# COMPLETAR Práctica 6: Análisis Cluster

Francisco Javier Mercader Martínez

**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 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. Necesitaremos los siguientes paquetes de R: **cluster** y **Nbclust**.

## 1. Estudio descriptivo inicial

Comenzaremos leyendo el archivo de R denominado *USArrests* mediante

```
d <- USArrests
```

Para tener información sobre el conjunto de datos podemos utilizar el comando **help(USArrests)** que nos describe el conjunto de datos que 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. Utilizando **View(d)** podemos visualizar el conjunto de datos completo y con la instrucción **head(d)** podemos ver los primeros datos de este conjunto de datos:

```
## completar aquí
```

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.

Podemos calcular las medias para cada variable:

```
## completar aquí
```

Los principales estadísticos:

```
## completar aquí
```

La matriz de varianzas y covarianzas:

```
## completar aquí
```

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

```
## completar aquí
```

O bien, utilizar gráficos de caja-bigotes:

```
## completar aquí
```

En las gráficas generadas se observan las diferentes escalas de nuestras variables por lo que procedemos a estandarizarlas

```
ds <- as.data.frame(scale(d))
```

y representarlas con gráficos de dispersión

```
## completar aquí
```

o con diagramas caja-bigotes:

```
## completar aquí
```

En el gráfico caja-bigote de *Rape* se observan dos valores atípicos. Para ver en qué estado se presentan los valores atípicos podemos utilizar **View(ds)** o bien

```
index = order(ds$Rape, decreasing = TRUE)[1:2]
ds[index, ]
```

```
##           Murder   Assault  UrbanPop     Rape
## Nevada 1.0129698 0.9748294  1.068066 2.644350
## Alaska 0.5078625 1.1068225 -1.211764 2.484203
```

mostrando que corresponden a los estados Nevada y Alaska donde hay muchos arrestos por violaciones.

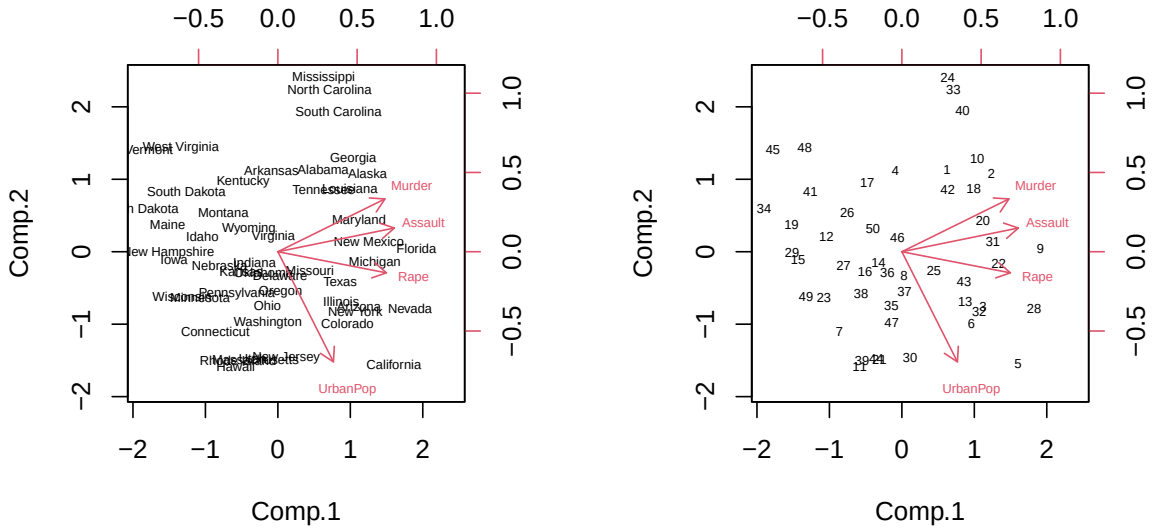
En los gráficos de puntos *scatterplots* no se observan la existencia de grupos.

