CASO PRÁCTICO: Análisis de Cluster

Análisis de datos socioeconómicos

En primer lugar, cargamos los datos en R. Los datos están en el fichero "SocioeconomicDataset.csv". Se trata de datos socioeconómicos correspondientes a 91 países. Las variables medidas para cada país son seis:

- Birth.Rate: tasa de natalidad
- Mortality.Rate: tasa de mortalidad
- Infant.mortality.Rate: tasa de mortalidad infantil
- Life.expectency.man: esperanza de vida en hombres
- Life.expectency.woman: esperanza de vida en mujeres
- GNP: Producto Interior Bruto (PIB)

Cargamos el fichero en el área de trabajo de R. Para ello navegamos por el sistema de archivos para elegir el fichero de datos, y elegimos el fichero "SocioeconomicDataset.csv" en el directorio donde lo hayamos guardado previamente:

```
> datos=read.csv2(choose.files(),header=TRUE,row.names=1,dec=",")
> datos=as.matrix(datos)
```

Los datos quedan cargados en la variable datos:

```
> dim(datos)
[1] 91  6
```

Guardamos el número de países y el número de variables en n y p respectivamente:

```
> n = dim(datos)[1]
> p = dim(datos)[2]
```

Si queremos ver los nombres de las variables podemos utilizar la orden:

```
> colnames(datos)
```

Y para ver los nombres de los países la orden:

```
> rownames(datos)
```

De cara a escalar mejor los datos, tomamos logaritmos del producto interior bruto (variable GPB), y modificamos la columna correspondiente:

```
> datos[,6] <- log(datos[,6])
> colnames(datos)[6] <- "logGNP"
```

A continuación estandarizamos los datos:

```
> datos.st <- scale(datos)
```

Por tanto, ya tenemos los datos listos para trabajar con ellos.

Algoritmo de k-medias

En primer lugar utilizaremos la función **`kmeans`** para calcular clusters utilizando el método de las k-medias. Así, para calcular dos clusters, haríamos lo siguiente:

```
> clusters2.datos=kmeans(datos.st,centers=2,nstart=25)
```

De este modo, en clusters2.datos tendremos almacenado el modelo generado. El parámetro *nstart* sirve para indicar el número de inicializaciones aleatorias del método, y así proporcionar la mejor de las 25 soluciones obtenidas. Para ver a qué cluster pertenece cada país, utilizamos la orden:

```
> clusters2.datos$cluster
```

Para ver la variabilidad dentro de los grupos correspondiente a esos dos clusters, utilizamos la orden:

```
clusters2.datos$withinss
```

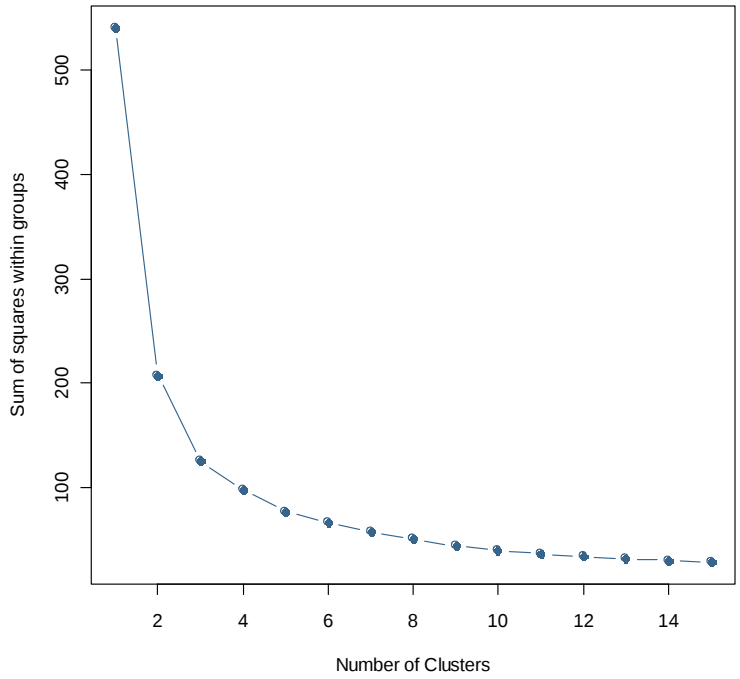
Para elegir el número de clusters óptimo, lo que haremos será calcular la variabilidad dentro de los grupos para distintas ejecuciones de la función **`kmeans`**. En concreto, ejecutamos la función **`kmeans`** para un número de entre 2 y 15 clusters, y elegimos el número de clusters que proporcione descenso en la variabilidad y, a la vez, un número de clusters no demasiado grande. Para ello generamos un vector, que denominaremos **`SSW`** con las sumas de las varianzas dentro de los grupos que se obtienen después de cada ejecución del método, y lo representamos gráficamente.

```
> #Inicializamos el vector
> SSW <- vector(mode = "numeric", length = 15)

> #Variabilidad de todos los datos, es decir, todos los datos como un único cluster
> SSW[1] <- (n - 1) * sum(apply(datos.st,2,var))

> #Variabilidad de cada modelo, desde 2 clusters hasta 15 clusters
> for (i in 2:15) SSW[i] <- sum(kmeans(datos.st,centers=i,nstart=25)$withinss)

> #Dibujamos un gráfico con el resultado
> plot(1:15, SSW, type="b", xlab="Number of Clusters", ylab="Sum of squares within
groups",pch=19, col="steelblue4")
```



