PRÁCTICA 6: ANÁLISIS CLUSTER.
ANÁLISIS ESTADÍSTICO MULTIVARIANTE.
GRADO EN CIENCIA E INGENIERÍA DE DATOS.

Sumario: En esta práctica mostramos cómo agrupar a los individuos de una población en K grupos según la similitud de sus valores en k variables aleatorias numéricas. A diferencia del Análisis Discriminante y la Regresión Logística, en este caso no existen grupos iniciales, por lo que este tipo de técnicas se denominan análisis no supervisado o aprendizaje automático. Los grupos dependerán de las distancias usadas para medir las similitudes y de las técnicas de agrupación aplicadas. El número de grupos final se podrá elegir de forma subjetiva o de forma objetiva siguiendo algún criterio predeterminado.

1. Estudio inicial de los datos

Comenzaremos leyendo el archivo de R denominado `USArrests` mediante

```
d<-USArrests
```

Con el comando `help(USArrests)` podemos ver que este archivo tiene tres variables numéricas que contienen los arrestos por cada 100000 habitantes por asesinatos (Murder), agresiones (Assults) y violaciones (Rape) en cada uno de los 50 estados de USA en el año 1973. También incluye el porcentaje de población urbana (UrbanPop) en cada estado. Haciendo `View(d)` o

```
head(d)
```

podemos ver los primeros datos de este archivo. Así observamos que en Alabama ese año hubo 13.2 arrestos por asesinato por cada 100000 habitantes, 236 por agresiones, y 21.2 por violaciones y que en ese estado hay un 58 % de población urbana.

Para calcular los principales estadísticos podemos hacer

```
summary(d)
cov(d)
colMeans(d)
```

De esta forma vemos que el vector de media muestral es $(7.788, 170.760, 65.540, 21.232)$ y que las cuasi-varianzas son $(18.970465, 6945.1657, 209.518776, 87.72916)$.

Claramente las variables tienen escalas muy diferentes (incluso una se mide en porcentajes) por lo que deberemos estandarizar los datos antes de aplicar un análisis cluster. Para confirmarlo podemos representar los datos con `plot(d)`, `boxplot(d)`, o

```
require(graphics)
pairs(d, panel = panel.smooth, main = 'USArrests',pch=20)
```

Así obtenemos las gráficas de la Figura 1 donde se observan las diferentes escalas de nuestras variables por lo que procedemos a estandarizarlas y representarlas con

```
ds <- as.data.frame(scale(d))
plot(ds,pch=20)
boxplot(ds)
```

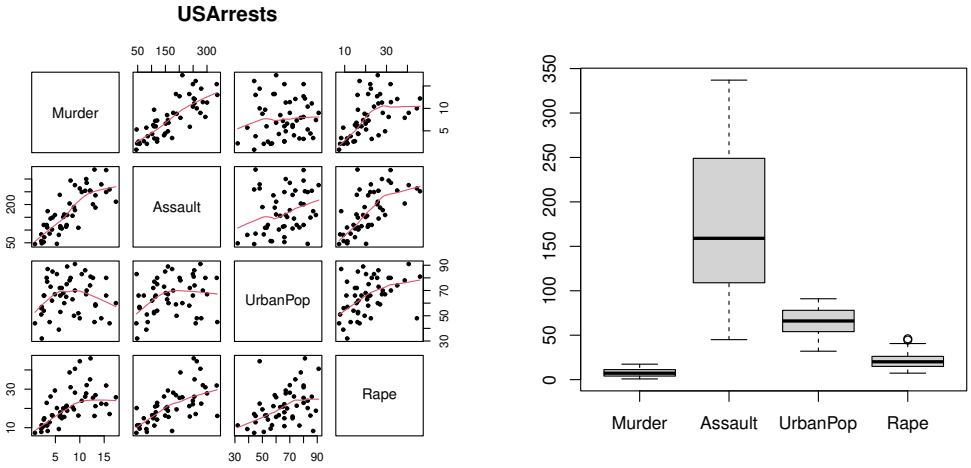

Figura 1: Gráficos de las variables de USArrest sin estandarizar.

