos los ejercicios deben ser presentados de forma razonada y clara, especificando todos y cada uno de los pasos que se hayan llevado a cabo para su resolución. No se aceptará ninguna respuesta que no esté claramente justificada.

Ejercicios prácticos

Para todas las PEC es necesario documentar en cada apartado del ejercicio práctico que se ha hecho y cómo se ha hecho.

Formato y fecha de entrega

El formato de entrega es: usernameestudiant-PECn.html/doc/docx/odt/pdf
Fecha de Entrega: 18/11/2020
Se debe entregar la PEC en el buzón de entregas del aula

Nota: Propiedad intelectual

> A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por lo tanto comprensible hacerlo en el marco de una práctica de los estudios de Informática, Multimedia y Telecomunicación de la UOC, siempre y cuando esto se documente claramente y no suponga plagio en la práctica.
>
> Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se debe presentar junto con ella un documento en qué se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar dónde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por copyright.
>
> Deberéis, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.

Ejemplo 1.1

Métodos de agregación con datos autogenerados

En este ejemplo vamos a generar un conjunto de muestras aleatorias para posteriormente usar el algoritmo kmeans para agruparlas. Se crearán las muestras alrededor de dos puntos concretos. Por lo tanto, lo lógico será agrupar en dos clústers. Puesto que inicialmente, en un problema real, no se conoce cual es el número más idóneo de clústers k, vamos a probar primero con dos (el valor óptimo) y posteriormente con 4 y 8 clústers. Para evaluar la calidad de cada proceso de agrupación vamos a usar la silueta media. La silueta de cada muestra evalúa como de bien o mal está clasificada la muestra en el clúster al que ha sido asignada. Para ello se usa una fórmula que tiene en cuenta la distancia a las muestras de su clúster y la distancia a las muestras del clúster vecino más cercano.

A la hora de probar el código que se muestra, es importante tener en cuenta que las muestras se generan de forma aleatoria y también que el algoritmo kmeans tiene una inicialización aleatoria. Por lo tanto, en cada ejecución se obtendrá unos resultados ligeramente diferentes.

Lo primero que hacemos es cargar la librería cluster que contiene las funciones que se necesitan

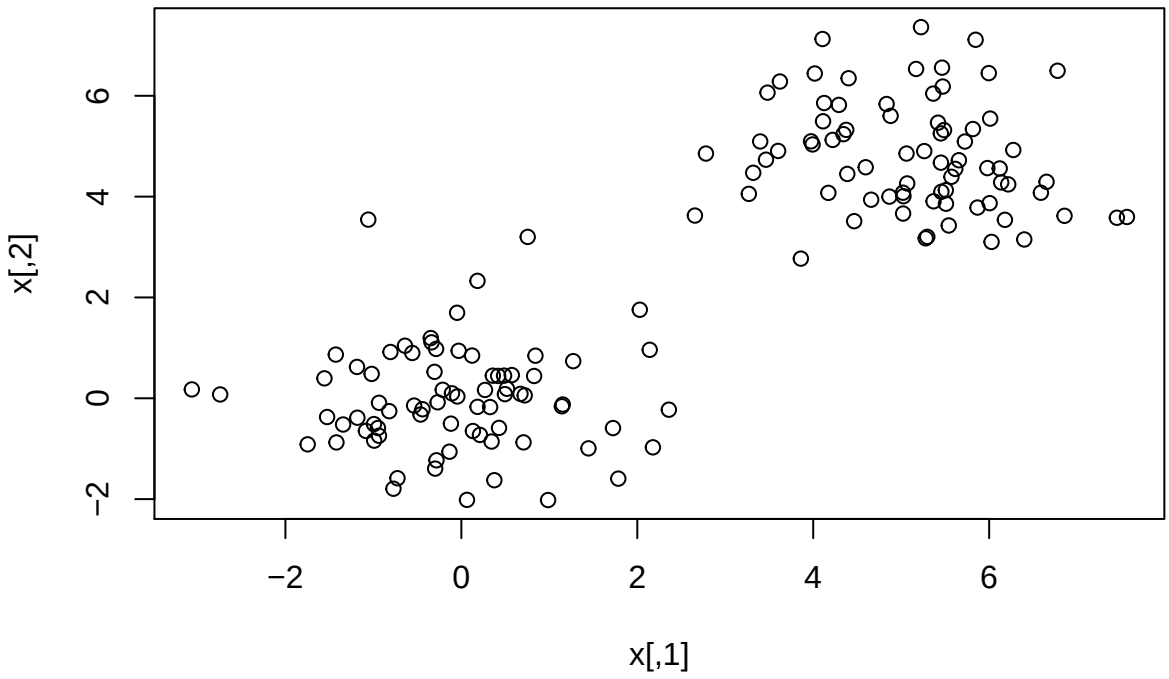
```
library(cluster)
```

Generamos las muestras de forma aleatoria tomando como centro los puntos [0,0] y [5,5].

```
n <- 150 # número de muestras
p <- 2   # dimensión

sigma <- 1 # varianza de la distribución
mean1 <- 0 # centro del primer grupo
mean2 <- 5 # centro del segundo grupo

n1 <- round(n/2) # número de muestras del primer grupo
n2 <- round(n/2) # número de muestras del segundo grupo

x1 <- matrix(rnorm(n1*p,mean=mean1,sd=sigma),n1,p)
x2 <- matrix(rnorm(n2*p,mean=mean2,sd=sigma),n2,p)
```

Juntamos todas las muestras generadas y las mostramos en una gráfica

```
x  <- rbind(x1,x2)
plot (x)
```



Como se puede comprobar las muestras están claramente separadas en dos grupos. Si se quiere complicar el problema se puede modificar los puntos centrales (mean1 y mean2) haciendo que estén más próximos y/o ampliar la varianza (sigma) para que las muestras estén más dispersas.

A continuación vamos a aplicar el algoritmo kmeans con 2, 4 y 8 clústers

```
fit2       <- kmeans(x, 2)
y_cluster2 <- fit2$cluster
```

```
fit4       <- kmeans(x, 4)
y_cluster4 <- fit4$cluster

fit8       <- kmeans(x, 8)
y_cluster8 <- fit8$cluster
```

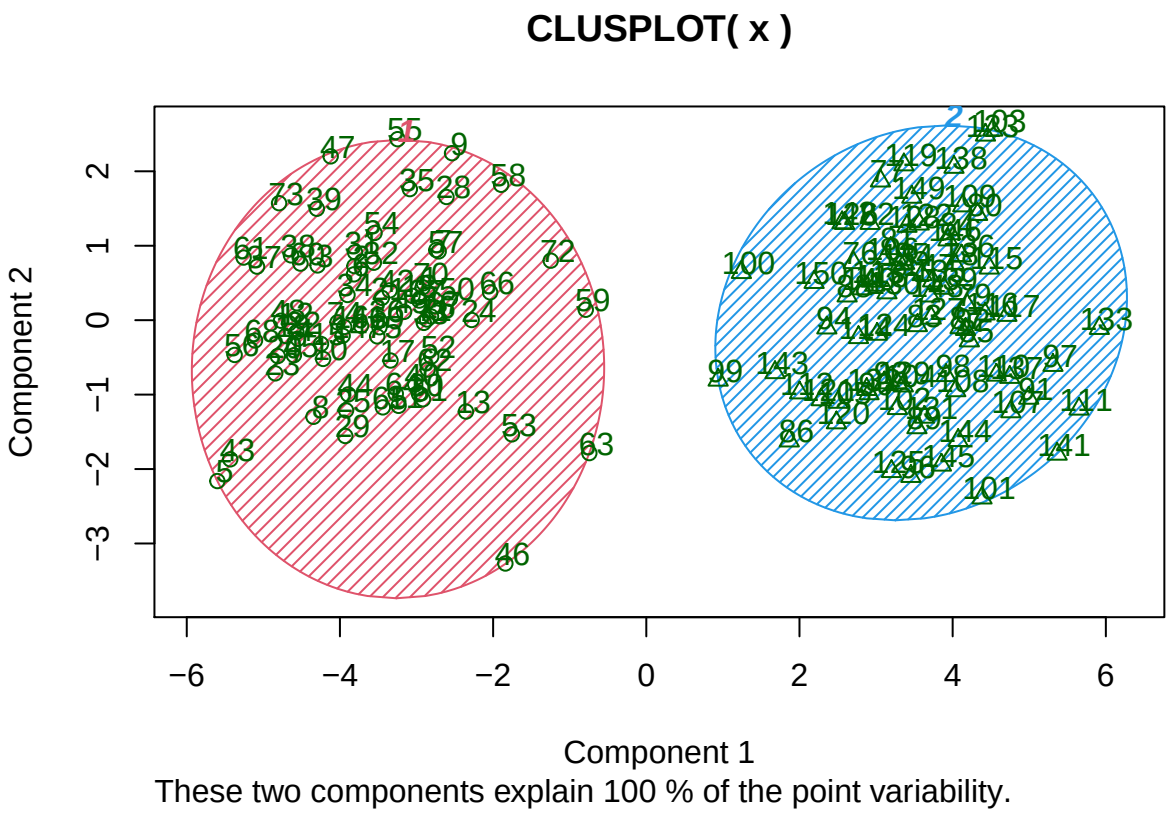
Las variables y_cluster2, y_cluster4 e y_cluster8 contienen para cada muestra el identificador del clúster a las que han sido asignadas. Por ejemplo, en el caso de los k=2 las muestras se han asignado al clúster 1 o al 2

```
y_cluster2
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2
```

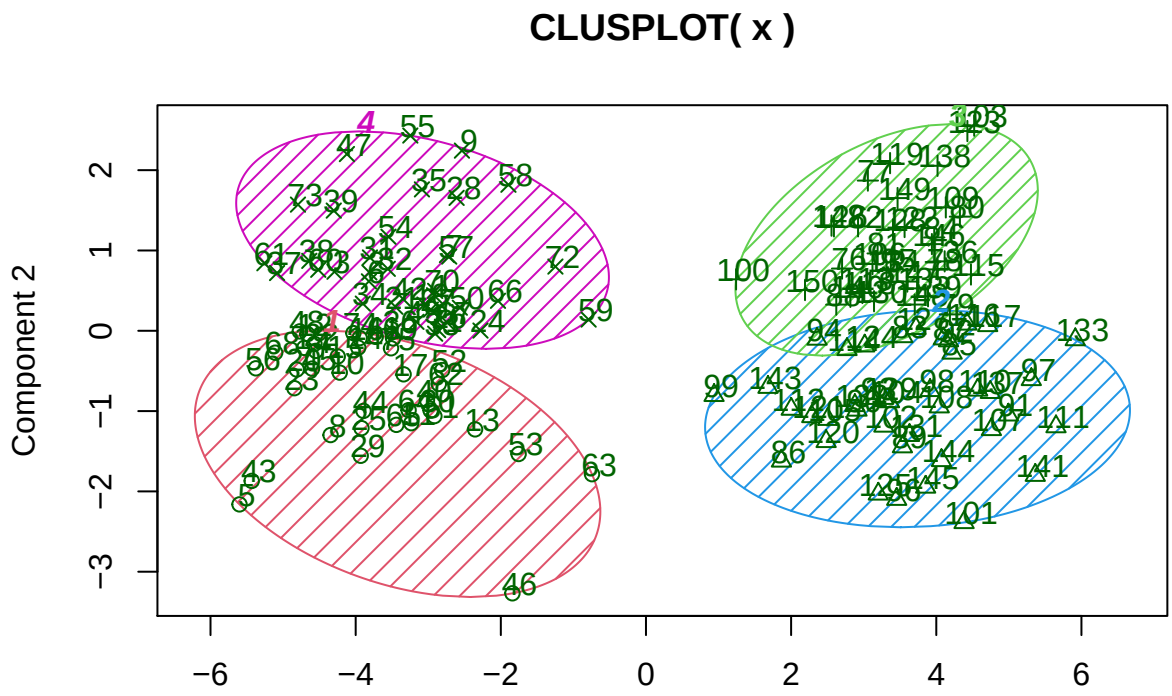
Para visualizar los clústers podemos usar la función clusplot. Vemos la agrupación con 2 clústers

```
clusplot(x, fit2$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



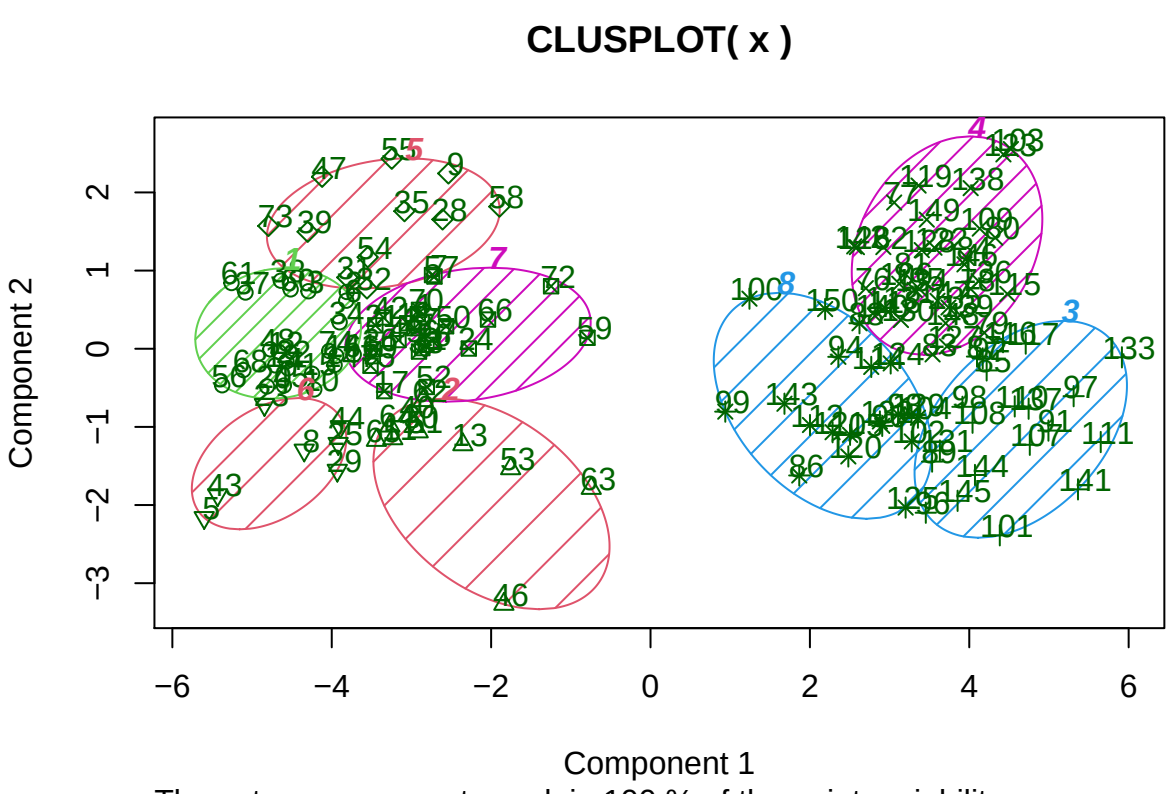
con 4

```
clusplot(x, fit4$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



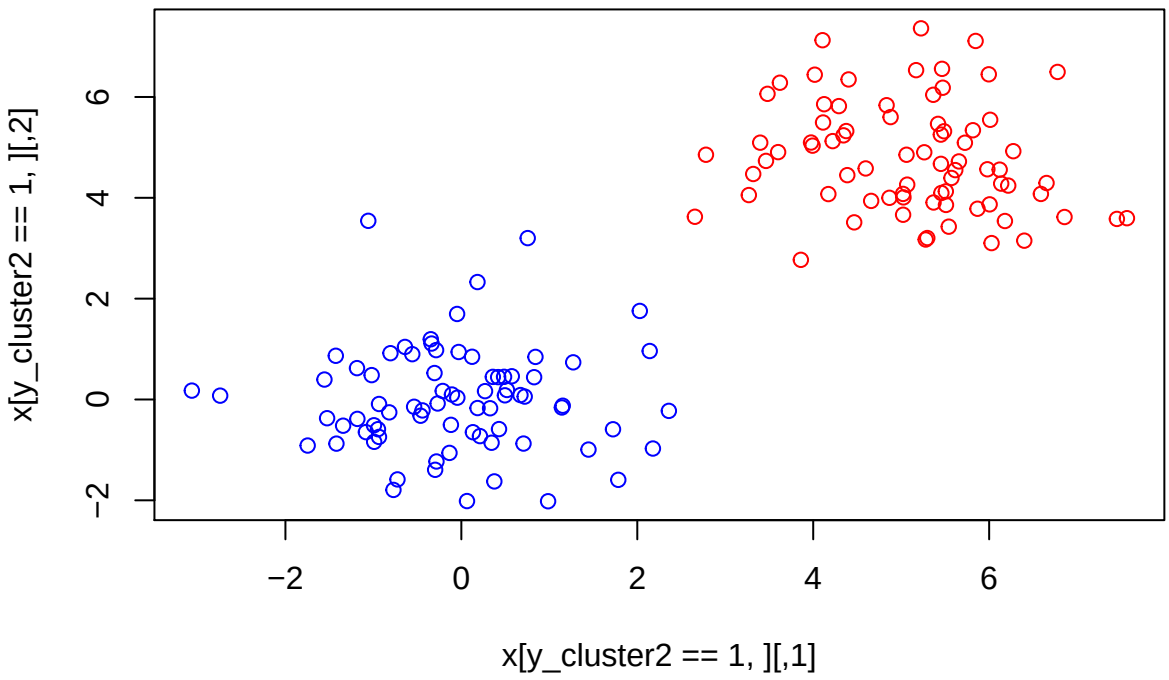
y con 8

```
clusplot(x, fit8$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



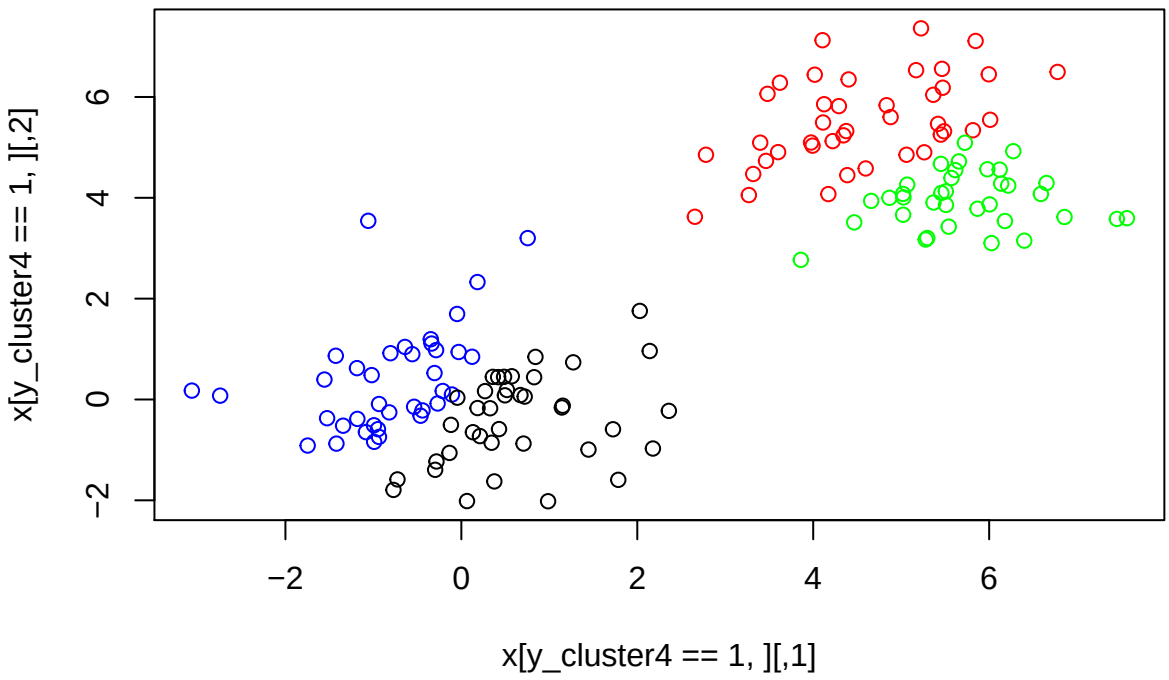
También podemos visualizar el resultado del proceso de agrupamiento con el siguiente código para el caso de 2 clústers