En este caso elegimos 4 como número de clusters. En el gráfico, es el valor a partir del cuál el descenso en la variabilidad es más suave (es decir, menos acentuado). Probablemente los valores 5 ó 6 también sean adecuados, pero por simplicidad del modelo, tomamos el menor de los valores candidatos. En caso de duda, se podría repetir el análisis para estos valores, y elegir el modelo que permita una mejor explicación de los datos. Por tanto, calculamos el modelo para cuatro clusters:

```
> # k-means para 4 grupos y 25 arranques diferentes
> clusters4.datos <- kmeans(datos.st, 4, nstart = 25)
```

Para calcular los centroides de cada cluster, es decir, los prototipos de cada cluster, utilizamos la orden:

```
> centroides=aggregate(datos.st,by=list(clusters4.datos
$cluster),FUN=mean)
```

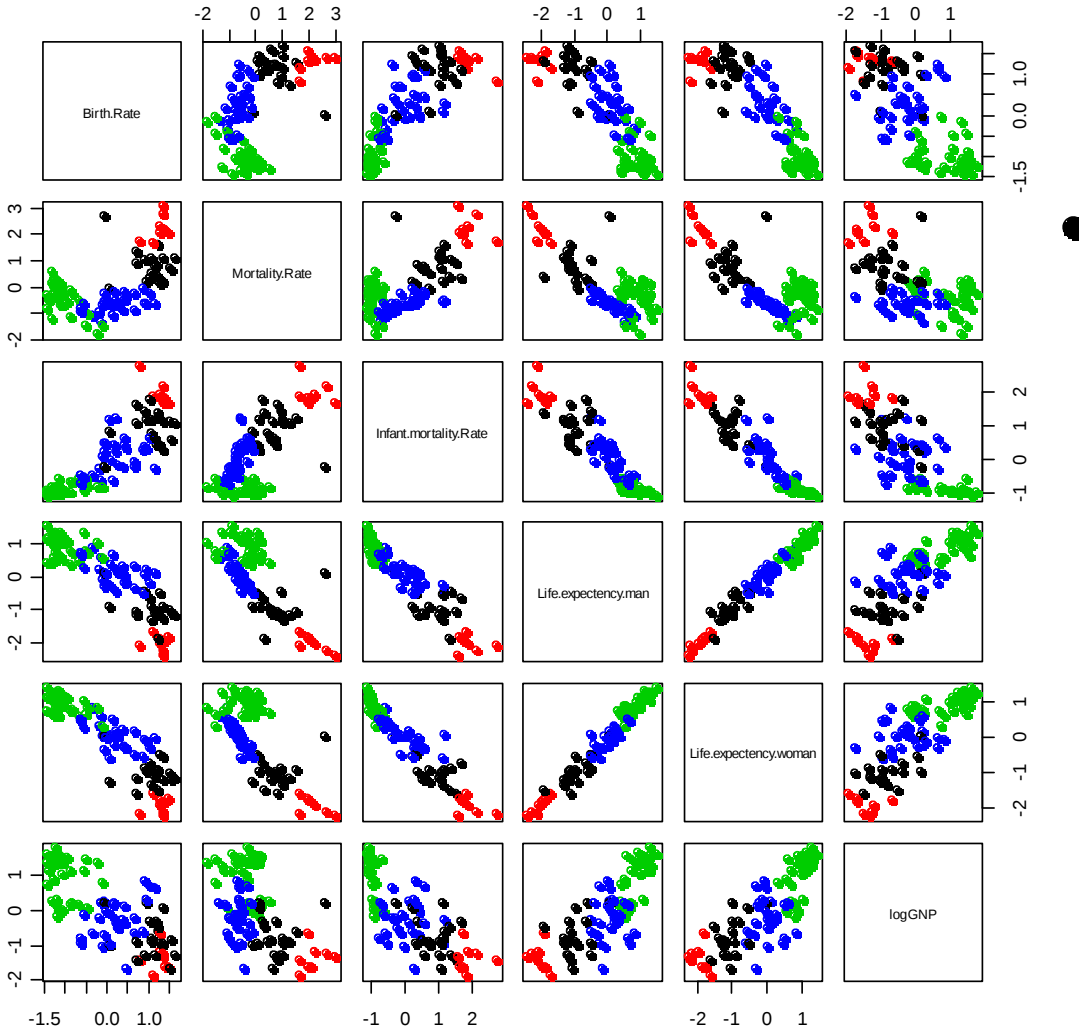
Para ver los 4 centroides:

```
> t(centroides)
                            [,1]      [,2]       [,3]        [,4]
Group.1                1.0000000  2.000000  3.0000000  4.00000000
Birth.Rate             1.0732598  1.273042 -1.0397580  0.24482196
Mortality.Rate         0.7883010  2.194461 -0.4046232 -0.64167769
Infant.mortality.Rate  0.9011199  1.925793 -0.9375144  0.04367497
Life.expectency.man   -0.9676394 -2.024511  0.8877078  0.09370553
Life.expectency.woman -1.0104112 -1.925379  0.9463145  0.01905436
logGNP                -0.7900894 -1.387819  0.9385329 -0.27403347
```