En el gráfico caja-bigote de `Rape` se observan dos valores atípicos. Con `View(ds)` vemos que corresponden a los estados Nevada y Alaska donde hay muchos arrestos por violaciones. En los gráficos de puntos (*scatterplots*) no se observan la existencia de grupos.

Para resumir la información de las cuatro variables en un único gráfico podemos aplicar un PCA (ver práctica 4) con la matriz de correlaciones haciendo:

```
PCA<-princomp(d,cor=TRUE)
L<-PCA$loadings
S<-PCA$scores
Y1<-S[,1]
Y2<-S[,2]
biplot(PCA,pc.biplot=TRUE,cex=0.5)
biplot(PCA,pc.biplot=TRUE,xlabs=1:n,cex=0.5)
plot(Y1,Y2,pch=20)
text(Y1,Y2-0.2,labels=row.names(d),cex=0.5,col='red')
```

Así obtenemos las gráficas de la Figura 2 donde sí que parecen observarse la existencia de dos grupos de estados separados principalmente por el valor en Y_1 que representa a los estados con un mayor número de arrestos. El estado de Missouri quedaría en la frontera de separación entre esos dos grupos.

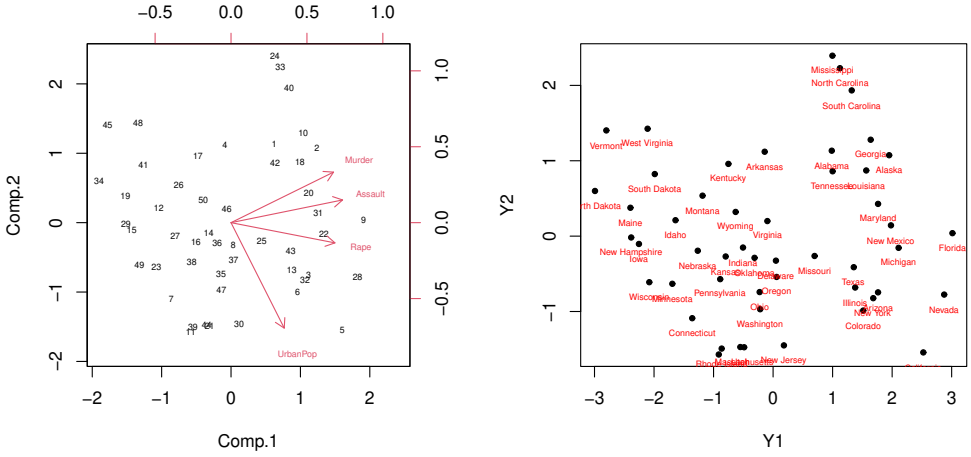

Figura 2: Gráficos PCA de las variables de USArrest estandarizadas (izquierda) y sin estandarizar (derecha).

2. Análisis cluster con K-means

Recordemos que en este procedimiento de análisis cluster (CA) debemos indicar el número de clusters K. Teniendo en cuenta el análisis previo tomaremos $K = 2$ para tratar de detectar a los estados más peligrosos (con más detenciones).

El algoritmo K-means se puede ejecutar de forma automática en R con el comando `kmean`. Por defecto, usa el algoritmo de Hartigan and Wong (1979). Con `help(kmean)` podemos ver los detalles y las distintas opciones. Para aplicarlo a los datos estandarizados de este ejemplo basta teclear:

```
CA1<-kmeans(ds,centers=2,nstar=10)
```

donde le hemos indicado que calcule dos grupos y que realice 10 inicializaciones al azar. Recordemos que el algoritmo comienza seleccionando K centroides al azar y agrupando los datos en el grupo del centroide más cercano. De esta forma, la solución final puede depender de esta selección al azar, por lo que es conveniente hacer varias inicializaciones distintas. El programa nos dará el mejor resultado de esas 10 opciones. Tecleando `CA1` podemos ver que se han formado dos grupos de tamaños 20 y 30 con medias (centroides):