```
plot(x[y_cluster2==1,],col='blue', xlim=c(min(x[,1]), max(x[,1])), ylim=c(min(x[,2]), max(x[,2])))
points(x[y_cluster2==2,],col='red')
```



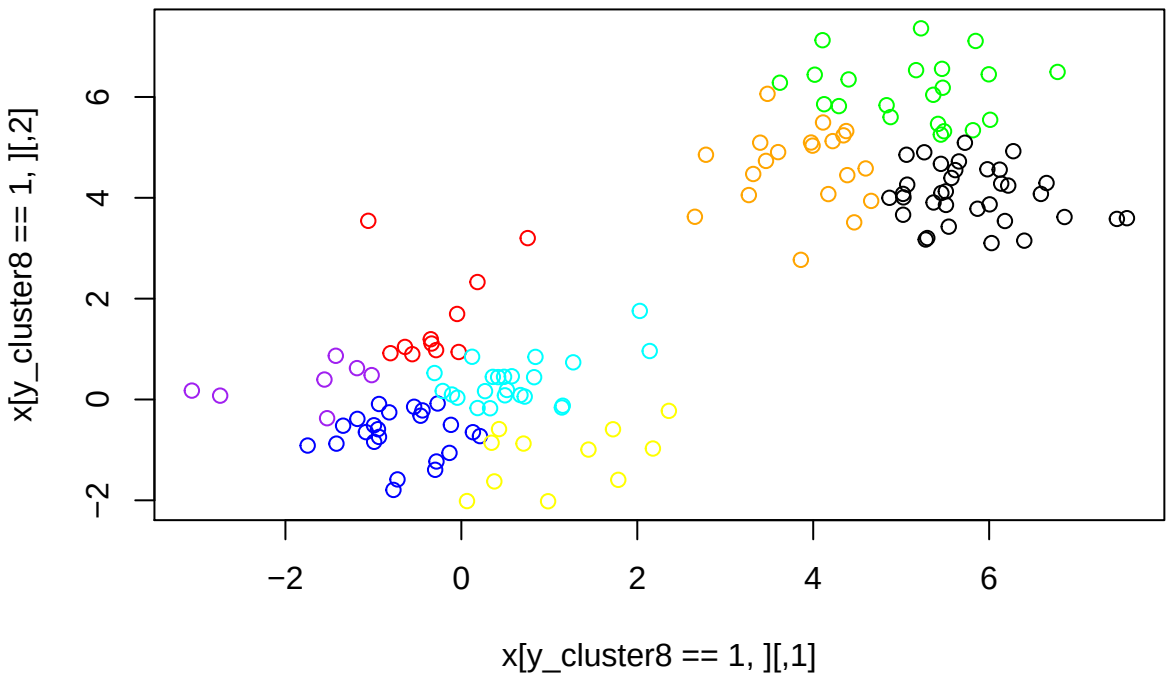
para 4

```
plot(x[y_cluster4==1,],col='blue', xlim=c(min(x[,1]), max(x[,1])), ylim=c(min(x[,2]), max(x[,2])))
points(x[y_cluster4==2,],col='red')
points(x[y_cluster4==3,],col='green')
points(x[y_cluster4==4,],col='black')
```



y para 8

```
plot(x[y_cluster8==1,],col='blue', xlim=c(min(x[,1]), max(x[,1])), ylim=c(min(x[,2]), max(x[,2])))
points(x[y_cluster8==2,],col='red')
points(x[y_cluster8==3,],col='green')
points(x[y_cluster8==4,],col='black')
points(x[y_cluster8==5,],col='yellow')
points(x[y_cluster8==6,],col='purple')
points(x[y_cluster8==7,],col='cyan')
points(x[y_cluster8==8,],col='orange')
```



Ahora vamos a evaluar la calidad del proceso de agregación. Para ello usaremos la función silhouette que calcula la silueta de cada muestra

```
d  <- daisy(x)
sk2 <- silhouette(y_cluster2, d)
sk4 <- silhouette(y_cluster4, d)
sk8 <- silhouette(y_cluster8, d)
```

La función silhouette devuelve para cada muestra, el clúster dónde ha sido asignado, el clúster vecino y el valor de la silueta. Por lo tanto, calculando la media de la tercera columna podemos obtener una estimación de la calidad del agrupamiento

```
mean(sk2[,3])
```

```
## [1] 0.7337687
```

```
mean(sk4[,3])
```

```
## [1] 0.3359072
```

```
mean(sk8[,3])
```

```
## [1] 0.3544198
```

Como se puede comprobar, agrupar con dos clúster es mejor que en 4 o en 8, lo cual es lógico teniendo en cuenta como se han generado los datos.

Ejemplo 1.2

Métodos de agregación con datos reales

A continuación vamos a ver otro ejemplo de cómo se usan los modelos de agregación. Para ello usaremos el fichero iris.csv. Esta base de datos se encuentra descrita en https://archive.ics.uci.edu/ml/datasets/iris. Este dataset está previamente trabajado para que los datos estén limpios y sin errores. De no ser así antes de nada deberíamos buscar errores, valores nulos u outlayers. Deberíamos mirar de discretizar o eliminar columnas. Incluso realizar este último paso varias veces para comprobar los diferentes resultados y elegir el que mejor performance nos dé. De todas formas vamos a visualizar la estructura y resumen de los datos

```
iris_data<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=T
attach(iris_data)
colnames(iris_data) <- c("sepalLength", "sepalWidth", "petalLength", "petalWidth", "class")
summary(iris_data)
```

```
##   sepalLength      sepalWidth     petalLength      petalWidth
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.400   Median :1.300
##  Mean   :5.848   Mean   :3.051   Mean   :3.774   Mean   :1.205
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##     class
##  Length:149
##  Class :character
##  Mode  :character
##
##
##
```

Como se puede comprobar, esta base de datos está pensada para problemas de clasificación supervisada que pretende clasificar cada tipo de flor en uno de las tres clases existentes (Iris-setosa, Iris-versicolor o Iris-virginica). Como en este ejemplo vamos a usar un método no supervisado, transformaremos el problema supervisado original en uno no supervisado. Para conseguirlo no usaremos la columna class, que es la variable que se quiere predecir. Por lo tanto, intentaremos encontrar agrupaciones usando únicamente los cuatro atributos que caracterizan a cada flor.

Cargamos los datos y nos quedamos únicamente con las cuatro columnas que definen a cada flor

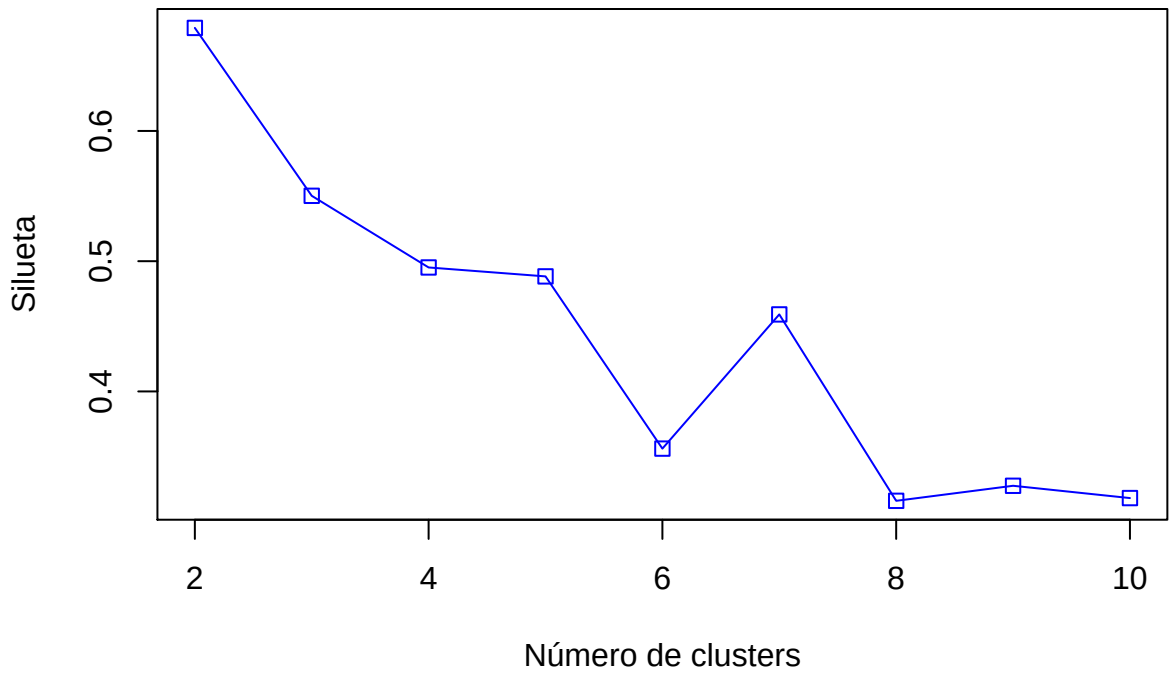
```
x <- iris_data[,1:4]
```

Como inicialmente no conocemos el número óptimo de clústers, probamos con varios valores

```
d <- daisy(x)
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(x, i)
  y_cluster     <- fit$cluster
  sk            <- silhouette(y_cluster, d)
  resultados[i] <- mean(sk[,3])
}
```

Mostramos en un gráfica los valores de las siluetas media de cada prueba para comprobar que número de clústers es el mejor

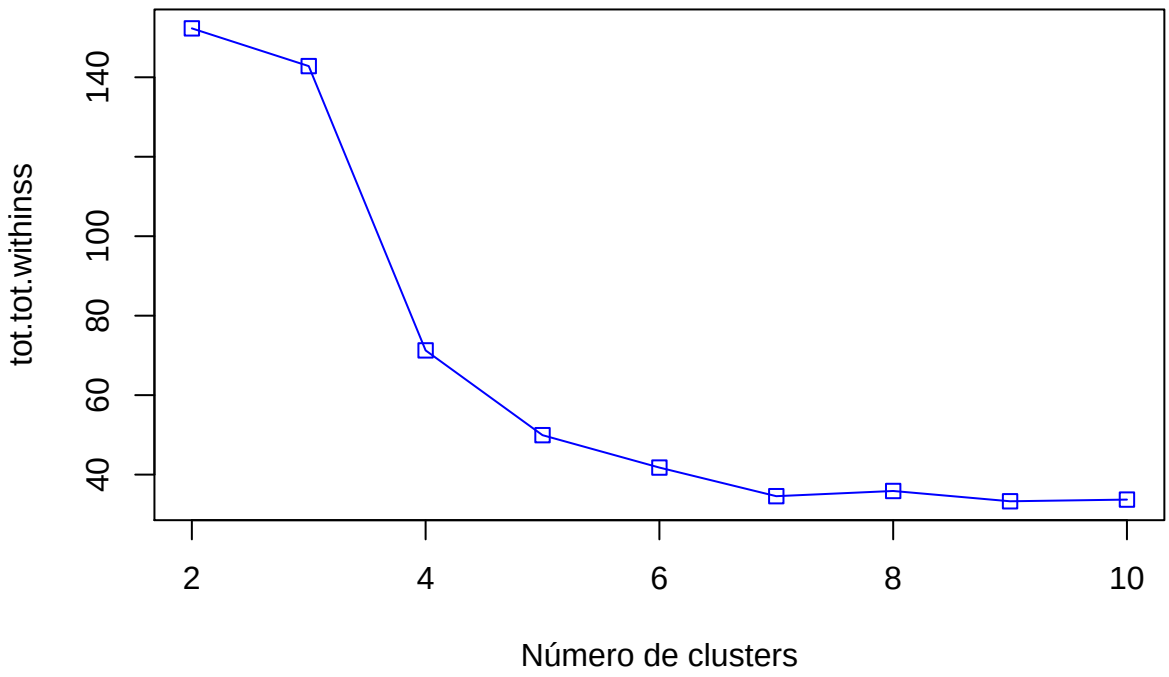
```
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="Silueta")
```



Aunque el valor esperado es k=3, dado que el conjunto original tiene 3 clases, el mejor valor que se obtiene es k=2.

Otro forma de evaluar cual es el mejor número de clústers es considerar el mejor modelo, aquel que ofrece la menor suma de los cuadrados de las distancias de los puntos de cada grupo con respecto a su centro (withinss), con la mayor separación entre centros de grupos (betweenss). Como se puede comprobar es una idea conceptualmente similar a la silueta. Una manera común de hacer la selección del número de clústers consiste en aplicar el método elbow (codo), que no es más que la selección del número de clústers en base a la inspección de la gráfica que se obtiene al iterar con el mismo conjunto de datos para distintos valores del número de clústers. Se seleccionará el valor que se encuentra en el "codo" de la curva

```
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(x, i)
  resultados[i] <- fit$tot.withinss
}
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="tot.tot.withinss")
```



En este caso el número óptimo de clústers son 4 que es cuando la curva comienza a estabilizarse.

También se puede usar la función kmeansruns del paquete fpc que ejecuta el algoritmo kmeans con un conjunto de valores, para después seleccionar el valor del número de clústers que mejor funcione de acuerdo a dos criterios: la silueta media ("asw") y Calinski-Harabasz ("ch").

```
library(fpc)
fit_ch  <- kmeansruns(x, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(x, krange = 1:10, criterion = "asw")
```

Podemos comprobar el valor con el que se ha obtenido el mejor resultado y también mostrar el resultado obtenido para todos los valores de k usando ambos criterios

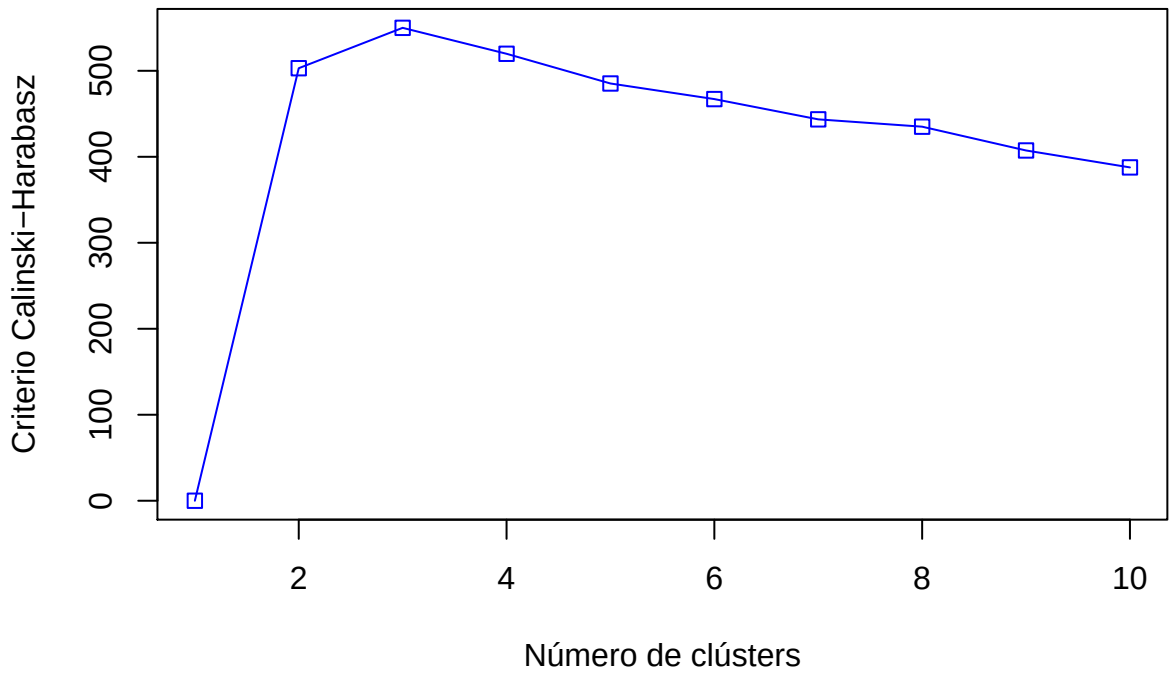
```
fit_ch$bestk
```

```
## [1] 3
```

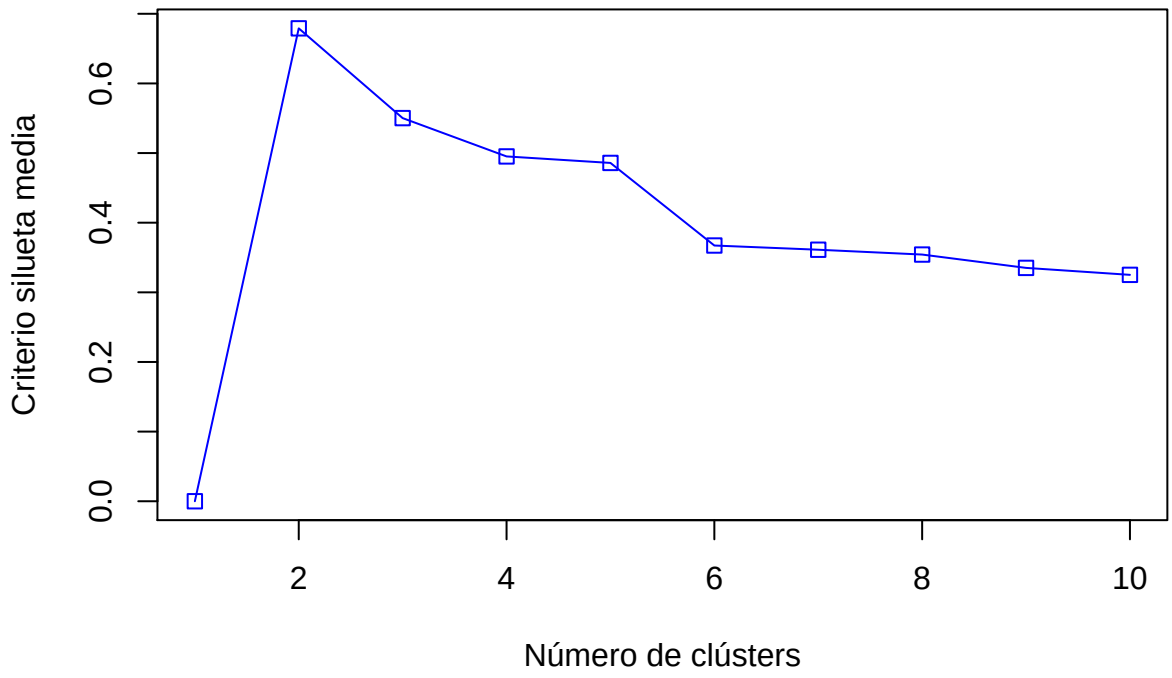
```
fit_asw$bestk
```

```
## [1] 2
```

```
plot(1:10,fit_ch$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio Calinski-Haraba
```