Veamos a continuación cómo representar los clusters. Una primera posibilidad es representarlos dibujando todas las variables dos a dos:

```
> # Dibujamos los clusters en el scatterplot (variables 2a2)
> nk=4 # nk es el numero de clusters
> pairs(datos.st,col= clusters4.datos$cluster,pch=19)
> points(clusters4.datos$centers, col = 1:nk, pch = 19, cex=2)
```

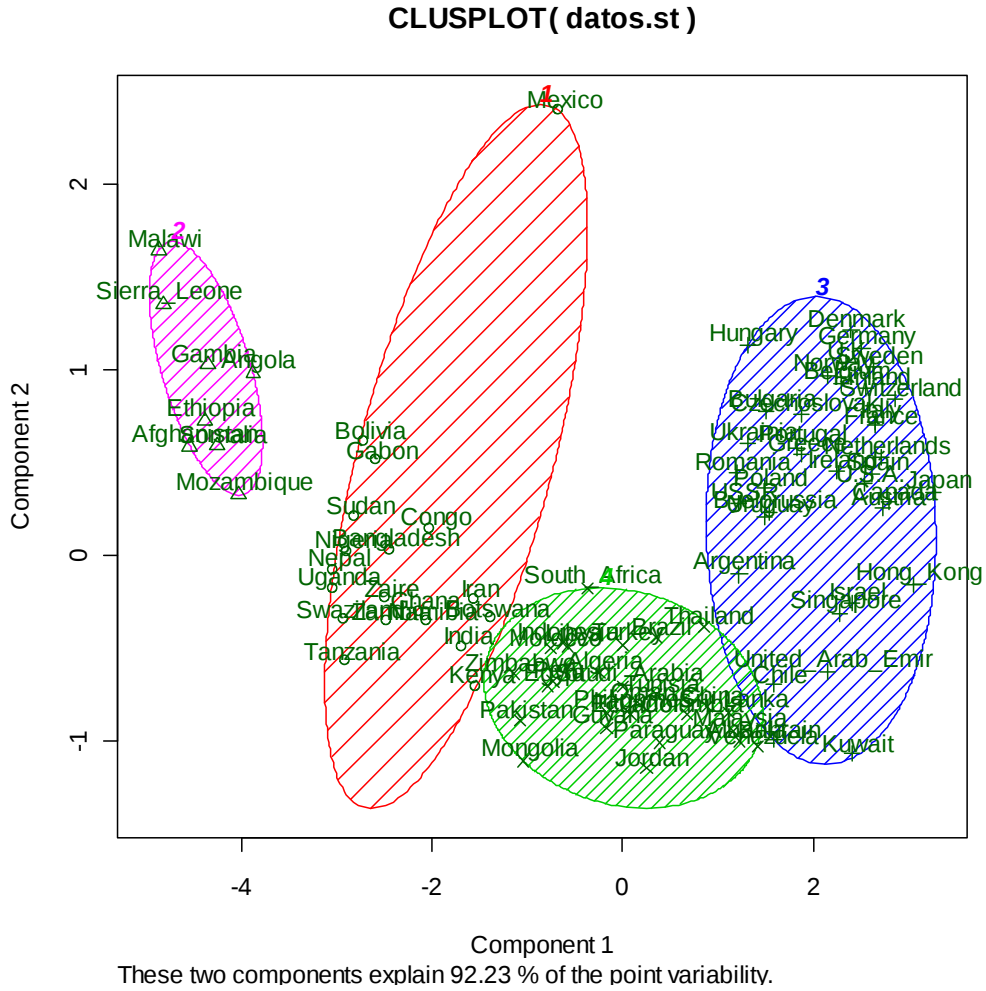
Se obtiene el siguiente gráfico, en el que cada color representa un cluster:



Una segunda posibilidad es reducir la dimensión de los datos utilizando el método de las componentes principales y pintar un gráfico con las dos primeras componentes. Para ello utilizaremos la librería **`cluster`**, que tiene implementada una orden para pintar este tipo de gráficos (la orden **`clusplot`**):

```
> # Guardamos el vector con el cluster correspondiente a cada país
> datos.clusters4 <- clusters4.datos$cluster
> # Vamos a hacer PCA para poder graficar los clusters en 2dimensiones!
> library(cluster)
> clusplot(datos.st, datos.clusters4, color=TRUE, shade=TRUE,
labels=2,lines=0)
```

Se obtiene el gráfico en dos dimensiones de los clusters, que es más explicativo que el gráfico anterior:



Métodos jerárquicos

Para calcular clusters jerárquicos, en primer lugar debemos obtener la matriz de distancias de los datos:

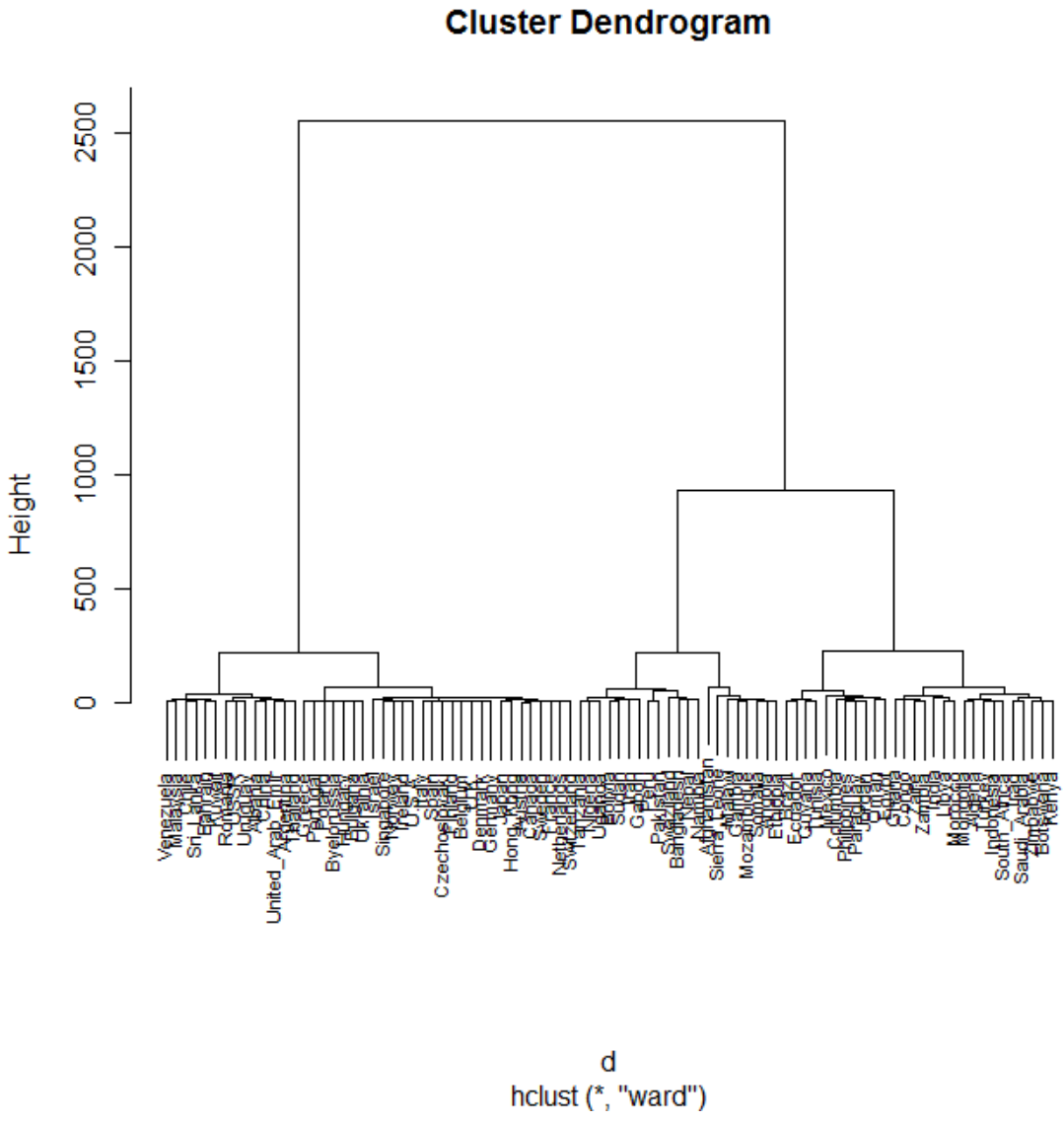
```
> # Primero obtenemos la matriz de distancias
> d <- dist(datos, method = "euclidean")
```