Centroides	Murder	Assault	UrbanPop	Rape
C1	1.004934	1.0138274	0.19758532	0.8469650
C2	-0.669956	-0.6758849	-0.1317235	-0.5646433

Claramente los estados del primer grupo tienen un número de arrestos mayor que los del segundo grupo. Además también tienen más población urbana. Para guardar los centroides podemos hacer

```
C1<-CA1$centers[1,]
C2<-CA1$centers[2,]
```

Las clasificaciones para cada estado están en la ventana consola y en el objeto `CA1$cluster`. Para incluirla en la tabla de datos podemos hacer

```
Y<-CA1$cluster
d1<-data.frame(d,Y)
View(d1)
```

Para representar los grupos podemos hacer

```
plot(ds[,1:2],col=CA1$cluster,pch=20)
plot(ds[,3:4],col=CA1$cluster,pch=20)
plot(ds,col=CA1$cluster,pch=20)
plot(Y1,Y2,col=CA1$cluster,pch=20)
text(Y1,Y2-0.2,labels=row.names(d),cex=0.5)
```

De esta forma se obtienen las gráficas de las Figuras 3 y 4 donde podemos apreciar las variables estandarizadas por parejas o proyectadas en las componentes principales representando cada grupo con un color distinto (negro para el grupo 1 y rojo para el 2). En esas gráficas podemos observar que las variables representando arrestos tienen valores grandes en el grupo primero pero que la variable población urbana no cambia mucho en cada grupo. Sin embargo, la primera componente principal sí que sirve para discriminar a las observaciones de cada grupo.

Si queremos usar distintos símbolos para cada grupo haremos

```
plot(ds[,1:2],pch=as.integer(CA1$cluster),col=CA1$cluster)
legend('topleft',legend=c('cluster1','cluster2'),pch=1:2,cex=0.7)
```

Para incluir los centroides en esos gráficos podemos hacer

```
plot(ds[,1:2],col=CA1$cluster,pch=20)
points(C1[1],C1[2],col='green')
text(C1[1],C1[2]+0.2,'C1',col='green')
```

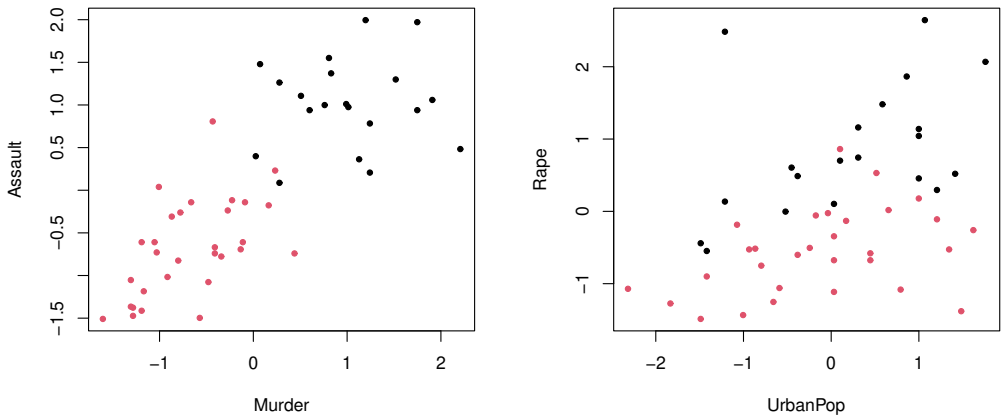

Figura 3: Gráficos de las variables de USArrest estandarizadas (izquierda) por grupos en Kmeans con $k = 2$.