```
plot(1:10,fit_asw$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio silueta media
```

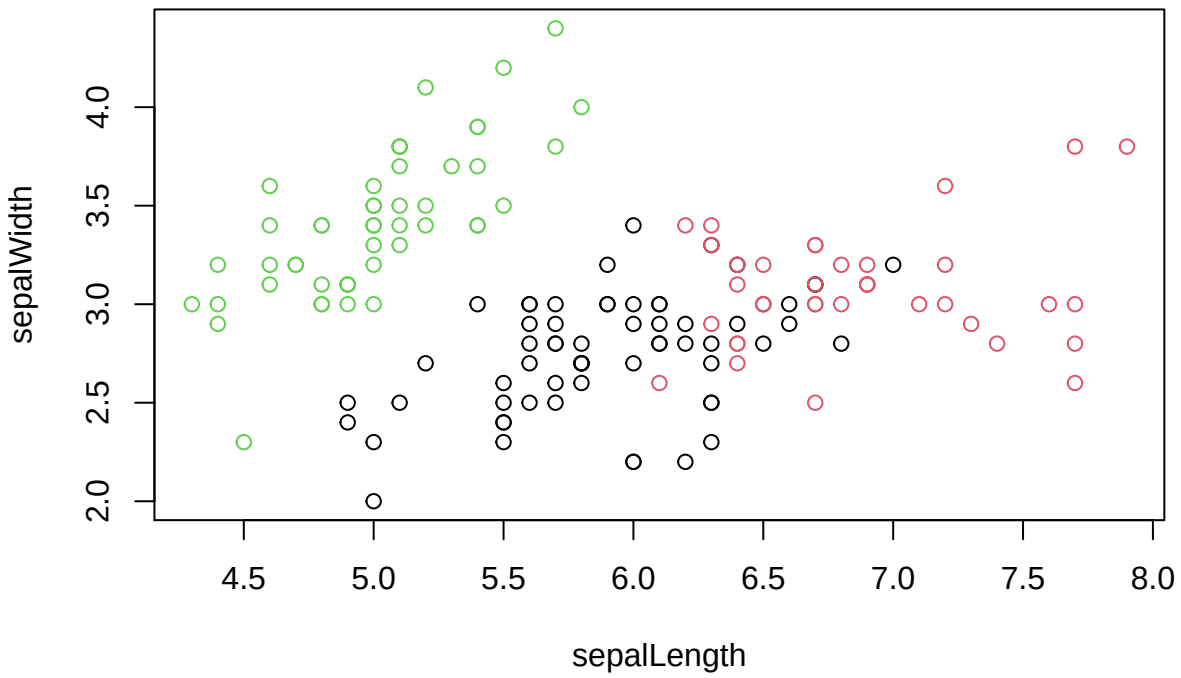


Los resultados son muy parecidos a los que hemos obtenido anteriormente. Con el criterio de la silueta media se obtienen dos clústers y con el Calinski-Harabasz se obtienen 3.

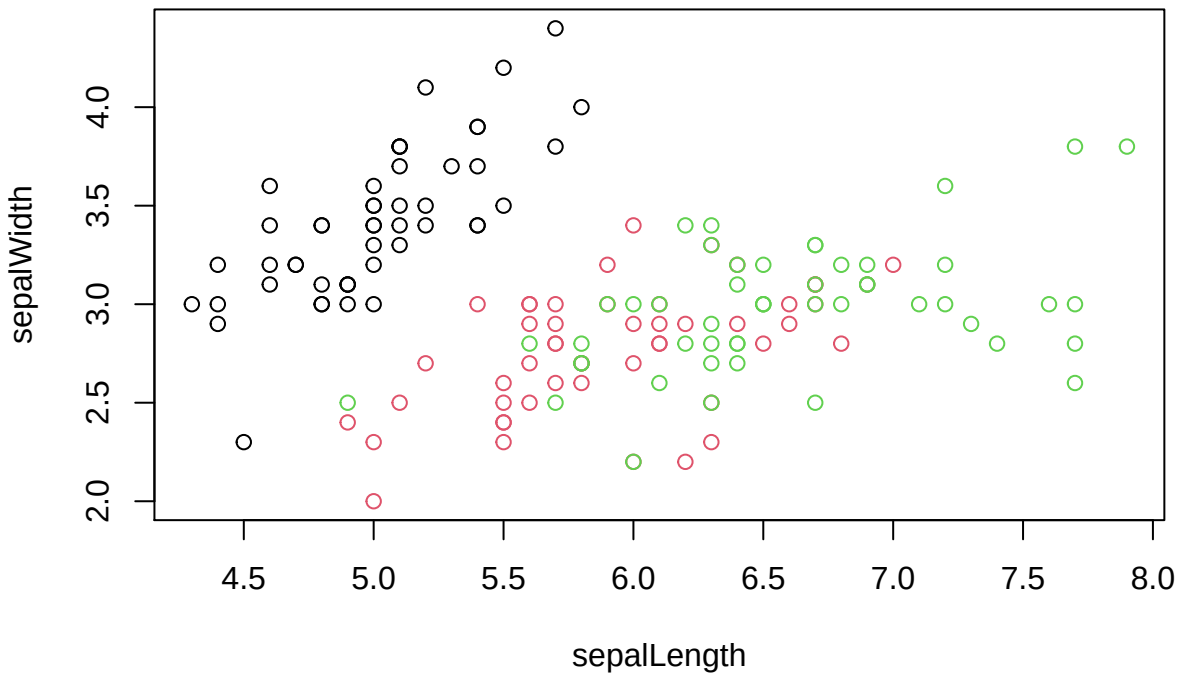
Como se ha comprobado, conocer el número óptimo de clústers no es un problema fácil. Tampoco lo es la evaluación de los modelos de agregación.

Como en el caso que estudiamos sabemos que los datos pueden ser agrupados en 3 clases, vamos a ver como se ha comportado el kmeans en el caso de pedirle 3 clústers. Para eso comparamos visualmente los campos dos a dos, con el valor real que sabemos está almacenado en el campo "class" del dataset original.

```
iris3clusters <- kmeans(x, 3)

# sepalLength y sepalWidth
plot(x[c(1,2)], col=iris3clusters$cluster)
```

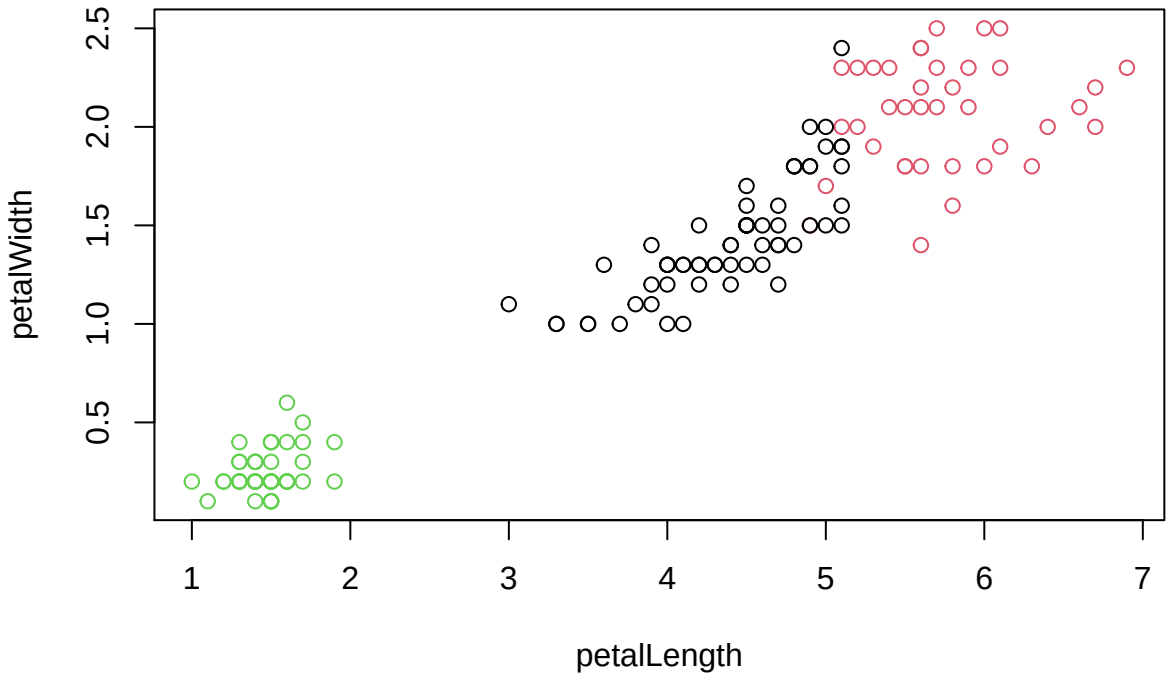



```
plot(x[c(1,2)], col=as.factor(iris_data$class))
```

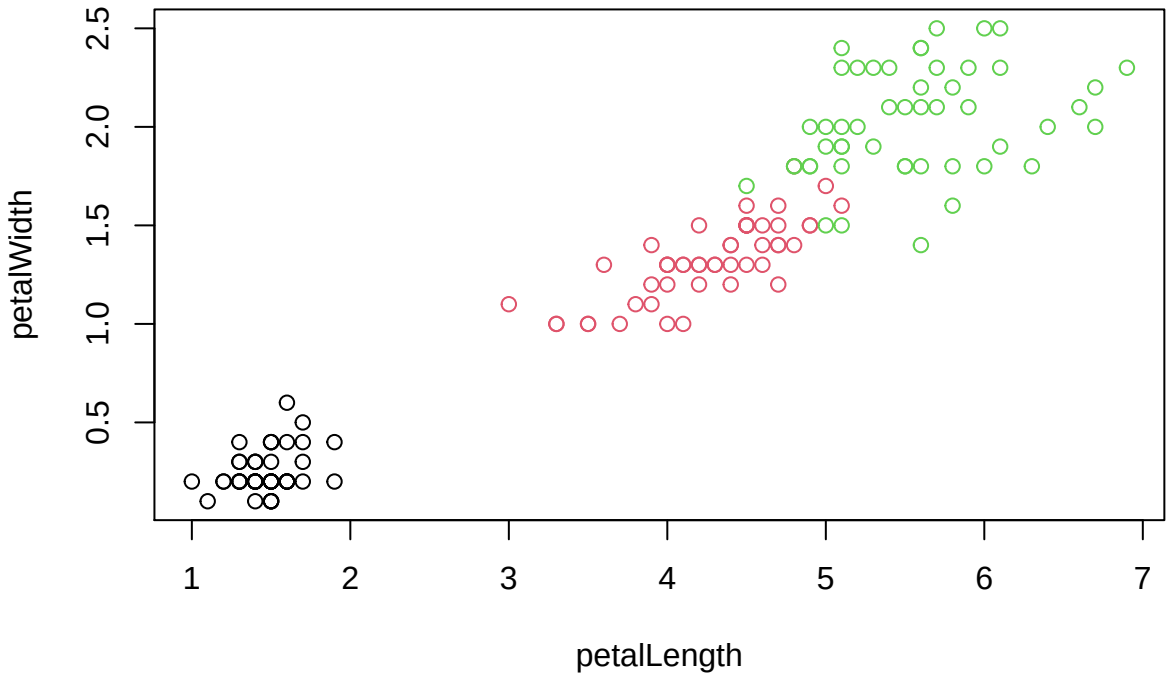


Podemos observar que el sépalo no es un buen indicador para diferenciar a las tres subespecies, no con la metodología de kmeans, dado que dos de las subespecies están demasiado mezcladas para poder diferenciar nada.

```
# petalLength y petalWidth
plot(x[c(3,4)], col=iris3clusters$cluster)
```




```
plot(x[c(3,4)], col=as.factor(iris_data$class))
```



El tamaño del pétalo sin embargo, parece hacer un mucho mejor trabajo para dividir las tres clases de flores. El grupo formado por los puntos negros que ha encontrado el algoritmo coincide con los de la flor Iris Setosa. Los otros dos grupos sin embargo se entremezclan algo más, y hay ciertos puntos que se clasifican como Versicolor cuando en realidad son Virginica.

Una buena técnica que ayuda a entender los grupos que se han formado, es mirar de darles un nombre. Cómo por ejemplo:

- Grupo 1: Sólo setosas
- Grupo 2: Principalmente versicolor
- Grupo 3: Virgínicas o iris pétalo grande

Esto nos ayuda a entender cómo están formados los grupos y a referirnos a ellos en análisis posteriores.

Una última cosa que nos queda por hacer, es saber cuales de las muestras iniciales han sido mal clasificadas y cómo. Eso lo conseguimos con el siguiente comando.

```
table(iris3clusters$cluster,iris_data$class)
```

```
##
##     Iris-setosa Iris-versicolor Iris-virginica
##   1           0              48             14
##   2           0               2             36
##   3          49               0              0
```

Y así, podemos sacar un porcentaje de precisión del modelo

```
100*(36 + 48 + 49)/(133+(2+14))
```

```
## [1] 89.26174
```

Ejercicio 1.1

Tomando como punto de partida los ejemplos mostrados, realizar un estudio similar con otro conjunto de datos. Pueden ser datos reales de vuestro ámbito laboral o de algún repositorio de datos de Internet. Mirad por ejemplo: http://www.ics.uci.edu/~mlearn/MLSummary.html.

A la hora de elegir la base de datos ten en cuenta que sea apropiada para problemas no supervisados y que los atributos sean también apropiados para su uso con el algoritmo kmeans.

No hay que olvidarse de la fase de preparación y análisis de datos.

Respuesta 1.1:

15% Se explican los campos de la base de datos, preparación y análisis de datos

Para la respuesta de este ejercicio usaremos el siguienre dataset ubicado en http://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+. Contiene datos sobre estimaciones de niveles de obesidad en individuos de paises como Colombia, Perú y México, basado en sus hábitos alimenticios y condicion física.

Este dataset está contiene 17 atributos/variables y 2111 instancias, según el siguiente detalle:

gender.- Cadena de texto que indica el genero del individuo: Male (Masculino) o Female (Femenino).

age.- Valor numérico que indica la edad del individuo. Entre 14 y 61.

height.- Valor numérico que indica la altura (en metros) del individuo.

weight.- Valor numérico que indica el peso (en kilogramos) del individuo.

family_history_with_overweight.- Valor booleano que indica si algún familiar ha sufrido o padece de sobrepeso (SI/NO).

FAVC.- Valor booleano que indica si el individuo consume alimentos ricos en calorías (SI/NO).

FCVC.- Valor numérico que indica la frecuencia en el consumo de vegetables por el individuo. Entre el 1 (Nunca) al 3 (Siempre).

NCP.- Número de comidas principales en el día. Entre 1 a 4.

CAEC.- Valor categórico que indica el consumo de alimentos entre comidas principales: No, A veces, Frecuentemente y Siempre.

SMOKE.- Valor booleano que indica si el individuo fuma (SI / NO).

CH2O.- Valor numérico que indica los litros de agua que el individuo consume. Entre 1 a 3.

SCC.- Valor booleano que indica si el individuo monitorea o no su consumo de calorías.

FAF.- Valor numérico que indica la frecuencia de actividad física. Entre el 0 a 3.

TUE.- Valor numérico que indica las horas que el individuo usa dispositivos de tecnología.

CALC.- Valor categórico que indica la frecuencia de consumo de alcohol por el individuo: No bebo, A veces, Frecuentemente, Siempre.

MTRANS.- Valor categórico que indica el método de transporte usado por el individuo: Automovil, Moto, Bicicleta, Transporte Público, Caminando.

NObeyesdad.- Valor de clase, donde se indica la estimación del nivel de obesidad en el individuo: Bajo peso, normal, Sobre Peso I, Sobre Peso II, Obesidad I, Obesidad II, Obesidad III. Esta etiqueta para los fines de la PEC no deberá ser tomada en cuenta para el análisis

Para el fin de la resolución de este ejercicio y el siguiente, solo vamos a usar los atributos: **Age (Edad)**, **Height (Altura)** y **Weight (Peso)**, los cuales corresponden a la edad expresada en años, la altura de la persona en metros y el peso en kilogramos, respectivamente. Además añadiremos una columna más que corresponde al indice de masa corporal calculado a partir del la altura y peso.Esta columna será calculada y se considerará en la etapa de preparación de los datos.

El archivo no contiene datos que deban ser tratados, como nulos o valores desconocidos. Los atributos que corresponden a los hábitos alimenticios y de condición física son categóricos, pero para la siguiente práctica no se los va a convertir ni tomar en cuenta. Para efectos de la práctica los grupos de objetos se clasificarán según los distintos niveles de obesidad, para lo cual necesitamos únicamente la edad, el peso, la altura y el IMC (índice de masa corporal), el cual se calcula a partir de la altura y peso.

Así mismo, la columna que corresponde a la clase (NObeyesdad) se va a omitir, ya que vamos a querer obtenerla a partir del análisis que hagamos.

En el siguiente artículo [1] relacionado al dataset, encontramos la descripción de los atributos, las clases que corresponden a los niveles de obesidad y además establece los rangos de la clase NObeyesdad. Para efectos del análisis vamos a considerar el índice de masa corpora óptimo, que se encuentra en el rango [18.5 - 24.9]. Este dato es importante conocerlo, ya que los demás niveles se establecerán tomándolo como referencia.

Por cultura general sabemos que hay individuos, donde según su índice de masa corporal pueden estar:

- Con un IMC por debajo del valor normal.
- Con varios niveles más por encima del valor normal.

Para cargar la estructura de datos lo haremos a partir de una archivo CSV, el mismo que se descargó desde http://archive.ics.uci.edu/ml/machine-learning-databases/00544/ObesityDataSet_raw_and_data_sinthetic%20(2).zip y se ha descomprimido en la misma ruta del archivo .Rmd. Cabe indicar que la primera fila del archivo corresponde a los nombres de los atributos. Ahora procedemos a cargar el dataset y ver el resumen de los datos.

```
obesity_data_all<-read.csv("ObesityDataSet_raw_and_data_sinthetic.csv", sep=",")
summary(obesity_data_all)
```

```
##     Gender               Age             Height          Weight
##  Length:2111        Min.   :14.00   Min.   :1.450   Min.   : 39.00
##  Class :character   1st Qu.:19.95   1st Qu.:1.630   1st Qu.: 65.47
```

```
##  Mode  :character   Median :22.78   Median :1.700   Median : 83.00
##                     Mean   :24.31   Mean   :1.702   Mean   : 86.59
##                     3rd Qu.:26.00   3rd Qu.:1.768   3rd Qu.:107.43
##                     Max.   :61.00   Max.   :1.980   Max.   :173.00
##  family_history_with_overweight     FAVC               FCVC
##  Length:2111                    Length:2111        Min.   :1.000
##  Class :character               Class :character   1st Qu.:2.000
##  Mode  :character               Mode  :character   Median :2.386
##                                                    Mean   :2.419
##                                                    3rd Qu.:3.000
##                                                    Max.   :3.000
##       NCP            CAEC              SMOKE               CH2O
##  Min.   :1.000   Length:2111        Length:2111        Min.   :1.000
##  1st Qu.:2.659   Class :character   Class :character   1st Qu.:1.585
##  Median :3.000   Mode  :character   Mode  :character   Median :2.000
##  Mean   :2.686                                         Mean   :2.008
##  3rd Qu.:3.000                                         3rd Qu.:2.477
##  Max.   :4.000                                         Max.   :3.000
##      SCC                 FAF              TUE            CALC
##  Length:2111        Min.   :0.0000   Min.   :0.0000   Length:2111
##  Class :character   1st Qu.:0.1245   1st Qu.:0.0000   Class :character
##  Mode  :character   Median :1.0000   Median :0.6253   Mode  :character
##                     Mean   :1.0103   Mean   :0.6579
##                     3rd Qu.:1.6667   3rd Qu.:1.0000
##                     Max.   :3.0000   Max.   :2.0000
##     MTRANS           NObeyesdad
##  Length:2111        Length:2111
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Como se manifestado anteriormente, este dataset está pensado para problemas de clasificación supervisada que pretende clasificar cada individuo según su nivel de obesidad en uno de los siguientes 6 niveles (bajo peso, normal, sobrepeso, obesidad I, obesidad II, obesidad III). Para esta pregunta vamos a usar un método no supervisado (clustering), por lo cual transformaremos el problema supervisado original en uno no supervisado. Para conseguirlo no usaremos la columna **NObeyesdad**, que es la variable que se quiere predecir. Para lo cual, intentaremos encontrar agrupaciones usando únicamente los dos atributos (Altura y Peso) que son los que intervienen para definir el índice de masa corporal.