En el caso de los clusters jerárquicos, la función de R que los calcula es la función **`hclust`**, que utiliza como argumentos de entrada la matriz de distancias y el método de agrupación (a elegir entre “ward”, “single”, “complete”, “average”, “mcquitty”, “median” o “centroid”). En Internet se puede obtener información detallada sobre cada uno de estos métodos. Lo habitual es probar con algunos de ellos y elegir aquel que permita dar una explicación más clara de los datos. Utilizaremos el método “ward”, que optimiza el optimiza el ratio entre varianzas “dentro” y “entre” los grupos:

```
> fit <- hclust(d, method="ward")
```

Finalmente, dibujamos el dendrograma de los datos, que nos muestra el resultado y las etapas de agrupación:

```
> plot(fit, labels=rownames(datos),cex=0.7)  # Dendrograma
```

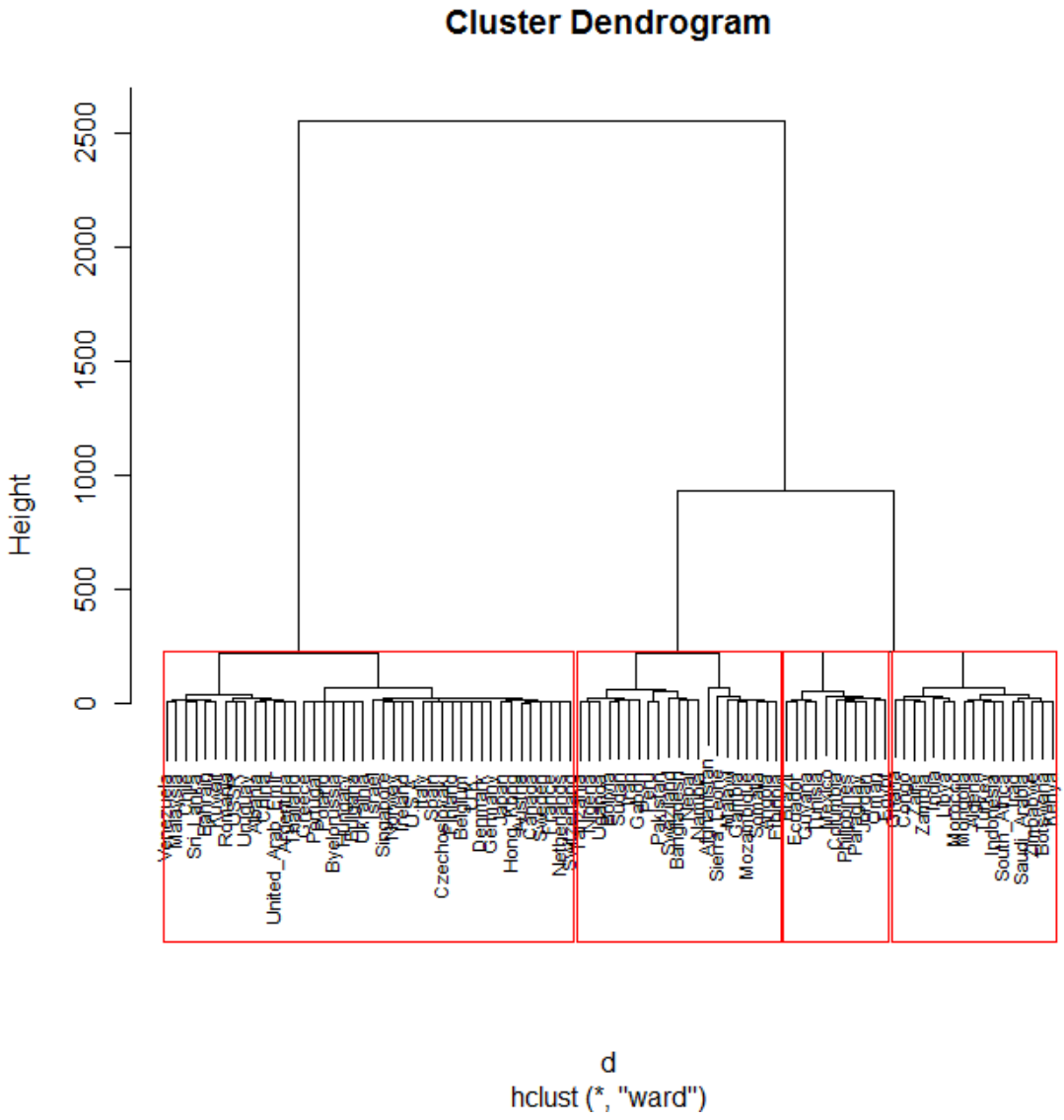


En la orden **`plot`**, el parámetro *labels* es un vector con las etiquetas de las hojas del árbol. El dendrograma nos permite decidir el número de clusters que queremos obtener. En este caso, parece razonable elegir 4, 5 ó 6 grupos. Si nos decidimos por cuatro grupos, podemos obtener el grupo al que pertenece cada país utilizando la orden **`cutree`**:

```
> groups <- cutree(fit, k=4) # Fijamos en 4 el número de clusters
```

Y si queremos recuadrar en el dendrograma cada uno de los clusters obtenidos, utilizamos la orden **`rect.hclust`**:

```
> rect.hclust(fit, k=4, border="red")
```



Cuestiones de evaluación

1. Utilizando el algoritmo de k-medias y los datos socioeconómicos del caso analizado, repetir el análisis en caso de que quisiéramos calcular 5 clusters.
2. Utilizando una muestra aleatoria de 50 datos de las flores iris (sin etiquetas), calcular clusters con los métodos jerárquicos "average" y "median" que se mencionan a lo largo del caso práctico. Utilizar para ello únicamente las variables numéricas de los datos, y etiquetar las hojas del dendrograma utilizando las especies de las flores.