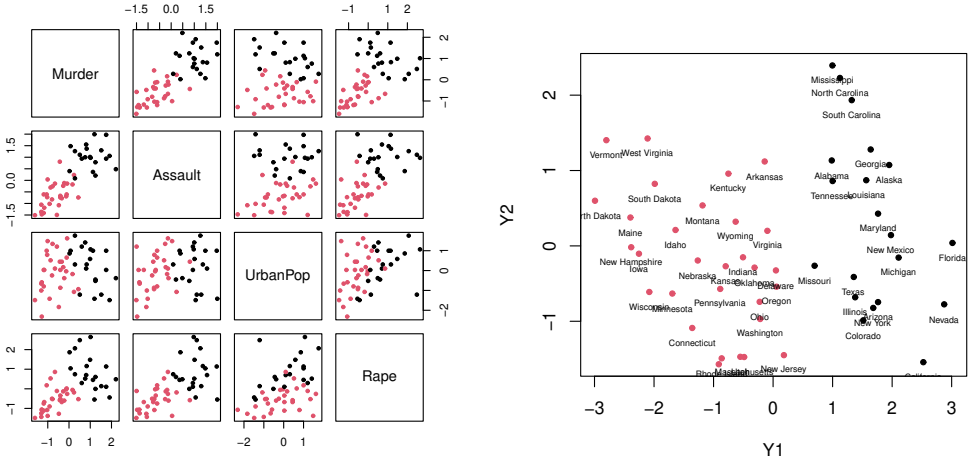

Figura 4: Gráficos por parejas (izq.) y PCA (der.) de las variables de USArrest por grupos en Kmeans con $k = 2$.

```
points(C2[1],C2[2],col='green')
text(C2[1],C2[2]+0.2,'C2',col='green')
plot(ds[,3:4],col=CA1$cluster,pch=20)
points(C1[3],C1[4],col='green')
text(C1[3],C1[4]+0.2,'C1',col='green')
points(C2[3],C2[4],col='green')
text(C2[3],C2[4]+0.2,'C2',col='green')
```

Otros paquetes de R permiten realizar gráficos más detallados. Por ejemplo con el paquete `cluster` podemos hacer este gráfico del CA con las componentes principales incorporadas de forma automática (ver Figura 5, izquierda):

```
library(cluster)
clusplot(ds,CA1$cluster,color=T,shade=T,labels=2,cex=0.5,lines=0)
```

Se puede hacer un gráfico similar con las funciones discriminantes (ver práctica 5) mediante (ver Figura 5, derecha)

```
library(fpc)
```

```
plotcluster(ds,CA1$cluster)
```

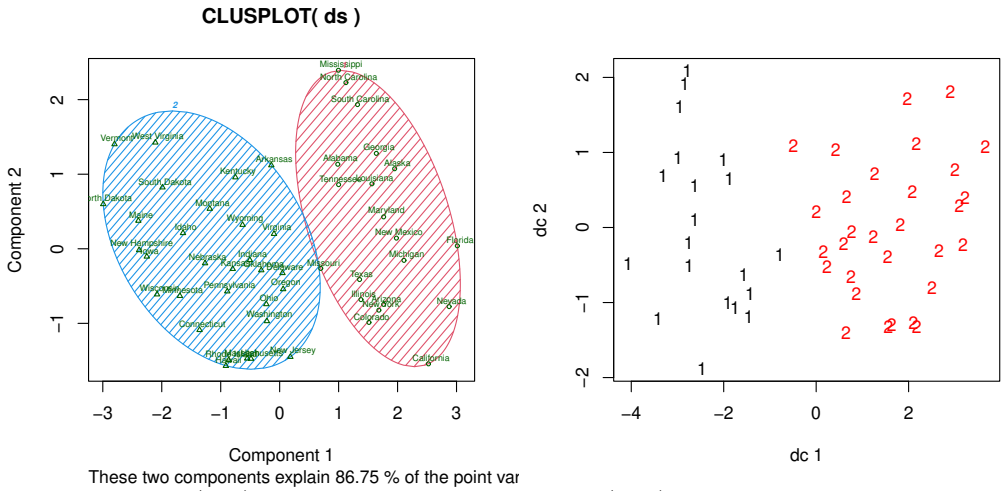

Figura 5: Gráficos PCA (izq.) y de funciones discriminates (der.) de las variables de USArrest por grupos en Kmeans con $k = 2$.