Cargamos los datos y nos quedamos únicamente con las columnas 3 (Altura) y 4 (Peso). Además calculamos una tercera columna (mass_body_idx), la cual corresponde al índice de masa corporal, que se calcula según la formula:

$$mass\_body\_idx = \frac{peso}{altura * altura}$$

```
data <- obesity_data_all[,2:4]
data["mass_body_idx"] <- ((data$Weight) / (data$Height * data$Height))
```

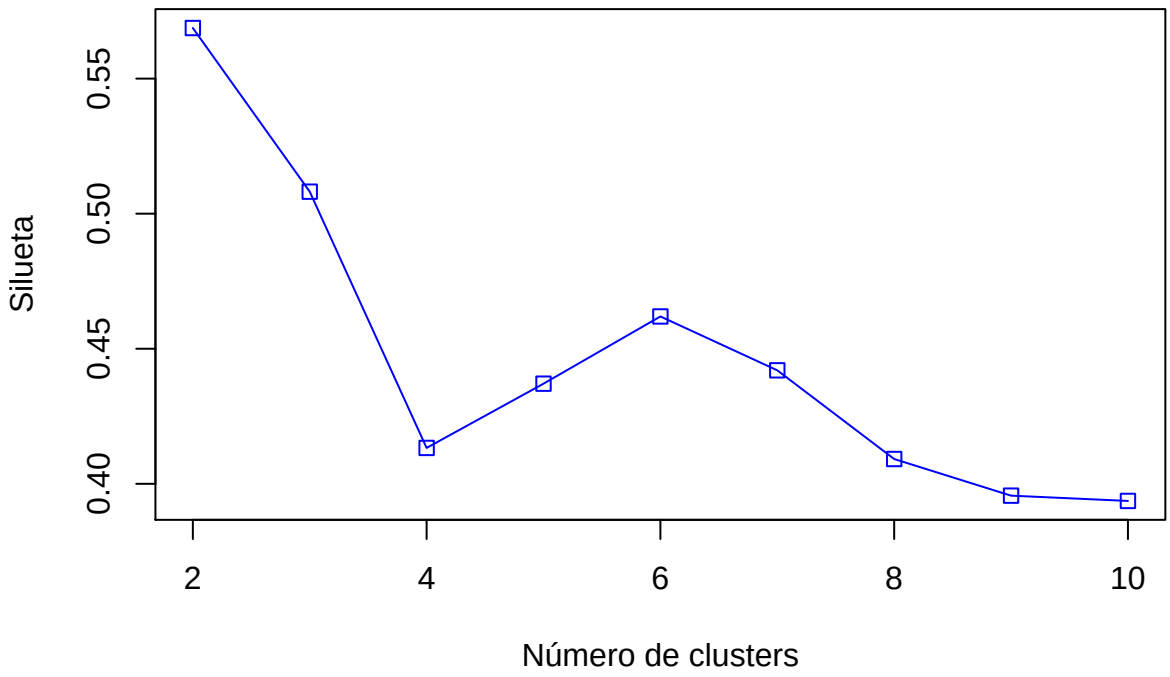
10% Se aplica el algoritmo de agrupamiento de forma correcta

Como inicialmente no conocemos el número óptimo de clústers, probamos con varios valores

```
library(cluster) # cargamos la librería que permite trabajar con las funciones de clustering

d <- daisy(data)
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(data, i)
  y_cluster     <- fit$cluster
  sk            <- silhouette(y_cluster, d)
  resultados[i] <- mean(sk[,3])
}
```

Mostramos en un gráfica los valores de las siluetas media de cada prueba para comprobar que número de clústers es el mejor

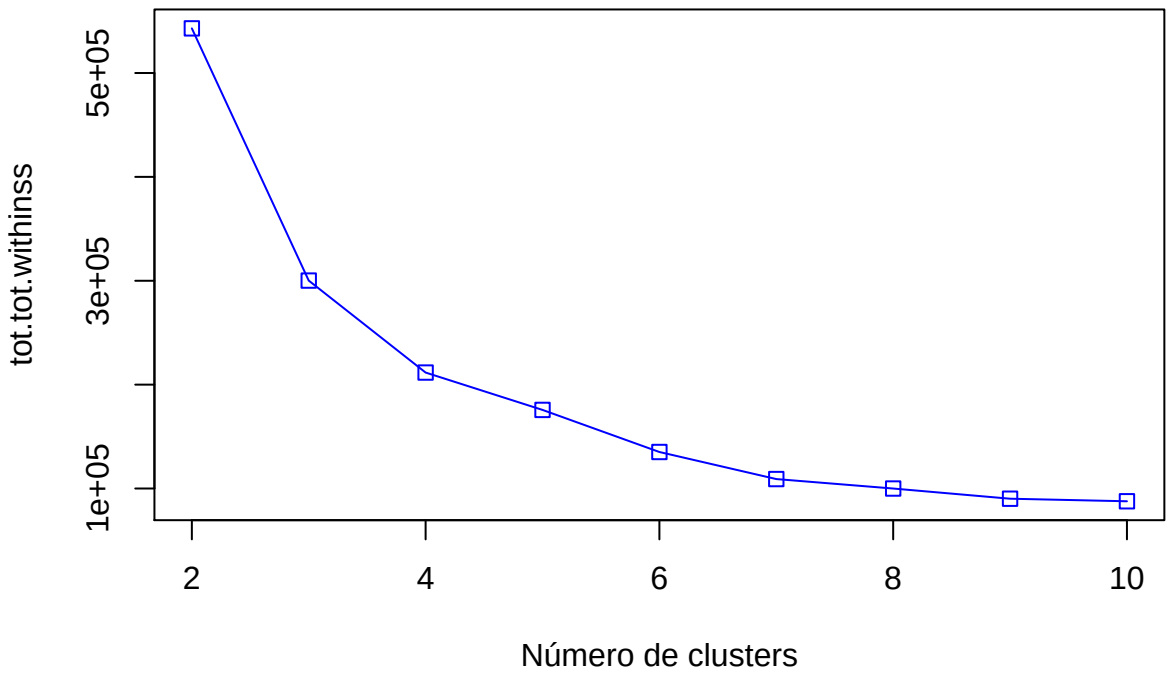
```
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="Silueta")
```



Interpretando la gráfica anterior, el mejor valor que se obtiene es k=2. Aunque las clases definidas en el artículo mencionado anteriormente son 6 (k=6) y en el dataset original existen 7 (k=7) clases.

Otro forma de evaluar cual es el mejor número de clústers es considerar el mejor modelo, aquel que ofrece la menor suma de los cuadrados de las distancias de los puntos de cada grupo con respecto a su centro (withinss), con la mayor separación entre centros de grupos (betweenss). Como se puede comprobar es una idea conceptualmente similar a la silueta. Una manera común de hacer la selección del número de clústers consiste en aplicar el método elbow (codo), que no es más que la selección del número de clústers en base a la inspección de la gráfica que se obtiene al iterar con el mismo conjunto de datos para distintos valores del número de clústers. Se seleccionará el valor que se encuentra en el "codo" de la curva

```
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(data, i)
  resultados[i] <- fit$tot.withinss
}
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="tot.tot.withinss")
```



En este caso el número óptimo de clústers son 4 que es cuando la curva comienza a estabilizarse.

También se puede usar la función kmeansruns del paquete fpc que ejecuta el algoritmo kmeans con un conjunto de valores, para después seleccionar el valor del número de clústers que mejor funcione de acuerdo a dos criterios: la silueta media ("asw") y Calinski-Harabasz ("ch").

```
library(fpc)
fit_ch  <- kmeansruns(data, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(data, krange = 1:10, criterion = "asw")
```

Podemos comprobar el valor con el que se ha obtenido el mejor resultado y también mostrar el resultado obtenido para todos los valores de k usando ambos criterios

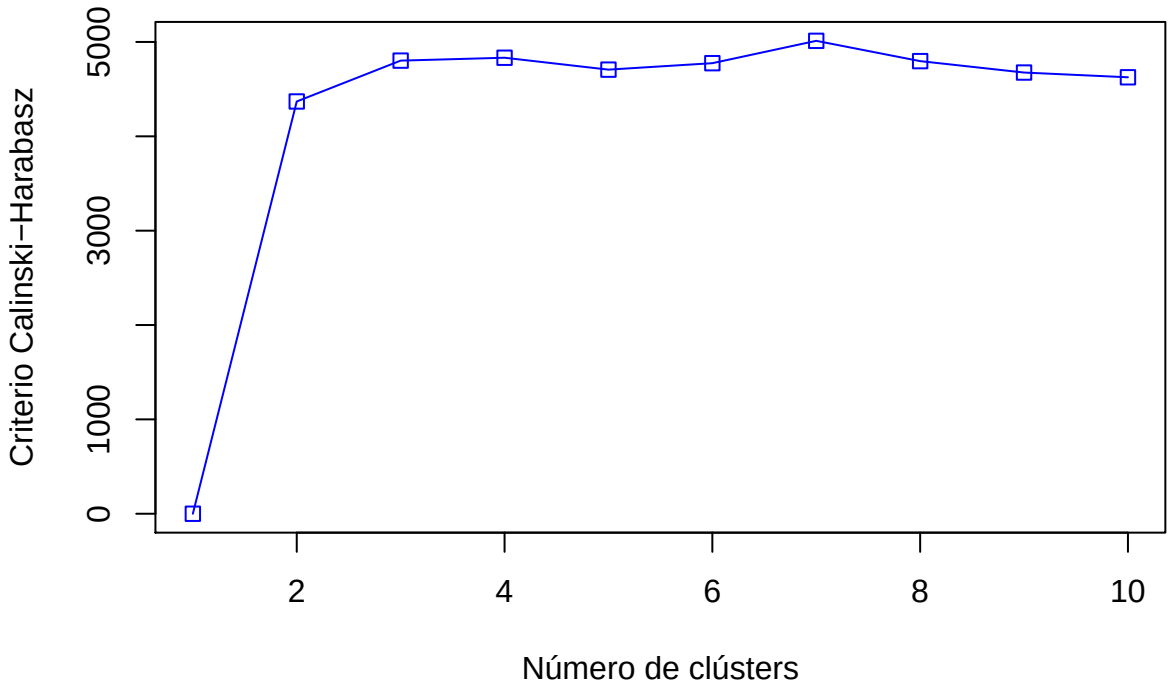
```
fit_ch$bestk
```

```
## [1] 7
```

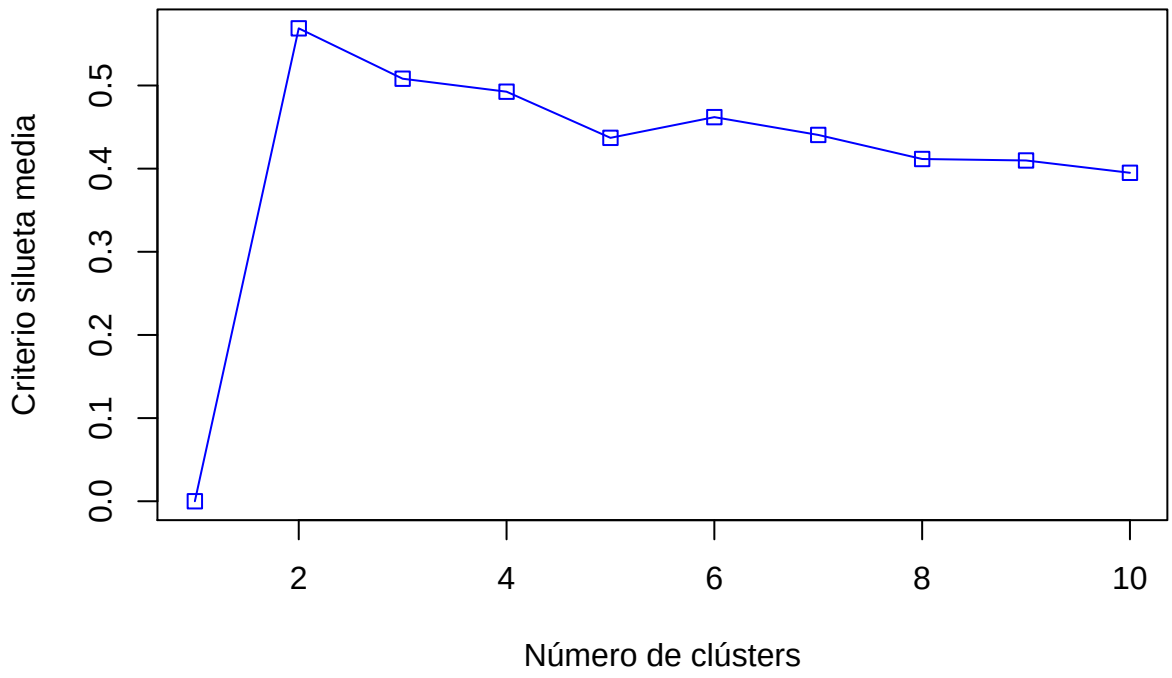
```
fit_asw$bestk
```

```
## [1] 2
```

```
plot(1:10,fit_ch$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio Calinski-Haraba
```




```
plot(1:10,fit_asw$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio silueta media
```



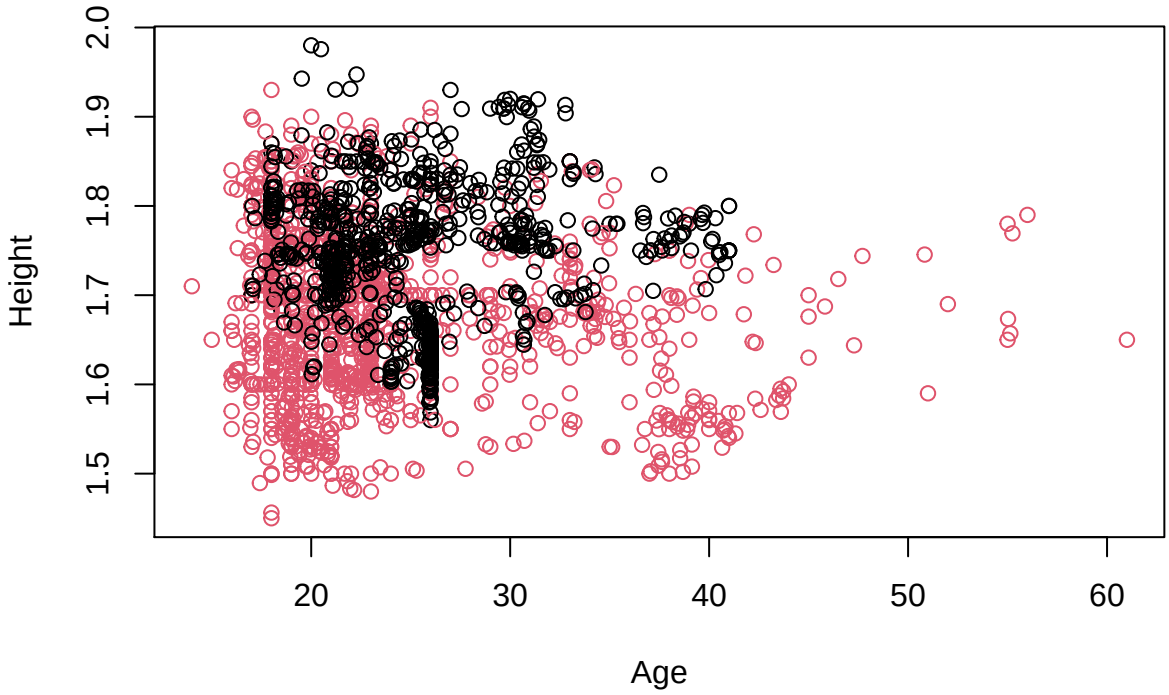
25% Se prueban con diferentes valores de k

Los resultados obtenidos son muy diferentes. Para el caso de las siluetas medias obtuvimos 2 y luego con el método elbow (codo) obtuvimos 4. Finalmente usando la función kmeansruns y aplicando los criterios Calinski-Harabasz y silueta media, se obtiene 7 y 2, respectivamente. Notamos que el valor k=2 coincide con el obtenido inicialmente.

Como sabemos de antemano en el dataset original existen 7 clases, que coinciden con el valor bestk=7 obtenido para el criterio Calinski-Harabasz. Ahora vamos a analizar los datos de manera visual, comparandolos de 2 en 2, con el valor real que sabemos está almacenado en el campo "NObeyesdad" del dataset original.

Vamos a analizar primero con el valor de k=2, para el par de atributos Age (Edad) y Height (Altura):

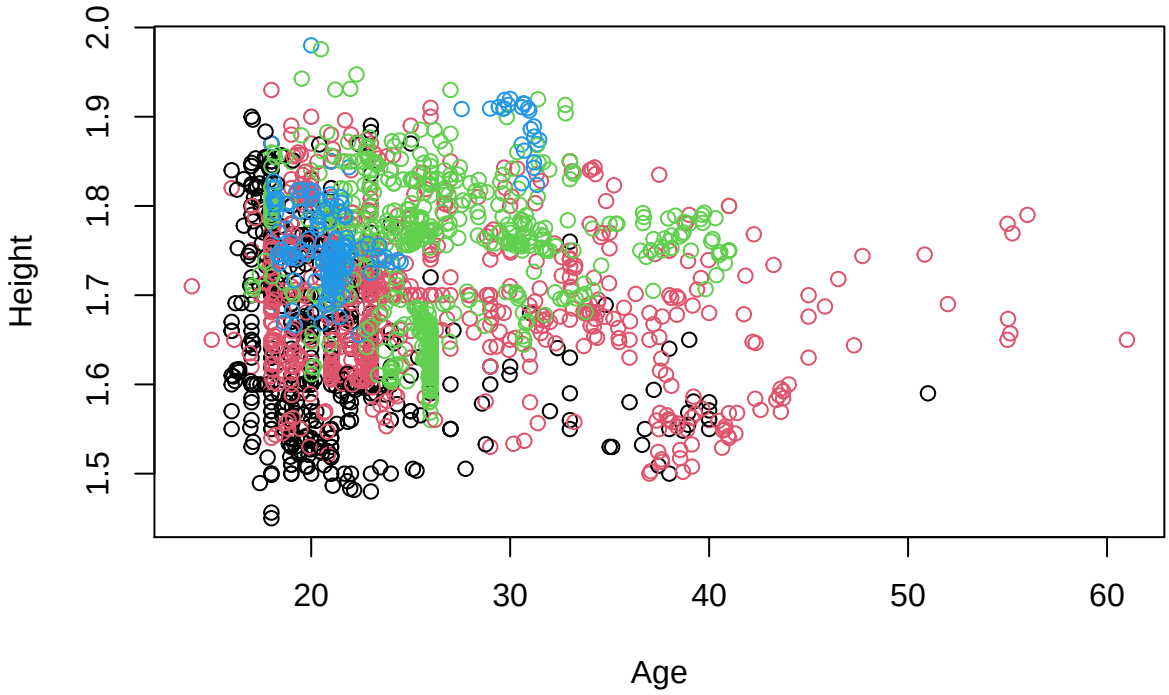
```
obesity2clusters <- kmeans(data, 2)

# Edad y Altura
plot(data[c(1,2)], col=obesity2clusters$cluster)
```



Este valor de k=2 no ayuda a definir correctamente los grupos, ya que están muy dispersos y entremezclados.