   NOTA. Los datos de las flores iris están en la librería **`datasets`**. La librería está instalada por defecto en R, así que para cargarla se utiliza el comando library(datasets):

```
> library(datasets)   # Carga la librería datasets
```

Los datos se denominan **`iris`**. Estos datos se corresponden con 150 muestras de tres especies de flores de Iris (Iris setosa, Iris virginica e Iris versicolor). De cada muestra se han medido cuatro variables: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo. Los datos se encuentran cargados en la variable **`iris`**:

```
> dim(iris)
[1] 150   5

> iris[1:5,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
```

En primer lugar generamos los índices aleatorios de la muestra que utilizaremos:

```
> ind=sample(1:150, 50)
```

Para hacer el análisis cluster, utilizaremos las cuatro variables numéricas, y quitamos las etiquetas. Por tanto, usaremos la matriz de datos **`iris.cl`**, que contendrá las 50 muestras elegidas y las cuatros primeras columnas de la matriz **`iris`**:

```
> iris.cl=iris[ind,1:4]
```

Además, guardamos la especie de cada flor en la variable **`etiquetas`**:

```
> etiquetas=iris[ind,5]
```

Por ejemplo, para el método “ward”, las órdenes para pintar el dendrograma etiquetando sus hojas con las especies de las flores serían:

```
> d <- dist(iris.cl, method = "euclidean")
> fit <- hclust(d, method="ward")
> plot(fit, labels=etiquetas, cex=0.7)
> groups <- cutree(fit, k=3)
> rect.hclust(fit, k=3, border="red")
```

A continuación se muestra el dendrograma construido. El método detecta perfectamente la especie “setosa”, pero no es capaz de distinguir bien las especies “virginica” y “versicolor”, pues el cluster de la derecha mezcla ambas especies.