El comando `kmean` proporciona otros datos de interés. Por ejemplo, con

```
CA1$withinss
```

obtenemos las sumas de las distancias al cuadrado de todos los puntos a los centroides de cada grupo. En este caso se obtienen 46.74796 y 56.11445, es decir, el segundo grupo está un poco más disperso. Para comprobar el resultado podemos hacer:

```
dE2<-function(x,C) (sum((x-C)*(x-C)))
n<-50
dC1<-1:n
for (i in 1:n) dC1[i]<-dE2(ds[i,],C1)
sum(dC1*(Y==1))
```

Haciendo lo mismo con la distancias al otro centroide podemos ver que cada punto se ha incluido en el grupo con el centroide más cercano con:

```
d1<-data.frame(d1,dC1,dC2)
View(d1)
```

La suma de todas esas distancias a los centroides más cercanos las podemos calcular con

```
CA1$tot.withinss
```

obteniendo $102.8624 = 46.74796 + 56.11445$. El comando

```
CA1$totss
```

proporciona la suma de las distancias al cuadrado sin grupos (o con un único grupo) obteniéndose 196 por lo que al agruparlos se ha producido una disminución del

$$1 - \frac{102.8624}{196} = 0.4751918$$

por uno, es decir, con dos grupos la "variabilidad" se reduce un 47.51918 %.

Tarea: Haga un estudio similar para $k = 3$ y/o $k = 4$.

3. Análisis cluster jerarquizado

La principal ventaja de este método es que no tenemos que indicar el número de clusters (grupos) que queremos formar. El dendograma nos indicará los grupos que se van formando y nos permitirá decidir con cuántos nos quedamos.

Para realizar este agrupamiento de forma automática en R podemos usar el comando `hclust`. En primer lugar calcularemos las distancias entre las observaciones (estados en este caso) con

```
D <- dist(ds, method = 'euclidean')
```

De nuevo hemos usado los datos estandarizados y la distancia euclídea. Lógicamente, el resultado dependerá de la distancia usada (usar `help` para ver las distintas opciones). Las distancias están representadas en forma de vector. Para verlas en forma de matriz usaremos el comando:

```
M<-as.matrix(D)[1:50,1:50]
```

Las primeras distancias las vemos con `M[1:5,1:5]` y son:

Distancias	Alabama	Alaska	Arizona	Arkansas	California
Alabama	0.000000	2.703754	2.293520	1.289810	3.263110
Alaska	2.703754	0.000000	2.700643	2.826039	3.012541
Arizona	2.293520	2.700643	0.000000	2.717758	1.310484
Arkansas	1.289810	2.826039	2.717758	0.000000	3.763641
California	3.263110	3.012541	1.310484	3.763641	0.000000

Estas distancias se pueden representar con un "mapa de calor" mediante:

```
heatmap(M)
heatmap(M[1:5,1:5])
```

El resultado se puede ver en la Figura 6. Las distancias mayores se representan con colores oscuros. Los dendogramas nos indican las agrupaciones que se irían haciendo. Así, si solo consideramos los cinco primeros estados, los más similares son Arkansas y Alabama y serían los primeros en unirse en un grupo. Los siguientes son Arizona y California, etc.