Ahora vamos a analizar con el otro valor de k=4 que se obtuvo. Graficando Age (Edad) vs. Height (Altura), se obtiene:

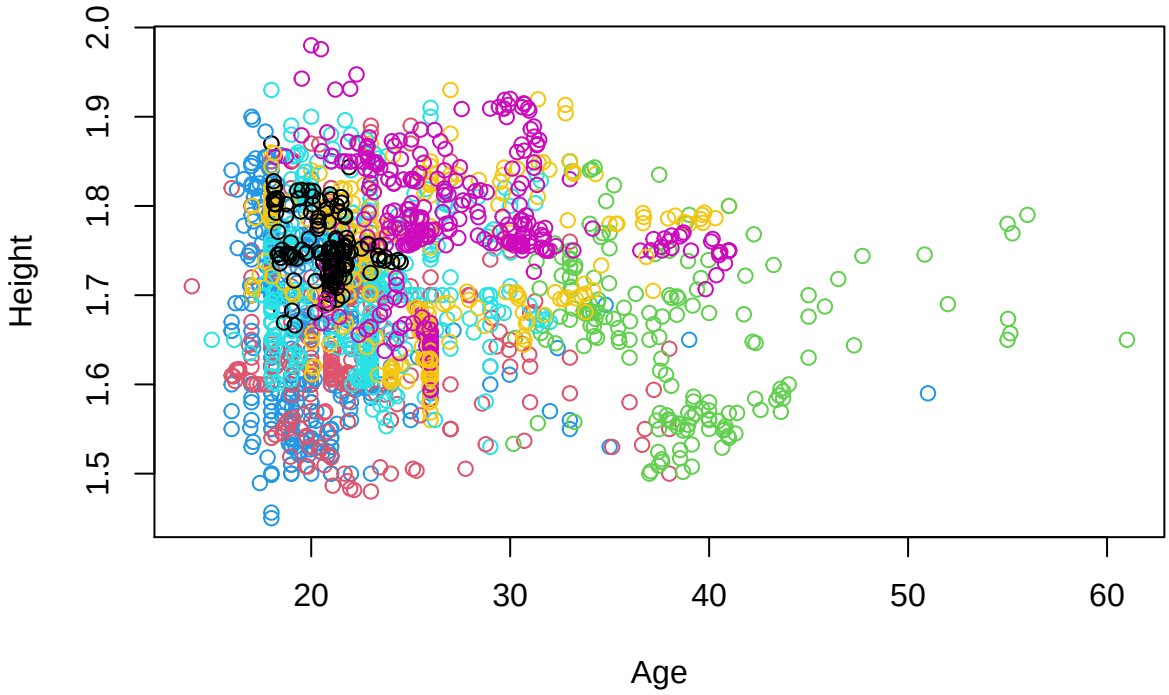
```
obesity4clusters <- kmeans(data, 4)

# Edad y Altura
plot(data[c(1,2)], col=obesity4clusters$cluster)
```



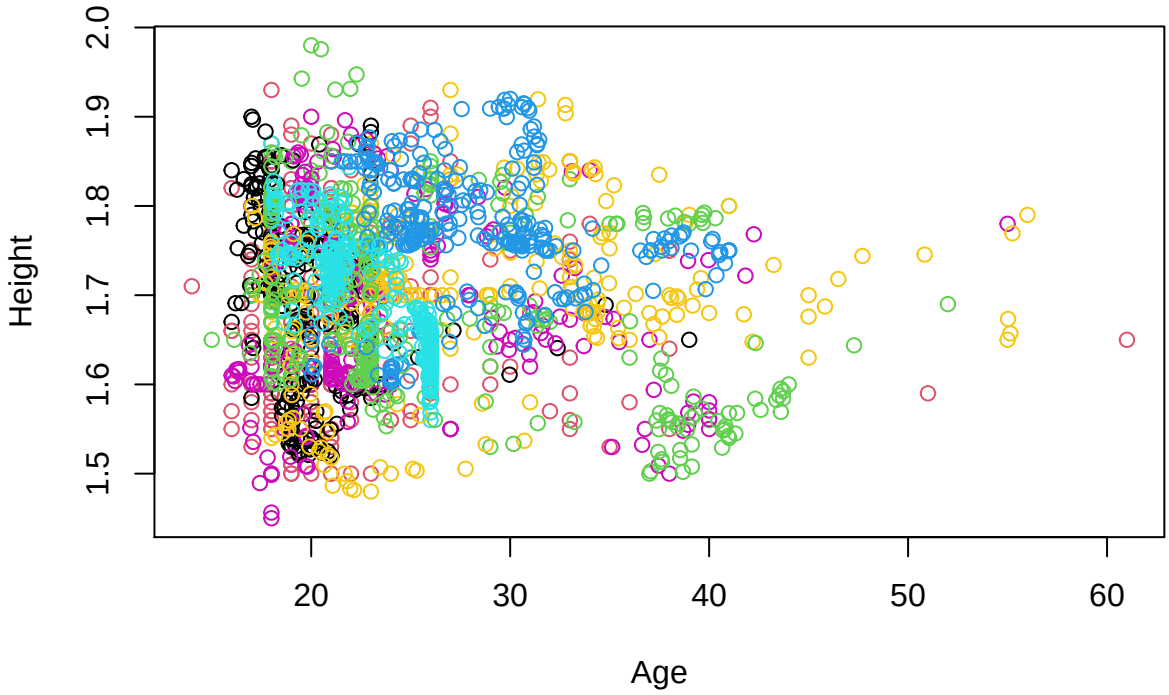
Este valor de k=4 al igual que el valor de k=2 no ayuda a definir correctamente los grupos, ya que vemos que se entremezclan demasiado los objetos evaluados.

Ahora vamos a analizar con el valor de k=7, para el par de atributos Age (Edad) vs. Height (Altura):

```
obesity7clusters <- kmeans(data, 7)

# Height y mass_body_idx
plot(data[c(1,2)], col=obesity7clusters$cluster)
```




```
plot(data[c(1,2)], col=as.factor(obesity_data_all$NObeyesdad))
```

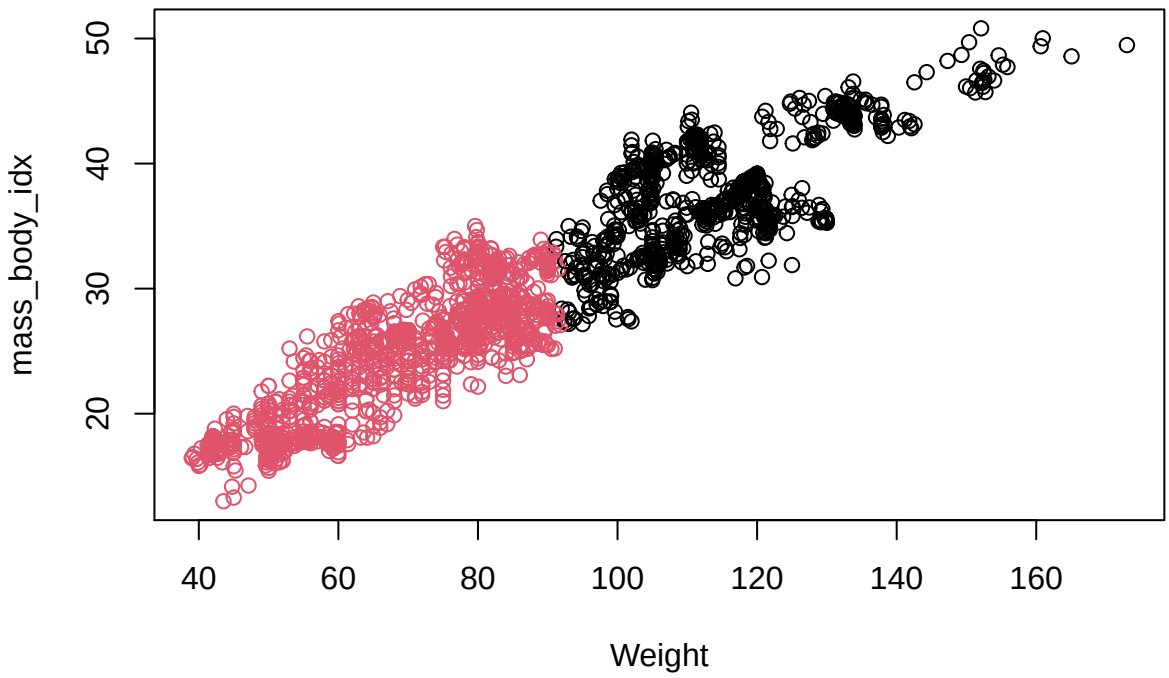


Vemos que a medida que se aumenta el valor de k se entremezclan más los grupos, por tal motivo no es una buena elección haber elegido la edad y la altura como variables para agrupar. Es razonable, ya que la altura depende de muchos factores, entre ellos la edad, pero será necesario analizarlo con otros atributos de los individuos.

Ahora procedamos a analizar los clusters Weight (Peso) vs. mass_body_idx (índice de masa corporal) y los mismos valores de k.

Para k=2:

```
# Weight y mass_body_idx
obesity2WMclust <- kmeans(data, 2)

plot(data[c(3,4)], col=obesity2WMclust$cluster)
```

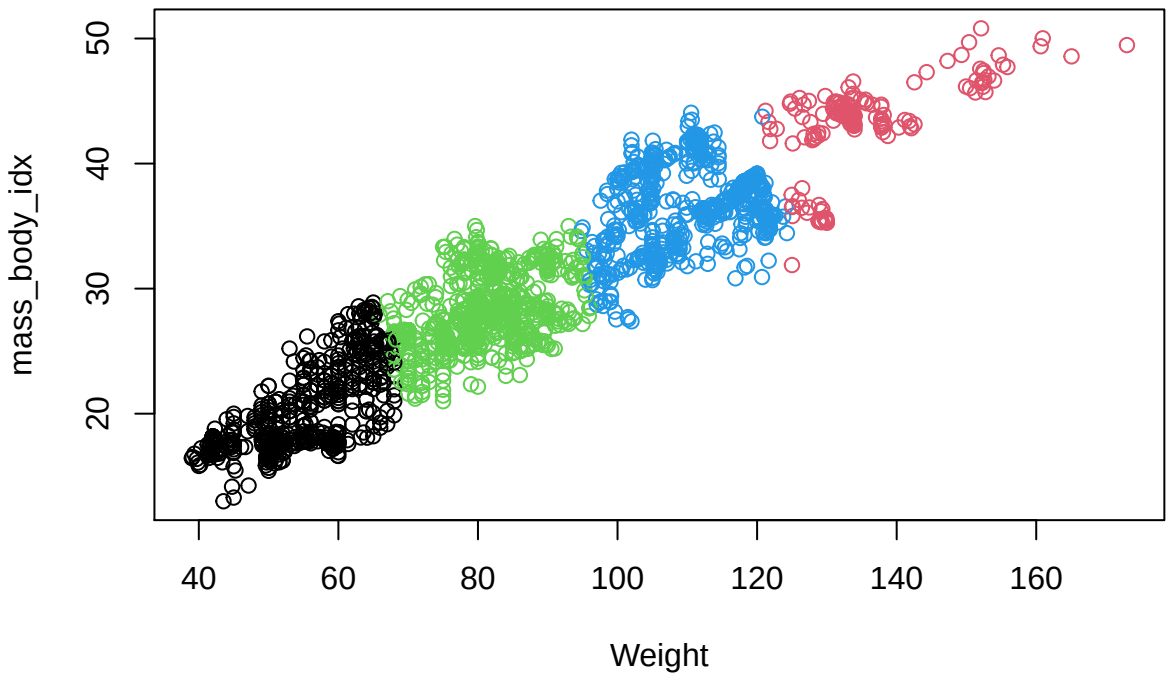


Vemos que para el análisis entre el peso y el índice de masa corporal, para el valor de k=2, se obtiene 2 grupos claramente definidos. Sabemos que el índice de masa corporal es un atributo calculado en base al peso, es decir que existe una relación entre los 2 valores.

Además podemos comprobar que el valor de k=2, obtenido mediante silueta media efectivamente refleja una estimación de calidad alta.

Ahora para k=4:

```
# Weight y mass_body_idx
obesity4WMclust <- kmeans(data, 4)

plot(data[c(3,4)], col=obesity4WMclust$cluster)
```



Vemos que para el análisis entre el peso y el índice de masa corporal, para el valor de k=4, se obtiene 4 grupos claramente definidos al igual que para k=2.

Los centros de cada cluster se obtiene para su análisis posterior:

```
# Weight y mass_body_idx
obesity4WMclust$centers
```

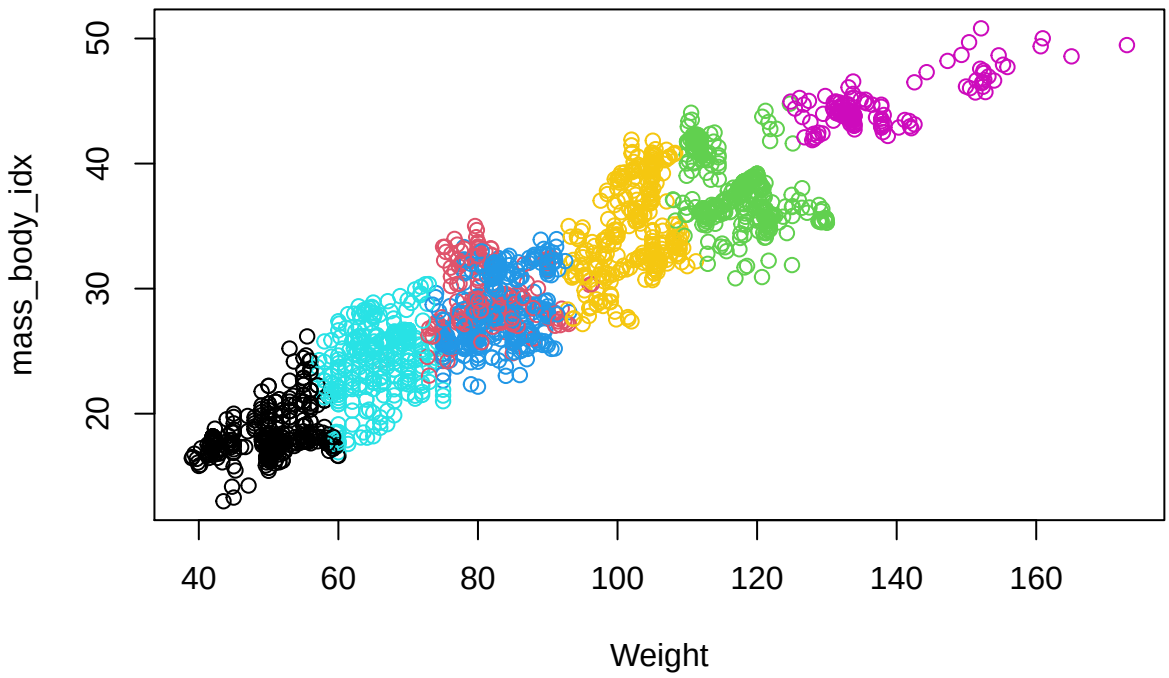
```
##        Age   Height    Weight mass_body_idx
## 1 20.92346 1.652483  54.81562      20.20531
## 2 22.01694 1.772465 135.29890      43.18440
## 3 25.62693 1.696666  80.66900      28.12623
## 4 26.43884 1.733822 109.85574      36.70856
```

Los puntos del centro en el cluster son:

- Grupo 1: [51.30, 18.67]
- Grupo 2: [69.49, 25.15]
- Grupo 3: [86.28, 29.72]
- Grupo 4: [117.35, 38.76]

Ahora para k=7:

```
# Weight y mass_body_idx
obesity7WMclust <- kmeans(data, 7)

plot(data[c(3,4)], col=obesity7WMclust$cluster)
```



Analizamos también los centros de cada cluster

```
# Weight y mass_body_idx
obesity7WMclust$centers
```

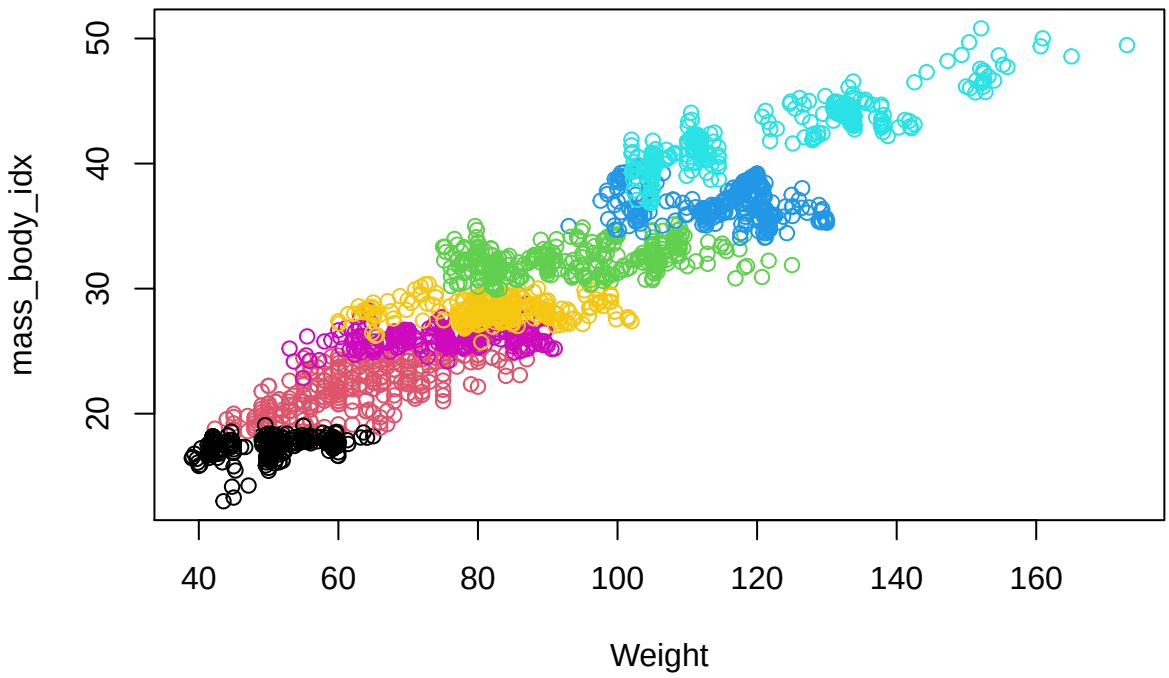
```
##        Age   Height    Weight mass_body_idx
## 1 20.05083 1.654755  50.02929      18.31944
## 2 37.64863 1.668116  81.27307      29.34083
## 3 27.22958 1.758044 116.70929      37.91304
## 4 21.88745 1.715398  82.49517      28.15688
## 5 22.20933 1.653384  65.59052      24.19031
## 6 20.59883 1.754183 137.32817      44.59289
## 7 25.27152 1.719509 102.27281      34.83355
```

Para este valor de k=7 los grupos no están claramente definidos como para los otros valores de k. Los grupos que se encuentran entremezclados son los que tienen por centros:

- Centro 1(Grupo 7): [83.84, 28.52]
- Centro 2(Grupo 8): [81.07, 29.42]

Lo comparamos con el valor real que sabemos está almacenado en el campo "NObeyesdad" del dataset original.

```
plot(data[c(3,4)], col=as.factor(obesity_data_all$NObeyesdad))
```



10% Se obtiene una medida de lo bueno que es el agrupamiento

Ahora vamos a evaluar la calidad del proceso de agregación. Para ello usaremos la función silhouette que calcula la silueta de cada muestra. Para los valores de k=2, k=4 y k=7 obtenemos que:

```
d  <- daisy(data)
sk2 <- silhouette(obesity2WMclust$cluster, d)
sk4 <- silhouette(obesity4WMclust$cluster, d)
sk7 <- silhouette(obesity7WMclust$cluster, d)
```

La función silhouette devuelve para cada muestra, el clúster dónde ha sido asignado, el clúster vecino y el valor de la silueta. Por lo tanto, calculando la media de la tercera columna podemos obtener una estimación de la calidad del agrupamiento

```
mean(sk2[,3])
```

```
## [1] 0.5687273
```

```
mean(sk4[,3])
```

```
## [1] 0.4925383
```

```
mean(sk7[,3])
```

```
## [1] 0.4392343
```

Según los valores obtenidos previamente, sobre la calidad del clustering para cada valor de k, tenemos que agrupar los valores en 2 clusters es mejor que en 4 y/o 8.

Sin embargo, es necesario recordar lo mencionado al inicio de este ejercicio.

10% Se ponen nombres a los clústers 20% Se describen e interpretan los diferentes clústers obtenidos

Si tomamos como referencia que el IMC óptimo agrupa a los individuos en al menos 3 grupos, no podemos entonces considerar el valor de k=2 como definitivo. Por lo cual lo descartamos. Sin embargo, luego de considerar los valores k=4 y k=7 de manera gráfica y su respectiva medida de calidad del agrupamiento, se concluye que es mejor asociar los individuos en 4 clusters, los cuales serían:

- Grupo 1: Peso Bajo, donde el valor del centro en el cluster para el espacio Peso vs. IMC es [51.30, 18.67]. Es decir que los individuos que tiene un IMC entre 18.67 y 25.15 se corresponden a este grupo.
- Grupo 2: Peso Normal, donde el valor del centro en el cluster para el espacio Peso vs. IMC es [69.49, 25.15]. Es decir que los individuos que tiene un IMC entre 25.15 y 29.72 se corresponden a este grupo.
- Grupo 3: Peso Alto, donde el valor del centro en el cluster para el espacio Peso vs. IMC es [86.28, 29.72]. Es decir que los individuos que tiene un IMC entre 29.72 y 38.76 se corresponden a este grupo.
- Grupo 4: Peso muy Alto donde el valor del centro en el cluster para el espacio Peso vs. IMC es [117.35, 38.76]. Es decir que los individuos que tiene un IMC superior a 38.76 se corresponden a este grupo.

Esto nos ayuda a entender cómo están formados los grupos y a referirnos a ellos en análisis posteriores.

> **NOTA**: Las clases del dataset original eran 7, las cuales se obtuvieron analizando más variables relacionadas con los hábitos alimenticios y condición física de cada individuo, aplicando otras técnicas de clasificación que no corresponden al alcance de este ejercicio ni al objetivo de esta PEC.

10% Se presenta el código y es fácilmente reproducible

Ejercicio 1.2

Buscar información sobre otros métodos de agregación diferentes al Kmeans. Partiendo del ejercicio anterior probar el funcionamiento de al menos 2 métodos diferentes y comparar los resultados obtenidos.