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

```
PCA <- princomp(d, cor = TRUE)
L <- PCA$loadings
L
```

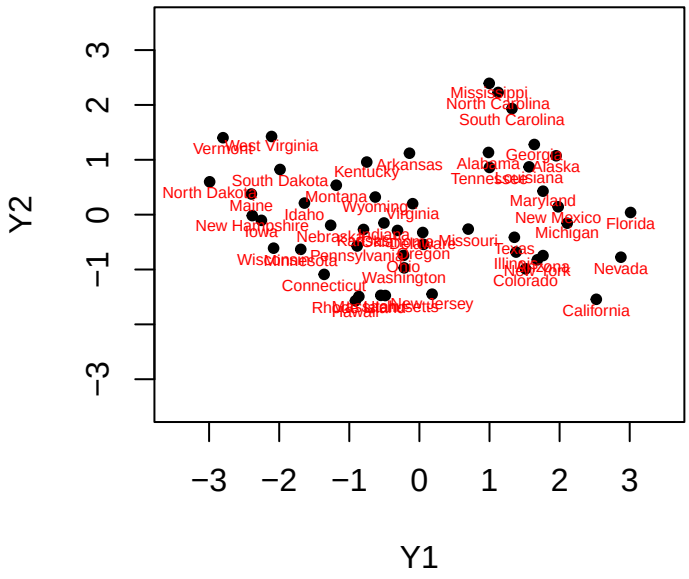
```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4
## Murder    0.536  0.418  0.341  0.649
## Assault   0.583  0.188  0.268 -0.743
## UrbanPop  0.278 -0.873  0.378  0.134
## Rape      0.543 -0.167 -0.818
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

```
S <- PCA$scores
Y1 <- S[,1]
Y2 <- S[,2]
n = dim(d)[1]
par(mfrow = c(1, 2))
biplot(PCA, pc.biplot = TRUE, cex = 0.5)
biplot(PCA, pc.biplot = TRUE, xlabs = 1:n, cex = 0.5)
```




Observamos que la primera componente principal se define fundamentalmente a partir de las variables *Assault*, *Murder* y *Rape* y con pesos asignados aproximadamente iguales, recogiendo esta componente la información relativa a los arrestos. En la segunda componente la variable *UrbanPoP* es la que tiene mayor peso y se puede interpretar como el nivel de urbanización del estado.

```
par(mfrow = c(1, 1))
plot(Y1, Y2, pch = 20, xlim = c(-3.5, 3.5), ylim = c(-3.5, 3.5))
text(Y1, Y2 -0.2, labels = row.names(d), cex = 0.5, col = "red")
```




En las gráficas que obtenemos 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.

## 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 **kmeans**. Por defecto, usa el algoritmo de Hartigan and Wong (1979). Con **help(kmeans)** podemos ver los detalles y las distintas opciones. Para aplicarlo a los datos estandarizados de este ejemplo utilizaremos la semilla **set.seed(123456)** y la instrucción:

```
set.seed(123456)
CA1 <- kmeans(ds, centers = 2, nstar = 10)
```

donde 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. En el objeto
**CA1** podemos ver que se han formado dos grupos de tamaños 20 y 30, y los centroides de cada grupo:

```
## completar aquí
```

Esto es, dos grupos 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. Con esta tabla, podemos decir que el centroide del primer grupo tiene mayor valor en población urbana que el centroide del segundo grupo. Pero esa diferencia entre los centroides podría no ser estadísticamente significativa y por tanto no reflejarse en las observaciones de cada grupo. Para guardar los centroides podemos hacer:

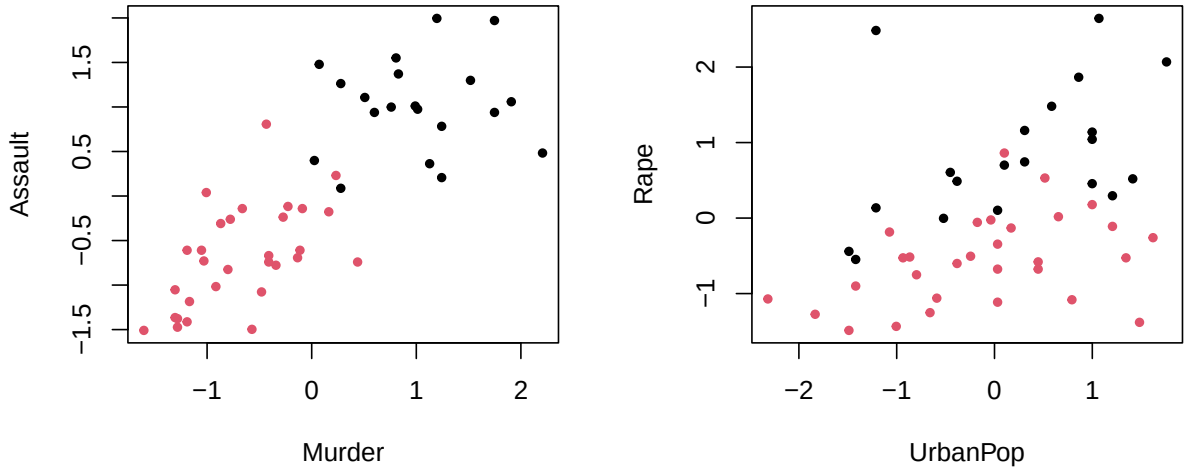
```
C1 <- CA1$centers[1, ]
C2 <- CA1$centers[2, ]
```

Las clasificaciones para cada estado están recogidos en el objeto **CA1$cluster**. Para incluirla en la tabla de datos podemos hacer

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

Y con **View(d1)** podremos visualizarla. Para representar los grupos podemos hacer *scatterplots* por parejas de variables:

```
par(mfrow = c(1, 2))
plot(ds[, 1:2], col = CA1$cluster, pch = 20)
plot(ds[, 3:4], col = CA1$cluster, pch = 20)
```

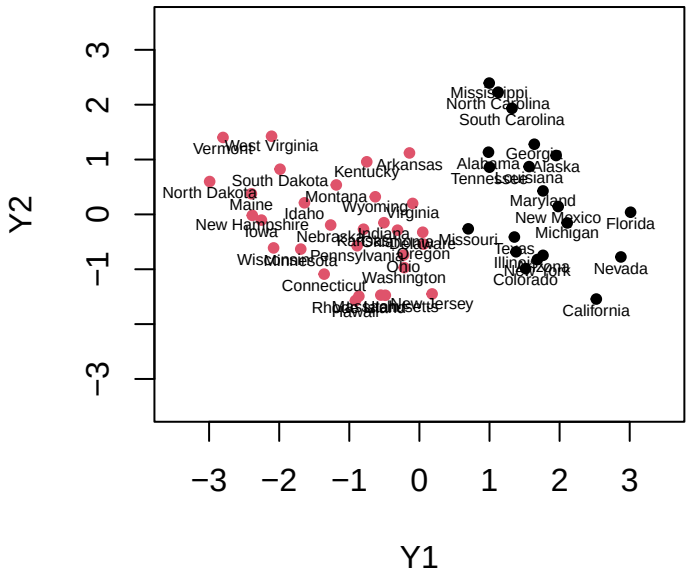



O bien, podemos representar las variables estandarizadas por parejas conjuntamente:

```
## completar aquí
```

También podríamos representar las variables estandarizadas proyectadas en las componentes principales identificando cada grupo con un color distinto (negro para el grupo 1 y rojo para el 2):