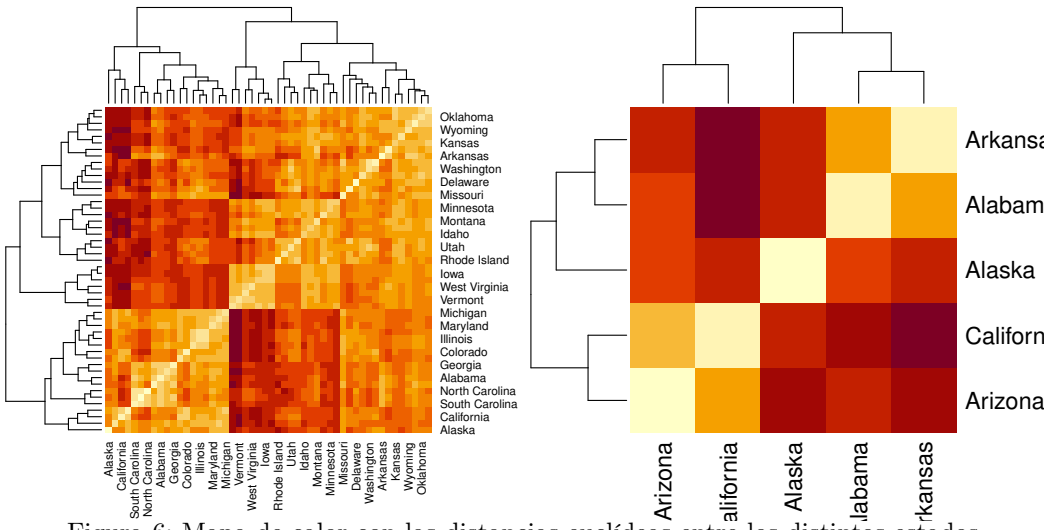

Figura 6: Mapa de calor con las distancias euclídeas entre los distintos estados.

Para hacer un CA jerarquizado en R basta teclear

```
CA2<- hclust(D,method='complete')
```

Para formar dos grupos podemos hacer:

```
grupos<- cutree(CA2, k=2)
sum(Y==grupos)
```

El segundo comando sirve para comprobar que todos los estados se clasifican igual que en el algoritmo Kmeans excepto Missouri (recuerde que este estado estaba muy cerca de la frontera). Si queremos cuatro grupos haremos

```
grupos<- cutree(CA2, k=4)
d2<-data.frame(ds,grupos)
```

Para representarlos gráficamente tenemos todas las opciones de la sección anterior. Por ejemplo podemos hacer:

```
plot(Y1,Y2,pch=as.integer(grupos),col=grupos,cex=0.7)
legend('topright',legend=c('Y=1','Y=2','Y=3','Y=4'),pch=1:4,cex=0.5)
text(Y1+0.15,Y2,1:n,cex=0.5)
```

De esta forma obtenemos los gráficos de la Figura 7 donde hemos representado las dos primeras componentes principales, los dos y cuatro grupos con colores y símbolos distintos y los estados por sus números de línea. Así comprobamos que el primer grupo contiene estados poco seguros y con poca población urbana, el segundo poco seguros y con mucha población urbana, el tercero estados con una seguridad media y con población urbana medio-alta y el cuarto contendría a los estados más seguros (con menos arrestos) y con población urbana media-baja.

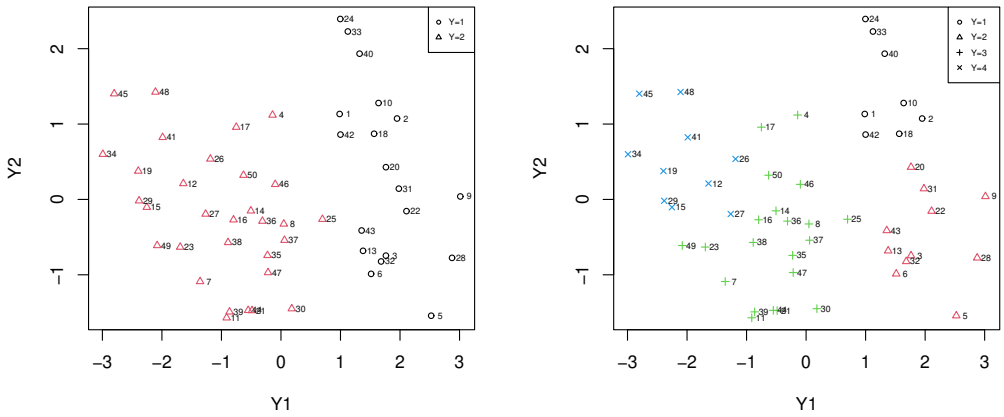

Figura 7: Gráfico PCA con dos (izquierda) o cuatro (derecha) grupos obtenidos por CA jerarquizado.