Respuesta 1.2

25% Se prueba un algoritmo diferente al kmeans

Clustering Jerárquico aglomerativo. El primero método seleccionado es el método Agglomerative Hierarchical Clustering que agrupa los individuos en base a su distancia en el cluster con sus vecinos. Esto inicia en la base del árbol (dendograma), donde cada observación forma un cluster individual. Los clusters se van combinado a medida que la estructura crece hasta converger en una única "rama" central.

```
data_scaled <- scale(data)

# Matriz de distancias euclídeas
mat_dist <- dist(x = data_scaled, method = "euclidean")

# Dendrogramas con linkage complete y average
hc_euclidea_complete <- hclust(d = mat_dist, method = "complete")
# Dendrogramas con linkage average
```

```r
hc_euclidea_average  <- hclust(d = mat_dist, method = "average")
# Dendrogramas con linkage ward
hc_euclidea_ward  <- hclust(d = mat_dist, method = "ward.D")

cor(x = mat_dist, cophenetic(hc_euclidea_complete))
```

```
## [1] 0.5282904
```

```r
cor(x = mat_dist, cophenetic(hc_euclidea_average))
```

```
## [1] 0.6363252
```

```r
cor(x = mat_dist, cophenetic(hc_euclidea_ward))
```

```
## [1] 0.5637662
```

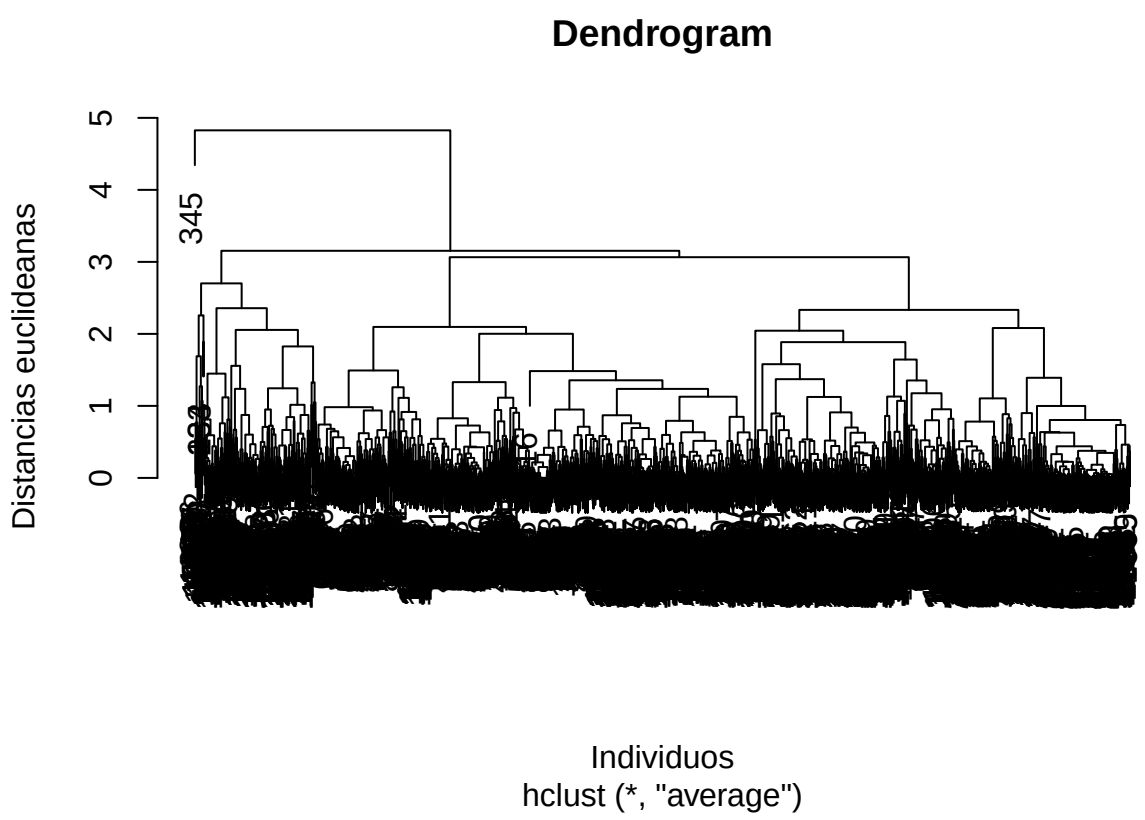
Vemos que al momento de evaluar las distancias originales entre observaciones usando el coeficiente de correlación cophenetic, obtenemos que el mejor método es **"average"** con un valor de 0.64. Entre más se acerca el valor a 1, mejor refleja el dendrograma la verdadera similitud entre las observaciones

Ahora vamos a evaluar su dendograma.

Dendograma de clusters. Primero procedamos a graficarlo para poder determinar el número de clusters que se podrían podrían obtener. Cabe recalcar que en este método no se requiere establecer de antemano el número de cluster. En esta ocasión la gráfica puede ayudarnos a determinar k.

Usaremos además método average para obtener la distancia euclidea entre los individuos a analizarse.

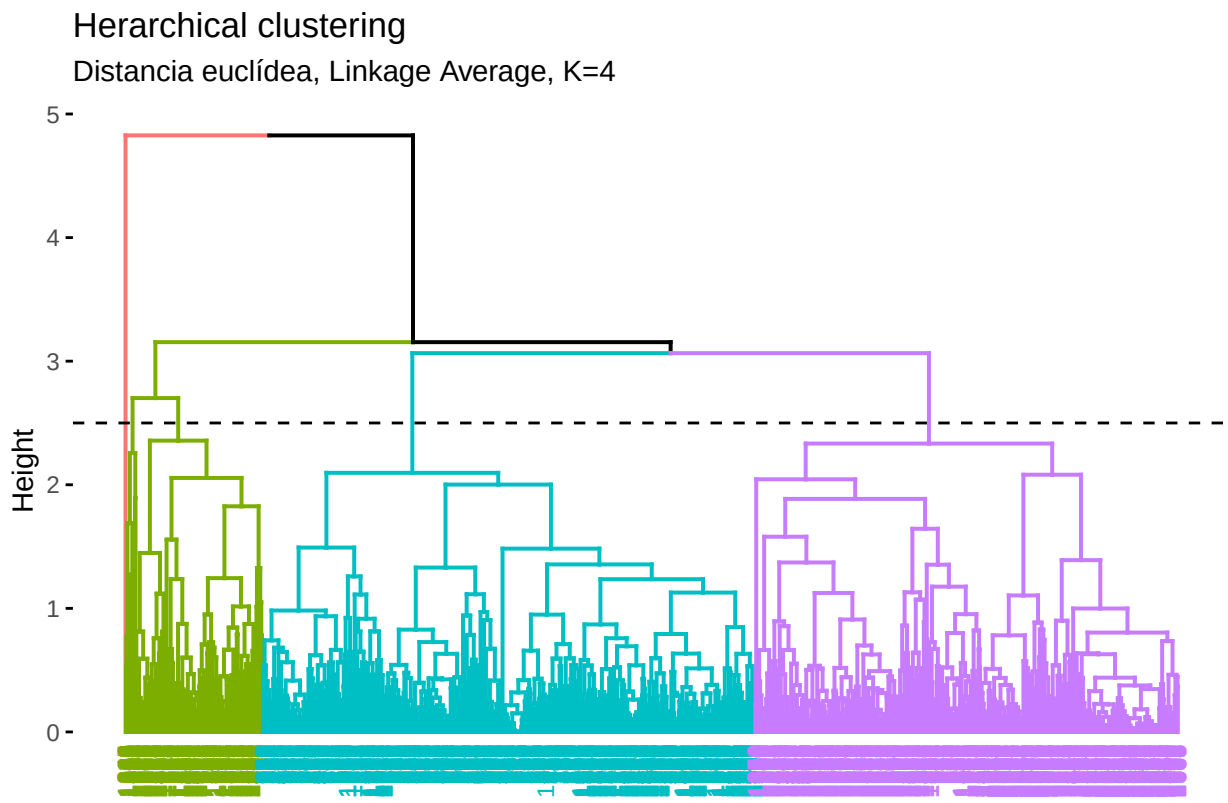
```r
# El método average intenta minimizar la varianza del grupo
dendrogram = hclust(d = dist(data_scaled, method = 'euclidean'), method = 'average')
plot(dendrogram,
     main = paste('Dendrogram'),
     xlab = 'Individuos',
     ylab = 'Distancias euclideanas')
```



Analizando el dendograma, si quiero tener 4 clusters al igual que el valor de k para el método k-means, la altura de corte es aproximadamente en 2.5. Cabe recalcar que la línea de corte tiene funcion similar al valor de k en el método k-means, es decir controla el número de clusters obtenidos.

Probamos para 4 clusters (k) y altura (yintercept) de 2.5:

```
library(factoextra)

set.seed(101)

hc_euclidea_average <- hclust(d = dist(x = data_scaled, method = "euclidean"), method = "average")

fviz_dend(x = hc_euclidea_average, k = 4, cex = 0.6) + geom_hline(yintercept = 2.5, linetype = "dashed"
```



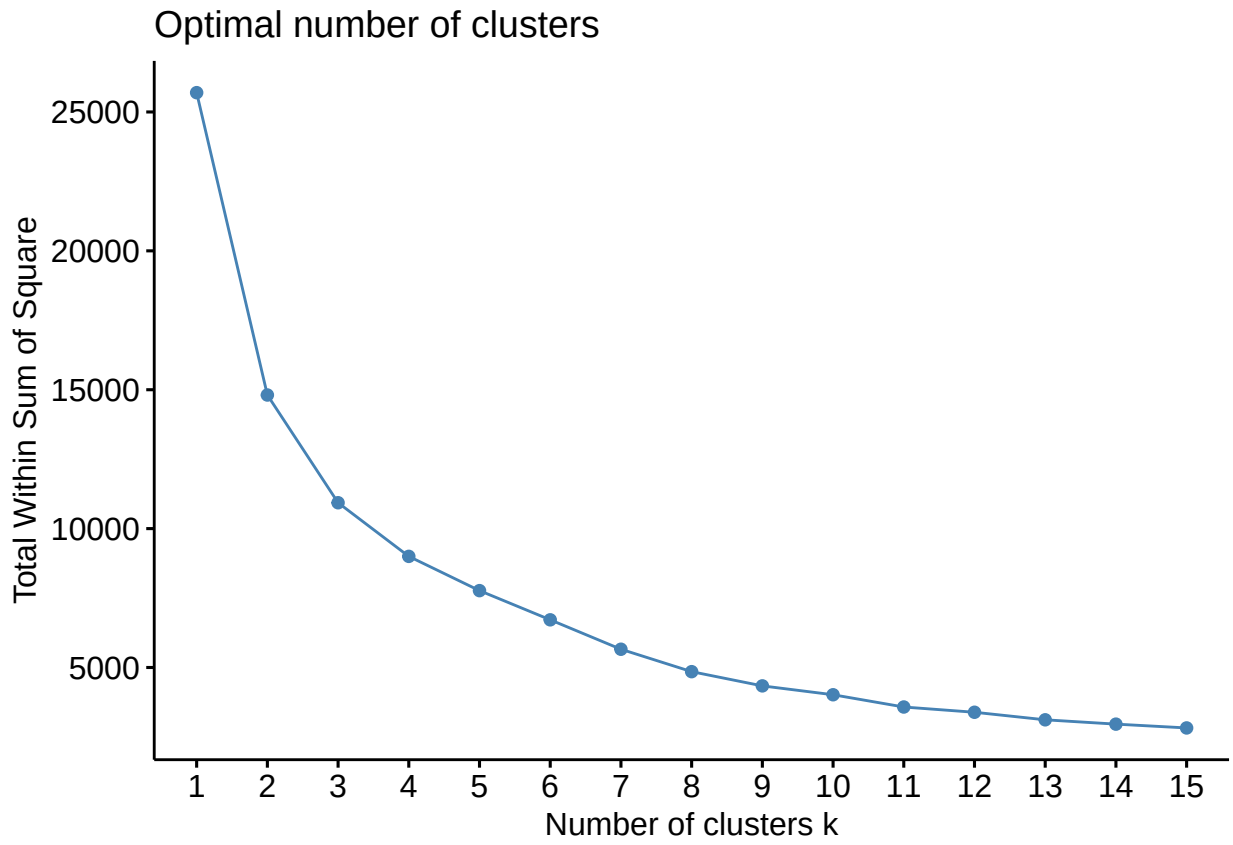
Como se puede ver en la gráfica la linea en 2.5 corta al dendograma en 4 puntos, para establecer los 4 clusters.

25% Se prueba otro algoritmo diferente al kmeans

K-medoids clustering (PAM) K-medoids es un método de clustering muy similar a K-means en cuanto a que ambos agrupan las observaciones en k clusters, donde K es un valor preestablecido por el analista. La diferencia es que, en K-medoids, cada cluster está representado por una observación presente en el cluster (medoid), mientras que en K-means cada cluster está representado por su centroide, que se corresponde con el promedio de todas las observaciones del cluster pero con ninguna en particular.

Comenzamos a realizar el análisis de los datos usando este método. Vamos a usar la técnica del codo, para obtener el valor aproximado de K según la curva de la gráfica. Para esto vamos a usar la función fviz_nbclust del paquete factoextra. Esta función recibe como un parámetro a FUNcluster que será el que haga las particiones de los objetos analizados para identificar los clusters, según el algoritmo PAM.

```
library(cluster)
library(factoextra)

data_medoids <- scale(data)

fviz_nbclust(x = data_medoids, FUNcluster = pam, method = "wss", k.max = 15,
             diss = dist(data_medoids, method = "manhattan"))
```



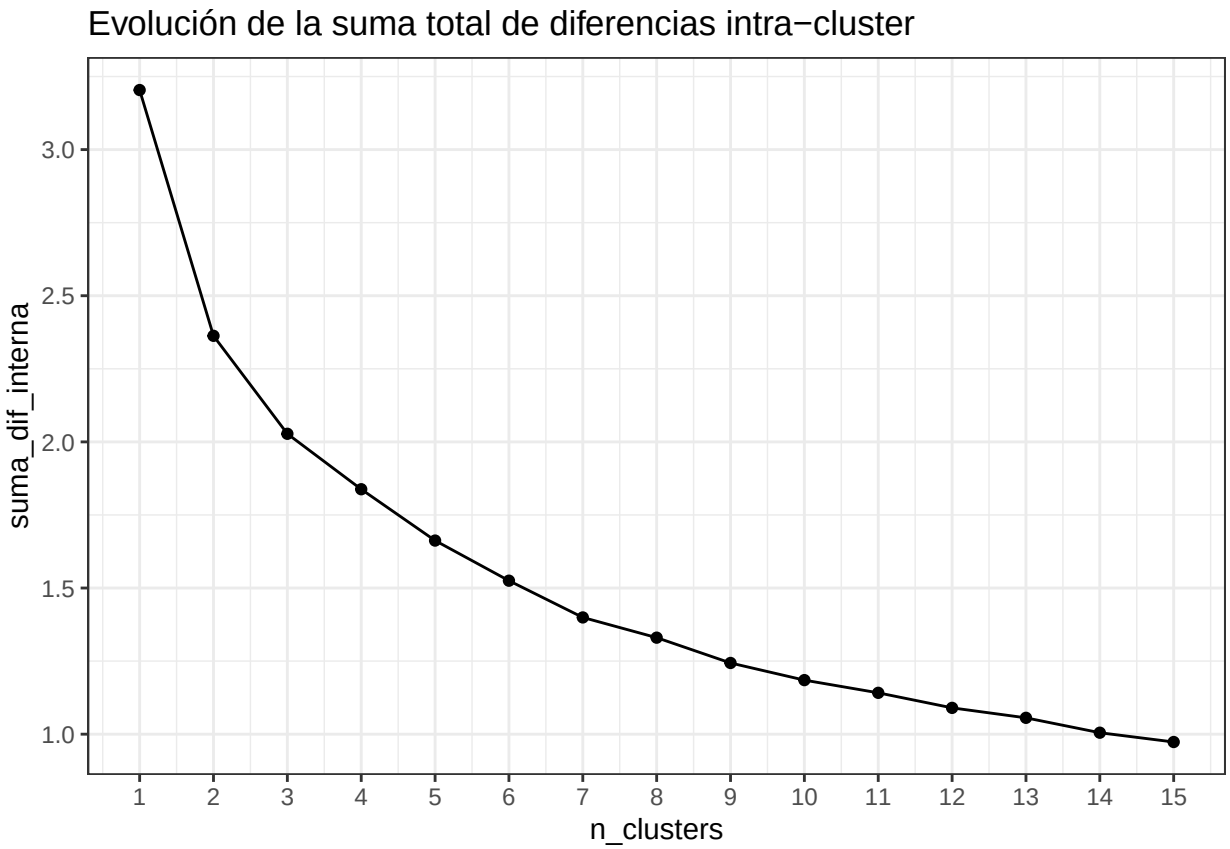
Vemos que la curva comienza a estabilizarse en entre los valores de 4 y 5.

Ahora vamos a probar con otro enfoque más elaborado para tener otra perspectiva. Este enfoque no usa la función fviz_nbclust. En esta función se aplica el algoritmo PAM para obtener la suma total de las diferencias internas.

NOTA: Este fragmento de código fue tomado del portal https://www.cienciadedatos.net [2]

```
library(purrr)
library(ggplot2)

# Mismo análisis pero sin recurrir a factoextra
# ==============================================================================
calcular_suma_dif_interna <- function(n_clusters, datos, distancia = "manhattan"){
  # Esta función aplica el algoritmo pam y devuelve la suma total de las
  # diferencias internas
  cluster_pam <- cluster::pam(x = datos, k = n_clusters, metric = distancia)
  # El objeto pam almacena la suma de las diferencias respecto a los medoides en
  # $objective["swap"]
  return(cluster_pam$objective["swap"])
}

# Se aplica esta función con para diferentes valores de k
suma_dif_interna <- map_dbl(.x = 1:15,
                            .f = calcular_suma_dif_interna,
                            datos = data_medoids)
data.frame(n_clusters = 1:15, suma_dif_interna = suma_dif_interna) %>%
```

```
ggplot(aes(x = n_clusters, y = suma_dif_interna)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:15) +
  labs(title = "Evolución de la suma total de diferencias intra-cluster") +
  theme_bw()
```



Vemos que los valores donde la curva se comienza a estabilizar son nuevamente entre 4 y 5.

40% Se comparan los resultados del kmeans y los otros dos métodos probados en este ejercicio

1. Agrupamiento jerárquico

Para el otro método seleccionado, **Agrupamiento jerárquico** tenemos que es un método de la categoría aglomerador o incrementa. Se diferencia de k-means porque al aplicar el algoritmo de dicho método se obtiene un árbol o dendograma, donde no se tiene ningún valor de k fijado de antemanos, sino que se establece a partir de la gráfica.

El agrupamiento jerárquico busca cuantificar la similitud entre dos clusters. Además interviene un nuevo concepto de enlace (linkage) que partiendo de clusters individuales se van agrupando en nuevos clusters al evaluar las distancias entre grupos (u objetos, en el primer paso), y creando los nuevos diferentes grupos finales por aglomeración.

Se obtuvo que el mejor linkage era “average”, usando el coeficiente de correlación cophenetic. Ya cuando se trazó el dendograma se procedió a inspeccionar que es posible obtener 4 clusters al cortar el dendrogama a una altura de 2.5

A manera de comparación con k-means es su alto consumo de recursos del CPU para realizar los calculos y trazo del dendograma. Llegó a tardar más de un minuto la obtención del dendrograma y la linea de corte.

Otro punto en contra de este método es que visualmente el dendrograma no es de la mejor calidad, debido al elevado número de hojas iniciales en el árbol. Si el dataset tuviera menos registros se obtendría un dendograma más claro.

2. Método k-medoids (PAM)

Para el método **k-medoids** se obtuvo los mismos resultados para k, ya que para ambos métodos es aplicable la técnica del codo, para lo cual en la gráfica que se corresponde con cada uno la curva comienza a estabilizarse al llegar al valor 4. Aunque para el método k-medoids la curva está mejor definida en comparación con la curva para k-means.