```
plot(Y1, Y2, col = CA1$cluster, pch = 20, xlim = c(-3.5, 3.5), ylim = c(-3.5, 3.5))
text(Y1, Y2-0.2, labels = row.names(d), cex = 0.5)
```

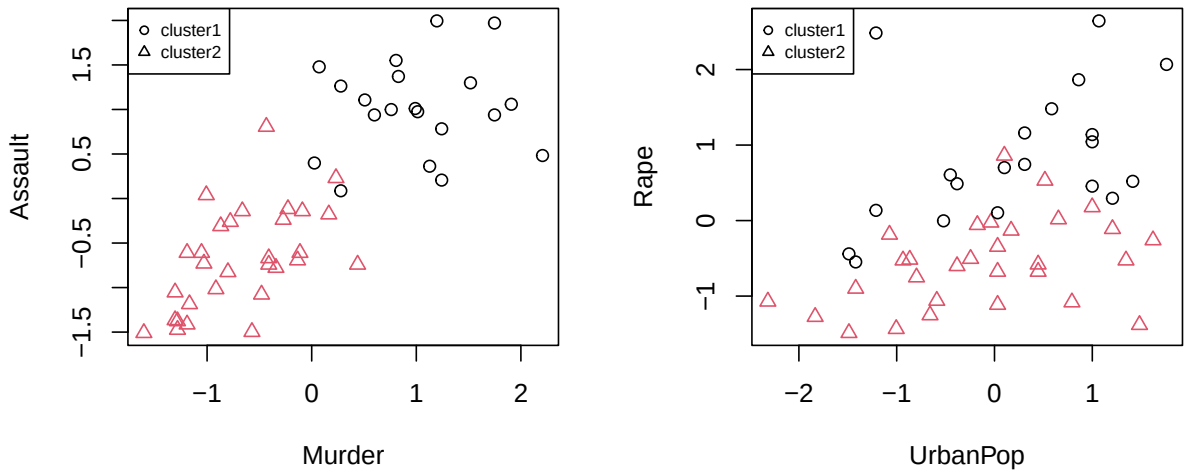



En estas gráficas podemos observar que las variables representando arrestos tienen valores grandes en el grupo

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.

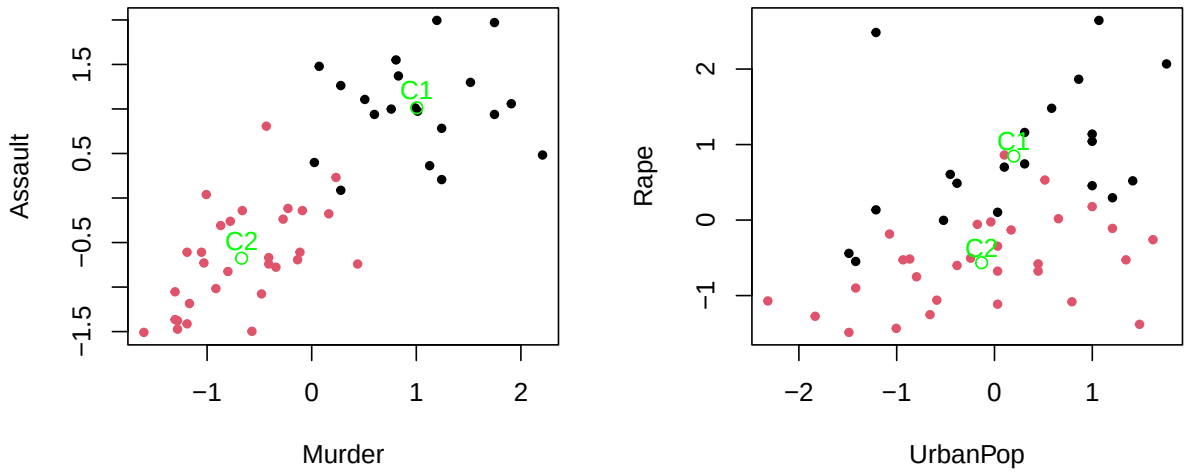
Si queremos usar distintos símbolos para cada grupo haremos

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

plot(ds[, 3:4], pch = as.integer(CA1$cluster), col = CA1$cluster)
legend("topleft", legend = c("cluster1", "cluster2"), pch = 1:2, cex = 0.7)
```