Para calcular los centroides de los cuatro grupos finales podemos hacer

```
colMeans(ds[grupos==1,])
colMeans(ds[grupos==2,])
colMeans(ds[grupos==3,])
colMeans(ds[grupos==4,])
```

Los resultados son

Centriod.	Murder	Assault	UrbanPop	Rape
C1	1.4463290	0.9838289	-0.8317925	0.3529110
C2	0.7499801	1.1199128	0.9361748	1.2156432
C3	-0.4400338	-0.4353831	0.3607592	-0.2830385
C4	-1.057970	-1.104663	-1.121953	-1.025155

Para hacer el dendograma que nos muestra cómo se han formado las distintas agrupaciones usaremos:

```
plot(CA2,cex=0.8,main='Dendograma',ylab='Distancia',xlab='Observaciones',sub='')
abline(h=4,col='red')
abline(h=5,col='green')
```

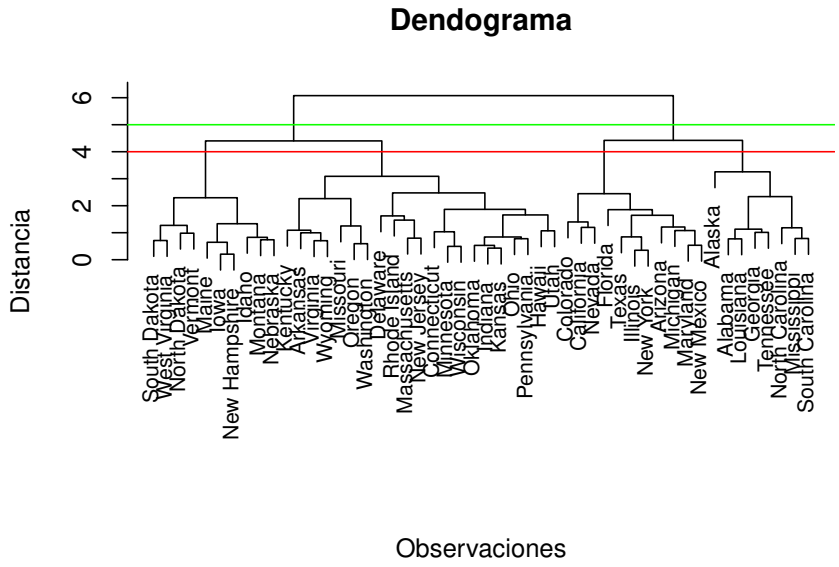

Figura 8: Gráfico dendograma para un CA jerarquizado con las divisiones para dos (verde) y cuatro (rojo) grupos.

En el gráfico observamos que los dos estados más similares (y que primero se unen) son Iowa y New Hampshire, los siguientes Illinois y New York, etc. Las líneas roja y verde nos muestran las divisiones para dos y cuatro grupos, respectivamente. En la parte derecha del dendograma se sitúan los grupos de estados con más arrestos.

4. Número de grupos

En primer lugar, debemos mencionar que no existe un número óptimo de grupos ya que esto depende de factores subjetivos. Para decidir sobre este punto, existen numerosos índices que, en algunos caso, nos pueden ayudar. Aunque no los vamos a ver en teoría, para calcularlos podemos hacer:

```
library(NbClust)
NbClust(ds,method='complete',index='all')$Best.nc
```

Así obtenemos el mensaje siguiente:

```
Among all indices:
* 9 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 6 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters
* 3 proposed 15 as the best number of clusters
***** Conclusion *****
* According to the majority rule, the best number of clusters is 2
```

Por lo tanto el número óptimo es $k = 2$ aunque hay seis métodos que recomiendan $k = 4$. El comando también proporciona dos métodos gráficos que pueden verse en la Figura 9. En ambos hay que buscar los

picos de crecimiento o decrecimiento (codo) y nos conducen a $k = 4$.