Cabe recalcar que se pudo notar que el método k-medoids requiere de muchos más recursos para realizar los calculos al momento de aplicar el algoitmo PAM. Por otro lado, la ligera rápidez ganada por k-means, el método k-medoid lo compensa siendo un método mucho más robusto, siendo particularmente recomendable para casos, como el dataset analizado, donde existen muchos o outliers o valores aislados.

10% Se presenta el código y es fácilmente reproducible

Ejemplo 2

Métodos de asociación

En este ejemplo vamos trabajar el algoritmo "apriori" para obtener reglas de asociación a partir de un data set Dichas reglas nos ayudarán a comprender cómo la información del data set se relaciona entre si. Para dicho objetivo vamos a trabajar el dataset de Groceries, que ya viene incluido con las librerías de arules.

```
# install.packages("arules")
library(arules)
data("Groceries")
```

Inspeccionamos el dataset y vemos que tiene un listado de elementos que fueron comprados juntos. Vamos a analizarlo un poco visualmente.

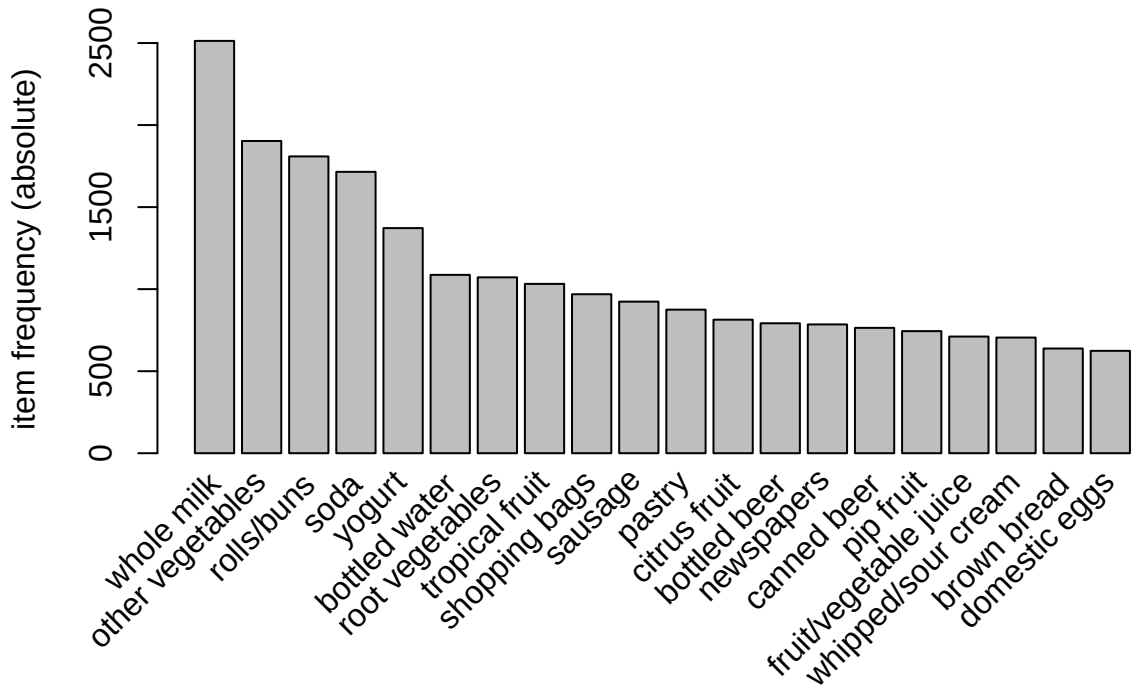
```
?Groceries
inspect(head(Groceries, 5))
```

```
##     items
## [1] {citrus fruit,
##      semi-finished bread,
##      margarine,
##      ready soups}
## [2] {tropical fruit,
##      yogurt,
##      coffee}
## [3] {whole milk}
```

```
## [4] {pip fruit,
##      yogurt,
##      cream cheese ,
##      meat spreads}
## [5] {other vegetables,
##      whole milk,
##      condensed milk,
##      long life bakery product}
```

En el siguiente plot podemos ver que los tres elementos más vendidos son la leche entera, otras verduras y bollería. Dada la simplicidad del Dataset no se pueden hacer mucho más análisis. Pero para datasets más complejos miraríamos la frecuencia y distribución de todos los campos, en busca de posibles errores.

```
itemFrequencyPlot(Groceries,topN=20,type="absolute")
```



Si lanzamos el algoritmo "apriori", generaremos directamente un set de reglas con diferente soporte, confianza y lift. El soporte indica cuantas veces se han encontrado las reglas {lsh => rhs} en el dataset, cuanto más alto mejor. La confianza habla de la probabilidad de que {rhs} se de en función de {lhs}. Y el lift es un parámetro que nos indica cuánto de aleatoriedad hay en las reglas. Un lift de 1 o menos es que las reglas son completamente fruto del azar.

```
grocery_rules <- apriori(Groceries, parameter = list(support = 0.01, confidence = 0.5))
```

```
## Apriori
##
## Parameter specification:
```

```
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(sort(grocery_rules, by = "confidence"), 3))
```

```
##     lhs                               rhs                support
## [1] {citrus fruit,root vegetables}   => {other vegetables} 0.01037112
## [2] {tropical fruit,root vegetables} => {other vegetables} 0.01230300
## [3] {curd,yogurt}                    => {whole milk}       0.01006609
##     confidence coverage   lift     count
## [1] 0.5862069  0.01769192 3.029608 102
## [2] 0.5845411  0.02104728 3.020999 121
## [3] 0.5823529  0.01728521 2.279125  99
```

Podemos probar a ordenar las reglas por los diferentes parámetros, para ver que información podemos obtener.

```
inspect(head(sort(grocery_rules, by = "support"), 3))
```

```
##     lhs                                      rhs          support    confidence
## [1] {other vegetables,yogurt}             => {whole milk} 0.02226741 0.5128806
## [2] {tropical fruit,yogurt}               => {whole milk} 0.01514997 0.5173611
## [3] {other vegetables,whipped/sour cream} => {whole milk} 0.01464159 0.5070423
##     coverage   lift     count
## [1] 0.04341637 2.007235 219
## [2] 0.02928317 2.024770 149
## [3] 0.02887646 1.984385 144
```

ordenando por support vemos que, con un lift de 2 y una confianza del 51%, podemos decir que la gente que en la misma compra hacía verduras y yogurt, compraban también leche entera. Hay que tener en cuenta que la leche entera es por otro lado el elemento más vendido de la tienda.

```
inspect(head(sort(grocery_rules, by = "lift"), 3))
```

```
##     lhs                               rhs                support
```

```
## [1] {citrus fruit,root vegetables}   => {other vegetables} 0.01037112
## [2] {tropical fruit,root vegetables} => {other vegetables} 0.01230300
## [3] {root vegetables,rolls/buns}     => {other vegetables} 0.01220132
##     confidence coverage   lift     count
## [1] 0.5862069  0.01769192 3.029608 102
## [2] 0.5845411  0.02104728 3.020999 121
## [3] 0.5020921  0.02430097 2.594890 120
```

Por otro lado, si ordenamos por lift, vemos que con un soporte del 1% y una confianza del 58%, la gente que compra cítricos y tubérculos compra también verduras

Esta información nos puede ayudar a dar consejos a la dirección de la disposición de los elementos en la tienda o de que productos poner en oferta según lo que se ha comprado. Y si tuviéramos más información podríamos hacer análisis más profundos y ver que clientes compran exactamente qué.

Ejercicio 2.1:

En este ejercicio seguiréis los pasos del ciclo de vida de un proyecto de minería de datos para el caso de un algoritmo de generación de reglas de asociación. Lo haréis con el fichero Lastfm.csv que encontraréis adjunto. Este fichero contiene un conjunto de registros. Estos registros son el histórico de las canciones que ha escuchado un usuario (user) en un portal Web de música. "artist" es el nombre del grupo que ha escuchado, sex y country corresponden a variables que describen al usuario.

Respuesta 2.1:

10% Se realiza un resumen de los datos incluidos en la base de datos

Este ejercicio consiste en extraer las reglas de asociación en los datos contenidos en el dataset lastfm.csv. El alcance solo será extraer estar reglas respecto al campo artist. Pero antes vamos a cargar el dataset y resumir la información contenida en el archivo.

```
library(arules)
library(dplyr)
library(plyr)

#Leemos el archivo CSV original
historico_artist <- read.csv("lastfm.csv", sep=",", encoding = "UTF-8", stringsAsFactors = TRUE)

summary(historico_artist)
```

```
##       user                         artist       sex
##  Min.   :    1   radiohead            :  2704   f: 78132
##  1st Qu.: 4935   the beatles          :  2668   m:211823
##  Median : 9838   coldplay             :  2378
##  Mean   : 9852   red hot chili peppers:  1786
##  3rd Qu.:14769   muse                 :  1711
##  Max.   :19718   metallica            :  1670
##                  (Other)              :277038
##            country
##  United States : 59558
##  United Kingdom: 27638
##  Germany       : 24251
##  Poland        : 17111
```

```
##  Sweden         : 12379
##  Brazil         : 11922
##  (Other)        :137096
```

El archivo contiene 4 atributos: user, artist, sex y country. Para efectos del ejercicio solo vamos a trabajar con las variables user y artist con las cuales formaremos las reglas de asociación.

Primero, vamos a proceder a preparar los datos

15% Se preparan los datos de forma correcta

```
# Creamos un array con los atributos user y artist, para poder agruparlos
array_user_artist <- historico_artist[, 1:2]

str(array_user_artist)
```

```
## 'data.frame':    289955 obs. of  2 variables:
##  $ user  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ artist: Factor w/ 1004 levels "...and you will know us by the trail of dead",..: 715 851 374 281 5
```

```
# Procedemos a transformar en formato transacciones
lista_artist <- ddply(array_user_artist, c("user"), function(df1)paste(df1$artist, collapse = ","))

# Eliminamos la columna user, por ya no ser necesaria
lista_artist$user <- NULL

# Dividimos la cadena en elementos. Aplicamos parametros para caracteres especiales
items <- strsplit(as.character(lista_artist$V1), ",", perl = TRUE, useBytes = FALSE)

# Obtenemos las transacciones
transacciones <- as(items, "transactions")

#Inspeccionamos las transacciones
inspect(head(transacciones, 5))
```

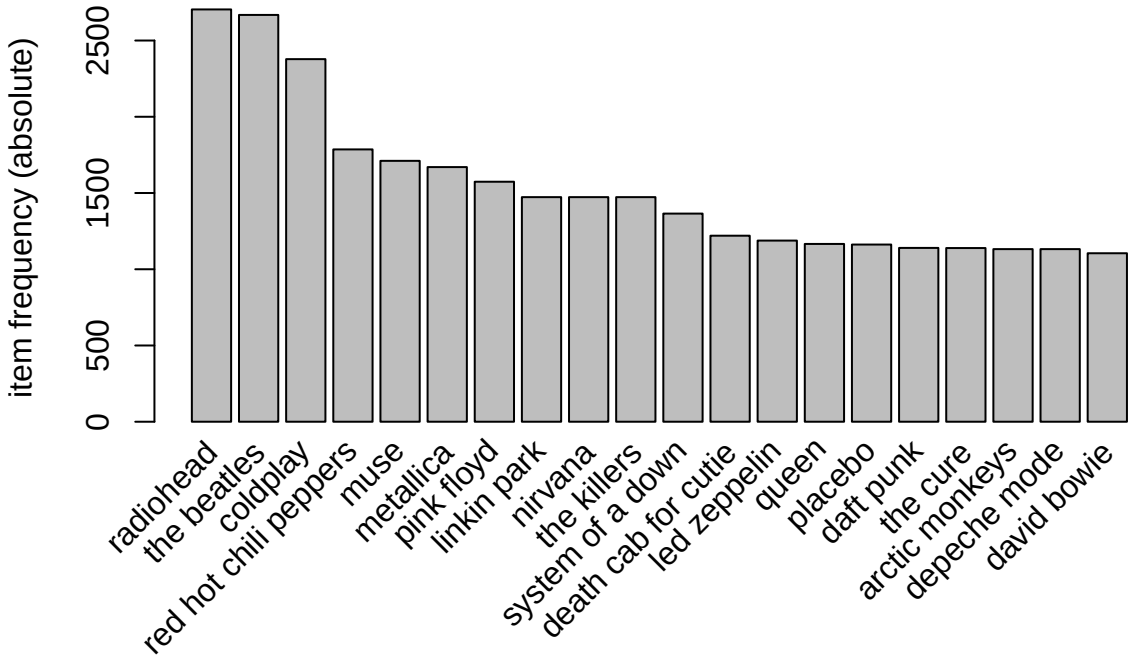
```
##     items
## [1] {dropkick murphys,
##      edguy,
##      eluveitie,
##      goldfrapp,
##      guano apes,
##      jack johnson,
##      john mayer,
##      judas priest,
##      le tigre,
##      red hot chili peppers,
##      rob zombie,
##      schandmaul,
##      the black dahlia murder,
##      the killers,
##      the rolling stones,
##      the who}
## [2] {aesop rock,
##      air,
```

```
##      amon tobin,
##      animal collective,
##      aphex twin,
##      arcade fire,
##      atmosphere,
##      autechre,
##      beastie boys,
##      boards of canada,
##      broken social scene,
##      cocorosie,
##      devendra banhart,
##      four tet,
##      goldfrapp,
##      joanna newsom,
##      m83,
##      massive attack,
##      max richter,
##      mf doom,
##      neutral milk hotel,
##      pavement,
##      plaid,
##      portishead,
##      prefuse 73,
##      radiohead,
##      sage francis,
##      the books,
##      the flashbulb}
## [3] {a tribe called quest,
##      air,
##      battles,
##      beck,
##      bon iver,
##      bonobo,
##      dj shadow,
##      fleetwood mac,
##      flight of the conchords,
##      kyuss,
##      late of the pier,
##      led zeppelin,
##      mgmt,
##      michael jackson,
##      muse,
##      pink floyd,
##      rjd2,
##      röyksopp,
##      simian mobile disco,
##      snow patrol,
##      the cinematic orchestra,
##      the decemberists,
##      the flaming lips,
##      the prodigy,
##      the rolling stones,
##      tool,
##      tv on the radio}
```

```
## [4] {ac/dc,
##      bob marley & the wailers,
##      children of bodom,
##      dream theater,
##      iron maiden,
##      megadeth,
##      metallica,
##      nightwish,
##      sublime,
##      trivium,
##      volbeat}
## [5] {depeche mode,
##      dream theater,
##      faith no more,
##      green day,
##      iron maiden,
##      jay-z,
##      justin timberlake,
##      kanye west,
##      lily allen,
##      manu chao,
##      metallica,
##      muse,
##      pearl jam,
##      pink floyd,
##      queen,
##      sigur rós,
##      snow patrol,
##      stevie wonder,
##      tenacious d,
##      the streets,
##      thievery corporation,
##      type o negative,
##      u2}
```

En el siguiente plot podemos ver que los 20 artistas más escuchados por los usuarios que intervienen en el dataset original son:

```
itemFrequencyPlot(transacciones, topN=20, type="absolute")
```



Ya se tiene una lista de transacciones en base a los artistas. Ahora procedamos a aplicar el algoritmo de reglas de asociación. Para ello vamos a usar la función apriori de la librería **arules**

10% Se aplica el algoritmo de reglas de asociación 20% Se realizan diferentes pruebas variando algunos parámetros

Para iniciar este punto de la rúbrica es necesario recordar que:

> NOTA: Para considerar una regla interesante, debe ser lo suficientemente frecuente y fuerte (esto es, tener bastante soporte y confianza).

> NOTA: El soporte indica cuantas veces se han encontrado las reglas {lsh => rhs} en el dataset, cuanto más alto mejor. La confianza habla de la probabilidad de que {rhs} se de en función de {lhs}.

Vamos a pruebas realizar una primera prueba general

Realizamos la primera prueba para un valor de soporte de 0.01 y confianza de 0.5

```
# Obtenemos las reglas
reglas <- apriori(transacciones,  parameter = list(supp = 0.01, conf = 0.50))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      1
```

```
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 150
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1004 item(s), 15000 transaction(s)] done [0.07s].
## sorting and recoding items ... [655 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.02s].
## writing ... [50 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Ordenamos las reglas por su confianza
reglas <- sort(reglas, by="confidence", decreasing=TRUE)

# Vemos la estructura de las reglas
str(reglas)
```

```
## Formal class 'rules' [package "arules"] with 4 slots
##   ..@ lhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:85] 641 895 772 848 482 704 789 203 926 848 ...
##   .. .. .. .. ..@ p       : int [1:51] 0 2 4 5 7 9 11 13 15 17 ...
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 50
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ rhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:50] 203 704 203 203 704 704 848 684 203 848 ...
##   .. .. .. .. ..@ p       : int [1:51] 0 1 2 3 4 5 6 7 8 9 ...
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 50
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ quality:'data.frame':  50 obs. of  5 variables:
##   .. ..$ support   : num [1:50] 0.0111 0.0105 0.0223 0.0101 0.0109 ...
##   .. ..$ confidence: num [1:50] 0.663 0.643 0.637 0.634 0.628 ...
##   .. ..$ coverage  : num [1:50] 0.0168 0.0163 0.0349 0.0159 0.0174 ...
##   .. ..$ lift      : num [1:50] 4.18 3.57 4.02 4 3.49 ...
##   .. ..$ count     : int [1:50] 167 157 334 151 164 172 155 160 156 172 ...
```

```
##   ..@ info   :List of 4
##   .. ..$ data         : symbol transacciones
##   .. ..$ ntransactions: int 15000
##   .. ..$ support      : num 0.01
##   .. ..$ confidence   : num 0.5
```

```
# Desplegamos las reglas en la pantalla
inspect(reglas)
```

```
##      lhs                                    rhs             support
## [1]  {oasis,the killers}                 => {coldplay}     0.01113333
## [2]  {sigur rós,the beatles}             => {radiohead}    0.01046667
## [3]  {keane}                             => {coldplay}     0.02226667
## [4]  {radiohead,snow patrol}             => {coldplay}     0.01006667
## [5]  {coldplay,the smashing pumpkins}    => {radiohead}    0.01093333
## [6]  {the beatles,the smashing pumpkins} => {radiohead}    0.01146667
## [7]  {bob dylan,pink floyd}              => {the beatles}  0.01033333
## [8]  {led zeppelin,the doors}            => {pink floyd}   0.01066667
## [9]  {snow patrol,the killers}           => {coldplay}     0.01040000
## [10] {bob dylan,the rolling stones}      => {the beatles}  0.01146667
## [11] {beck,the beatles}                  => {radiohead}    0.01300000
## [12] {death cab for cutie,the killers}   => {coldplay}     0.01086667
## [13] {oasis,radiohead}                   => {coldplay}     0.01273333
## [14] {coldplay,sigur rós}                => {radiohead}    0.01206667
## [15] {the pussycat dolls}                => {rihanna}      0.01040000
## [16] {led zeppelin,the rolling stones}   => {the beatles}  0.01066667
## [17] {david bowie,pink floyd}            => {the beatles}  0.01006667
## [18] {bob dylan,radiohead}               => {the beatles}  0.01386667
## [19] {david bowie,the rolling stones}    => {the beatles}  0.01000000
## [20] {the beatles,the shins}             => {radiohead}    0.01066667
## [21] {t.i.}                              => {kanye west}   0.01040000
## [22] {radiohead,the rolling stones}      => {the beatles}  0.01060000
## [23] {travis}                            => {coldplay}     0.01373333
## [24] {the beatles,the strokes}           => {radiohead}    0.01046667
## [25] {broken social scene}               => {radiohead}    0.01506667
## [26] {pink floyd,the doors}              => {led zeppelin} 0.01066667
## [27] {the beatles,the killers}           => {coldplay}     0.01253333
## [28] {the kinks}                         => {the beatles}  0.01360000
## [29] {the flaming lips}                  => {radiohead}    0.01306667
## [30] {led zeppelin,radiohead}            => {the beatles}  0.01306667
## [31] {megadeth}                          => {metallica}    0.01626667
## [32] {snow patrol}                       => {coldplay}     0.02646667
## [33] {radiohead,the killers}             => {coldplay}     0.01506667
## [34] {simon & garfunkel}                 => {the beatles}  0.01540000
## [35] {bloc party,the killers}            => {coldplay}     0.01106667
## [36] {blur}                              => {radiohead}    0.01753333
## [37] {david bowie,radiohead}             => {the beatles}  0.01393333
## [38] {radiohead,u2}                      => {coldplay}     0.01140000
## [39] {oasis,the beatles}                 => {coldplay}     0.01060000
## [40] {the fray}                          => {coldplay}     0.01126667
## [41] {placebo,radiohead}                 => {muse}         0.01366667
## [42] {sonata arctica}                    => {nightwish}    0.01346667
## [43] {red hot chili peppers,the killers} => {coldplay}     0.01086667
## [44] {beck}                              => {radiohead}    0.02926667
```

```
## [45] {muse,the killers}                 => {coldplay}     0.01513333
## [46] {judas priest}                     => {iron maiden}  0.01353333
## [47] {muse,the beatles}                 => {radiohead}    0.01380000
## [48] {pink floyd,the doors}             => {the beatles}  0.01000000
## [49] {death cab for cutie,the shins}    => {radiohead}    0.01006667
## [50] {death cab for cutie,the beatles}  => {radiohead}    0.01246667
##      confidence coverage   lift      count
## [1]  0.6626984  0.01680000  4.180183 167
## [2]  0.6434426  0.01626667  3.569393 157
## [3]  0.6374046  0.03493333  4.020634 334
## [4]  0.6344538  0.01586667  4.002021 151
## [5]  0.6283525  0.01740000  3.485683 164
## [6]  0.6209386  0.01846667  3.444556 172
## [7]  0.6150794  0.01680000  3.458092 155
## [8]  0.5970149  0.01786667  5.689469 160
## [9]  0.5954198  0.01746667  3.755802 156
## [10] 0.5910653  0.01940000  3.323081 172
## [11] 0.5909091  0.02200000  3.277972 195
## [12] 0.5884477  0.01846667  3.711823 163
## [13] 0.5876923  0.02166667  3.707058 191
## [14] 0.5801282  0.02080000  3.218167 181
## [15] 0.5777778  0.01800000 13.415893 156
## [16] 0.5776173  0.01846667  3.247474 160
## [17] 0.5741445  0.01753333  3.227949 151
## [18] 0.5730028  0.02420000  3.221530 208
## [19] 0.5703422  0.01753333  3.206572 150
## [20] 0.5673759  0.01880000  3.147425 160
## [21] 0.5672727  0.01833333  8.854413 156
## [22] 0.5638298  0.01880000  3.169958 159
## [23] 0.5628415  0.02440000  3.550304 206
## [24] 0.5607143  0.01866667  3.110471 157
## [25] 0.5472155  0.02753333  3.035589 226
## [26] 0.5387205  0.01980000  6.802027 160
## [27] 0.5340909  0.02346667  3.368950 188
## [28] 0.5298701  0.02566667  2.979030 204
## [29] 0.5297297  0.02466667  2.938589 196
## [30] 0.5283019  0.02473333  2.970213 196
## [31] 0.5281385  0.03080000  4.743759 244
## [32] 0.5251323  0.05040000  3.312441 397
## [33] 0.5243619  0.02873333  3.307582 226
## [34] 0.5238095  0.02940000  2.944956 231
## [35] 0.5236593  0.02113333  3.303150 166
## [36] 0.5228628  0.03353333  2.900496 263
## [37] 0.5225000  0.02666667  2.937594 209
## [38] 0.5213415  0.02186667  3.288529 171
## [39] 0.5196078  0.02040000  3.277594 159
## [40] 0.5168196  0.02180000  3.260006 169
## [41] 0.5137845  0.02660000  4.504247 205
## [42] 0.5101010  0.02640000  8.236292 202
## [43] 0.5093750  0.02133333  3.213047 163
## [44] 0.5092807  0.05746667  2.825152 439
## [45] 0.5089686  0.02973333  3.210483 227
## [46] 0.5075000  0.02666667  8.562992 203
## [47] 0.5073529  0.02720000  2.814458 207
```

```
## [48] 0.5050505  0.01980000  2.839489 150
## [49] 0.5033333  0.02000000  2.792160 151
## [50] 0.5013405  0.02486667  2.781105 187
```

```
# Revisamos si existen reglas duplicadas
duplicated(reglas)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

```
# Verificamos si hay reglas redundantes
reglas_redundantes <- is.redundant(reglas)
```

Encontramos que para estos valores de confianza y soporte hay 50 reglas de asociación. No hay reglas duplicadas ni redundantes para estas reglas obtenidas.

Vamos a ordenarlas ahora por su valor de soporte, de mayor a menor, para conocer cuales reglas superan el valor de soporte mínimo (*min_sop*) y puedan considerarse **grandes grupos** y sean consideras para la siguiente prueba.

```
# Ordenamos las reglas por su valor de soporte
reglas <- sort(reglas, by="support", decreasing=TRUE)

# Vemos la estructura de las reglas
str(reglas)
```

```
## Formal class 'rules' [package "arules"] with 4 slots
##   ..@ lhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:85] 102 789 482 131 564 776 605 895 151 704 ...
##   .. .. .. .. ..@ p       : int [1:51] 0 1 2 3 4 5 6 8 9 11 ...
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 50
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ rhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:50] 704 203 203 704 567 848 203 704 203 848 ...
##   .. .. .. .. ..@ p       : int [1:51] 0 1 2 3 4 5 6 7 8 9 ...
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 50
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
```

```
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ quality:'data.frame':  50 obs. of  5 variables:
##   .. ..$ support   : num [1:50] 0.0293 0.0265 0.0223 0.0175 0.0163 ...
##   .. ..$ confidence: num [1:50] 0.509 0.525 0.637 0.523 0.528 ...
##   .. ..$ coverage  : num [1:50] 0.0575 0.0504 0.0349 0.0335 0.0308 ...
##   .. ..$ lift      : num [1:50] 2.83 3.31 4.02 2.9 4.74 ...
##   .. ..$ count     : int [1:50] 439 397 334 263 244 231 227 226 226 209 ...
##   ..@ info   :List of 4
##   .. ..$ data         : symbol transacciones
##   .. ..$ ntransactions: int 15000
##   .. ..$ support      : num 0.01
##   .. ..$ confidence   : num 0.5
```

```
# Desplegamos las reglas en la pantalla
inspect(reglas)
```

```
##      lhs                                    rhs             support
## [1]  {beck}                              => {radiohead}     0.02926667
## [2]  {snow patrol}                       => {coldplay}      0.02646667
## [3]  {keane}                             => {coldplay}      0.02226667
## [4]  {blur}                              => {radiohead}     0.01753333
## [5]  {megadeth}                          => {metallica}     0.01626667
## [6]  {simon & garfunkel}                 => {the beatles}   0.01540000
## [7]  {muse,the killers}                  => {coldplay}      0.01513333
## [8]  {broken social scene}               => {radiohead}     0.01506667
## [9]  {radiohead,the killers}             => {coldplay}      0.01506667
## [10] {david bowie,radiohead}             => {the beatles}   0.01393333
## [11] {bob dylan,radiohead}               => {the beatles}   0.01386667
## [12] {muse,the beatles}                  => {radiohead}     0.01380000
## [13] {travis}                            => {coldplay}      0.01373333
## [14] {placebo,radiohead}                 => {muse}          0.01366667
## [15] {the kinks}                         => {the beatles}   0.01360000
## [16] {judas priest}                      => {iron maiden}   0.01353333
## [17] {sonata arctica}                    => {nightwish}     0.01346667
## [18] {the flaming lips}                  => {radiohead}     0.01306667
## [19] {led zeppelin,radiohead}            => {the beatles}   0.01306667
## [20] {beck,the beatles}                  => {radiohead}     0.01300000
## [21] {oasis,radiohead}                   => {coldplay}      0.01273333
## [22] {the beatles,the killers}           => {coldplay}      0.01253333
## [23] {death cab for cutie,the beatles}   => {radiohead}     0.01246667
## [24] {coldplay,sigur rós}                => {radiohead}     0.01206667
## [25] {the beatles,the smashing pumpkins} => {radiohead}     0.01146667
## [26] {bob dylan,the rolling stones}      => {the beatles}   0.01146667
## [27] {radiohead,u2}                      => {coldplay}      0.01140000
## [28] {the fray}                          => {coldplay}      0.01126667
## [29] {oasis,the killers}                 => {coldplay}      0.01113333
## [30] {bloc party,the killers}            => {coldplay}      0.01106667
## [31] {coldplay,the smashing pumpkins}    => {radiohead}     0.01093333
## [32] {death cab for cutie,the killers}   => {coldplay}      0.01086667
## [33] {red hot chili peppers,the killers} => {coldplay}      0.01086667
## [34] {led zeppelin,the doors}            => {pink floyd}    0.01066667
## [35] {led zeppelin,the rolling stones}   => {the beatles}   0.01066667
## [36] {the beatles,the shins}             => {radiohead}     0.01066667
## [37] {pink floyd,the doors}              => {led zeppelin}  0.01066667
```

```
## [38] {radiohead,the rolling stones}      => {the beatles}  0.01060000
## [39] {oasis,the beatles}                 => {coldplay}     0.01060000
## [40] {sigur rós,the beatles}             => {radiohead}    0.01046667
## [41] {the beatles,the strokes}           => {radiohead}    0.01046667
## [42] {snow patrol,the killers}           => {coldplay}     0.01040000
## [43] {the pussycat dolls}                => {rihanna}      0.01040000
## [44] {t.i.}                              => {kanye west}   0.01040000
## [45] {bob dylan,pink floyd}              => {the beatles}  0.01033333
## [46] {radiohead,snow patrol}             => {coldplay}     0.01006667
## [47] {david bowie,pink floyd}            => {the beatles}  0.01006667
## [48] {death cab for cutie,the shins}     => {radiohead}    0.01006667
## [49] {david bowie,the rolling stones}    => {the beatles}  0.01000000
## [50] {pink floyd,the doors}              => {the beatles}  0.01000000
##      confidence coverage   lift      count
## [1]  0.5092807  0.05746667  2.825152 439
## [2]  0.5251323  0.05040000  3.312441 397
## [3]  0.6374046  0.03493333  4.020634 334
## [4]  0.5228628  0.03353333  2.900496 263
## [5]  0.5281385  0.03080000  4.743759 244
## [6]  0.5238095  0.02940000  2.944956 231
## [7]  0.5089686  0.02973333  3.210483 227
## [8]  0.5472155  0.02753333  3.035589 226
## [9]  0.5243619  0.02873333  3.307582 226
## [10] 0.5225000  0.02666667  2.937594 209
## [11] 0.5730028  0.02420000  3.221530 208
## [12] 0.5073529  0.02720000  2.814458 207
## [13] 0.5628415  0.02440000  3.550304 206
## [14] 0.5137845  0.02660000  4.504247 205
## [15] 0.5298701  0.02566667  2.979030 204
## [16] 0.5075000  0.02666667  8.562992 203
## [17] 0.5101010  0.02640000  8.236292 202
## [18] 0.5297297  0.02466667  2.938589 196
## [19] 0.5283019  0.02473333  2.970213 196
## [20] 0.5909091  0.02200000  3.277972 195
## [21] 0.5876923  0.02166667  3.707058 191
## [22] 0.5340909  0.02346667  3.368950 188
## [23] 0.5013405  0.02486667  2.781105 187
## [24] 0.5801282  0.02080000  3.218167 181
## [25] 0.6209386  0.01846667  3.444556 172
## [26] 0.5910653  0.01940000  3.323081 172
## [27] 0.5213415  0.02186667  3.288529 171
## [28] 0.5168196  0.02180000  3.260006 169
## [29] 0.6626984  0.01680000  4.180183 167
## [30] 0.5236593  0.02113333  3.303150 166
## [31] 0.6283525  0.01740000  3.485683 164
## [32] 0.5884477  0.01846667  3.711823 163
## [33] 0.5093750  0.02133333  3.213047 163
## [34] 0.5970149  0.01786667  5.689469 160
## [35] 0.5776173  0.01846667  3.247474 160
## [36] 0.5673759  0.01880000  3.147425 160
## [37] 0.5387205  0.01980000  6.802027 160
## [38] 0.5638298  0.01880000  3.169958 159
## [39] 0.5196078  0.02040000  3.277594 159
## [40] 0.6434426  0.01626667  3.569393 157
```

```
## [41] 0.5607143  0.01866667  3.110471 157
## [42] 0.5954198  0.01746667  3.755802 156
## [43] 0.5777778  0.01800000 13.415893 156
## [44] 0.5672727  0.01833333  8.854413 156
## [45] 0.6150794  0.01680000  3.458092 155
## [46] 0.6344538  0.01586667  4.002021 151
## [47] 0.5741445  0.01753333  3.227949 151
## [48] 0.5033333  0.02000000  2.792160 151
## [49] 0.5703422  0.01753333  3.206572 150
## [50] 0.5050505  0.01980000  2.839489 150
```

Al redondear los valores de confianza a 3 decimales, se obtiene que 39 reglas sobrepasan el valor de soporte mínimo (*min_sop*) de confianza.

Ahora vamos a realizar una segunda prueba, para un valor de soporte de 0.011 y confianza de 0.55

```
# Obtenemos las reglas
reglas <- apriori(transacciones,  parameter = list(supp = 0.011, conf = 0.55))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.55    0.1    1 none FALSE            TRUE       5   0.011      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 165
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1004 item(s), 15000 transaction(s)] done [0.07s].
## sorting and recoding items ... [595 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.02s].
## writing ... [9 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
# Ordenamos las reglas por su valor de soporte
reglas <- sort(reglas, by="support", decreasing=TRUE)

# Vemos la estructura de las reglas
str(reglas)
```

```
## Formal class 'rules' [package "arules"] with 4 slots
##   ..@ lhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:16] 482 133 704 969 102 848 641 704 203 772 ...
##   .. .. .. .. ..@ p       : int [1:10] 0 1 3 4 6 8 10 12 14 16
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 9
```

```
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ rhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
##   .. .. ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. .. .. ..@ i       : int [1:9] 203 848 203 704 203 704 848 704 203
##   .. .. .. .. ..@ p       : int [1:10] 0 1 2 3 4 5 6 7 8 9
##   .. .. .. .. ..@ Dim     : int [1:2] 1004 9
##   .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. .. ..$ : NULL
##   .. .. .. .. ..@ factors : list()
##   .. .. ..@ itemInfo   :'data.frame':    1004 obs. of  1 variable:
##   .. .. .. ..$ labels: chr [1:1004] "...and you will know us by the trail of dead" "[unknown]" "2pac
##   .. .. ..@ itemsetInfo:'data.frame':    0 obs. of  0 variables
##   ..@ quality:'data.frame':  9 obs. of  5 variables:
##   .. ..$ support   : num [1:9] 0.0223 0.0139 0.0137 0.013 0.0127 ...
##   .. ..$ confidence: num [1:9] 0.637 0.573 0.563 0.591 0.588 ...
##   .. ..$ coverage  : num [1:9] 0.0349 0.0242 0.0244 0.022 0.0217 ...
##   .. ..$ lift      : num [1:9] 4.02 3.22 3.55 3.28 3.71 ...
##   .. ..$ count     : int [1:9] 334 208 206 195 191 181 172 172 167
##   ..@ info   :List of 4
##   .. ..$ data         : symbol transacciones
##   .. ..$ ntransactions: int 15000
##   .. ..$ support      : num 0.011
##   .. ..$ confidence   : num 0.55
```

```
# Desplegamos las reglas en la pantalla
inspect(reglas)
```

```
##     lhs                                  rhs           support    confidence
## [1] {keane}                           => {coldplay}    0.02226667 0.6374046
## [2] {bob dylan,radiohead}             => {the beatles} 0.01386667 0.5730028
## [3] {travis}                          => {coldplay}    0.01373333 0.5628415
## [4] {beck,the beatles}                => {radiohead}   0.01300000 0.5909091
## [5] {oasis,radiohead}                 => {coldplay}    0.01273333 0.5876923
## [6] {coldplay,sigur rós}              => {radiohead}   0.01206667 0.5801282
## [7] {bob dylan,the rolling stones}    => {the beatles} 0.01146667 0.5910653
## [8] {the beatles,the smashing pumpkins} => {radiohead}   0.01146667 0.6209386
## [9] {oasis,the killers}               => {coldplay}    0.01113333 0.6626984
##     coverage   lift     count
## [1] 0.03493333 4.020634 334
## [2] 0.02420000 3.221530 208
## [3] 0.02440000 3.550304 206
## [4] 0.02200000 3.277972 195
## [5] 0.02166667 3.707058 191
## [6] 0.02080000 3.218167 181
## [7] 0.01940000 3.323081 172
## [8] 0.01846667 3.444556 172
## [9] 0.01680000 4.180183 167
```

35% Se explican las conclusiones que se obtienen

Finalmente, se obtienen 9 reglas, que nos permite concluir que:

- Entre los usuarios que escuchan a **keane** hay un 63% de probabilidad que también escuchen a **coldplay**. Su valor de soporte de 0.022 es el mayor valor entre las demás. Respecto a esta regla también podemos decir que **coldplay** está en el 3er lugar entre los artistas más escuchados.
- Entre los usuarios que escuchan a **bob dylan** y **radiohead** hay un 57% de probabilidad que también escuchen a **the beatles**. Esta regla aparece en el dataset en 208 ocasiones, con un valor de lift de 3.22.
- Si escuchas a **travis** hay un 56% de probabilidad que también escuches a **coldplay**.
- Si escuchas a **beck** y **the beatles** hay un 59% de probabilidad que también escuches a **radiohead**.
- Si escuchas a **oasis** y **radiohead** hay un 58% de probabilidad que también escuches a **coldplay**.
- Si escuchas a **coldplay** y **sigur rós** hay un 58% de probabilidad que también escuches a **radiohead**.
- Entre los usuarios que escuchan a **bob dylan** y **the rolling stones**, hay un 59% de probabilidad que también escuchen a **the beatles**.
- Entre los usuarios que escuchan a **the beatles** y **the smashing pumpkins**, hay un 62% de probabilidad que también escuchen a **radiohead**.
- Si escuchas a **oasis** y **the killers** hay un 66% de probabilidad que también escuches a **coldplay**.
- Al analizar las reglas 2 y 7, encontramos que entre los usuarios que escuchan a **bod dylan** también existe la probabilidad que escuchen a **the beatles**.
- También se puede decir que **coldplay** aparece como rhs (consecuente) en la mayoría de reglas, es decir, que si estás entre los usuarios que escuchan a keane, travis, oasis, radiohead o the killers es muy probable que también escuches a coldplay.
- Entre los usuarios que escuchan a **radiohead** hay una probabilidad que también escuchen a artistas como **beck**, **the beatles**, **coldplay**, **sigur rós**, **the smalling pumpkins.**

10% Se presenta el código y es fácilmente reproducible

Bibliografía

[1] Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Alojado en https://doi.org/10.1016/j.dib.2019.104344

[2] Clustering y heatmaps: aprendizaje no supervisado. Alojado en https://www.cienciadedatos.net/documentos/37_clustering_y_heatmaps#K-medoids_clustering_(PAM)

[3] Algoritmo Apriori, https://rociochavezml.com/algoritmo-apriori-en-r/, Rocio Chavez

---

Rúbrica

Ejercicio 1.1

- 15%. Se explican los campos de la base de datos, preparación y análisis de datos
- 10%. Se aplica el algoritmo de agrupamiento de forma correcta.
- 25%. Se prueban con diferentes valores de k.
- 10%. Se obtiene una medida de lo bueno que es el agrupamiento.
- 10%. Se ponen nombres a las asociaciones.
- 20%. Se describen e interpretan los diferentes clústers obtenidos.
- 10%. Se presenta el código y es fácilmente reproducible.

Ejercicio 1.2

- 25%. Se prueba un algoritmo diferente al kmeans.
- 25%. Se prueba otro algoritmo diferente al kmeans.
- 40%. Se comparan los resultados del kmeans y los otros dos métodos probados en este ejercicio.
- 10%. Se presenta el código y es fácilmente reproducible.

Ejercicio 2.1

- 10%. Se realiza un resumen de los datos incluidos en la base de datos.
- 15%. Se preparan los datos de forma correcta.
- 10%. Se aplica el algoritmo de reglas de asociación.
- 20%. Se realizan diferentes pruebas variando algunos parámetros.
- 35%. Se explican las conclusiones que se obtienen.
- 10%. Se presenta el código y es fácilmente reproducible.