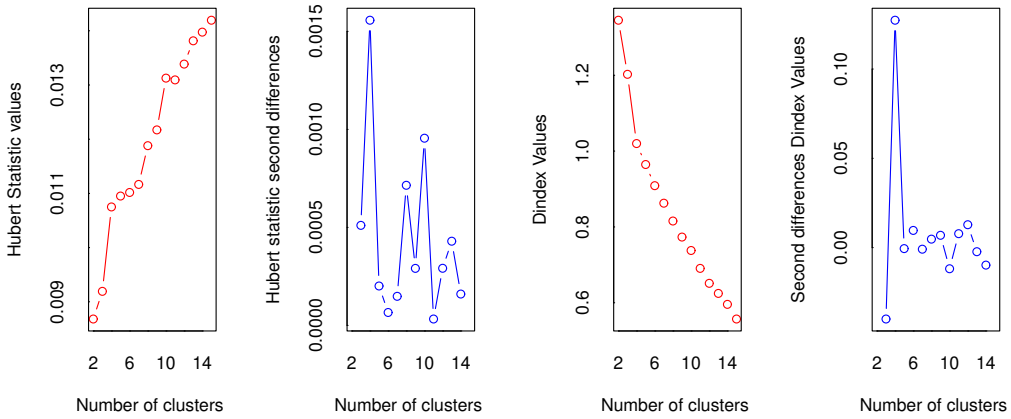

Figura 9: Gráficos para determinar el número óptimo de grupos.

Otra opción es usar el estadístico GAP que compara el total de las distancias intra-cluster para diferentes valores de k con la que se obtendría con datos al azar (sin grupos). Para calcularlo haremos

```
library(cluster)
gap<-clusGap(ds,FUNcluster=kmeans,K.max=15)
gap
```

El valor máximo del GAP es 0.3033210 y se obtiene con $k = 4$. Para representarlos podemos hacer

```
dev.off()
plot(gap$Tab[,3],type='b',xlab='k',ylab='GAP')
library(ggplot2)
library(factoextra)
fviz_gap_stat(gap)
```

obteniendo las gráficas de la Figura 10.

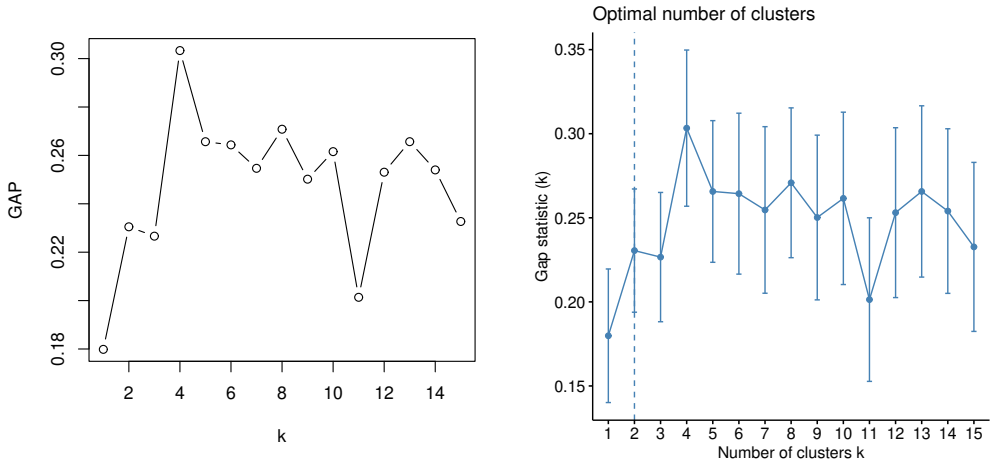

Figura 10: Gráficos GAP para determinar el número óptimo de grupos.

5. Ejercicios.

1. Aplicar un análisis cluster al fichero `iris`. Comprobar si los grupos obtenidos coinciden con los grupos establecidos por las distintas especies (usar `help` para ver las descripciones de las variables).

2. Aplicar un análisis cluster al fichero `heptathlon` clasificando a las atletas en dos grupos (usar `help` para ver las descripciones de las variables).

3. Aplicar un análisis cluster al fichero `decatlon.rda` (aula virtual) clasificando a los atletas en dos grupos.

4. Aplicar un análisis cluster al fichero `madres.rda` (aula virtual) clasificando a las observaciones en dos grupos.

5. **(Tarea)** Aplicar un análisis cluster a un conjunto de datos reales aplicando todas las técnicas que consideres oportunas.