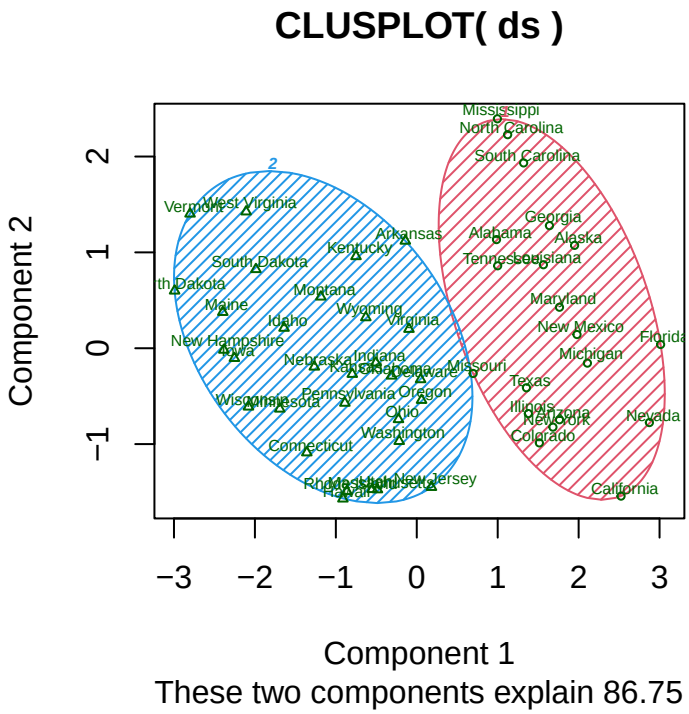

Para incluir los centroides en esos gráficos podemos hacer

```
par(mfrow = c(1, 2))
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")
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 análisis cluster con las componentes principales incorporadas de forma automática:

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




La función **kmeans** proporciona otros datos de interés. Por ejemplo, obtenemos las sumas de las distancias al cuadrado de todos los puntos a los centroides de cada grupo con

```
## completar aquí
```

En este caso se obtienen los valores 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))
```

```
## [1] 46.74796
```

Haciendo lo mismo con la distancias al otro centroide

```
dC2 <- 1:n
for (i in 1:n) dC2[i] <- dE2(ds[i,], C2)
sum(dC2*(Y==2))
```

```
## [1] 56.11445
```

podemos ver que cada punto se ha incluido en el grupo con el centroide más cercano con:

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

```
##            Murder Assault UrbanPop Rape Y      dC1       dC2
## Alabama      13.2     236       58 21.2 1 1.349203  6.252049
## Alaska       10.0     263       48 44.5 1 4.922542 15.027254
## Arizona       8.1     294       80 31.0 1 1.767868  9.055252
## Arkansas      8.8     190       50 19.5 2 3.890585  2.667666
## California    9.0     276       91 40.6 1 4.518302 15.162095
## Colorado      7.9     204       78 38.7 1 2.813250  8.527160
```

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

```
## completar aquí
```

obteniendo $102.8624 = 46.74796 + 56.11445$. El comando

```
CA1$totss
```

```
## [1] 196
```

proporciona la suma de las distancias al cuadrado sin grupos (o con un único grupo). En efecto, si realizamos

```
n <- 50
dC <- 1:n
C = colMeans(ds)
for (i in 1:n) dC[i] <- dE2(ds[i,], C)
sum(dC)
```

```
## [1] 196
```

obtenemos 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%.

Podemos hacer un estudio similar comparando la disminución de la variabilidad al pasar de $k = 2$ grupos a $k = 3$ grupos.

# 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 la función **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:

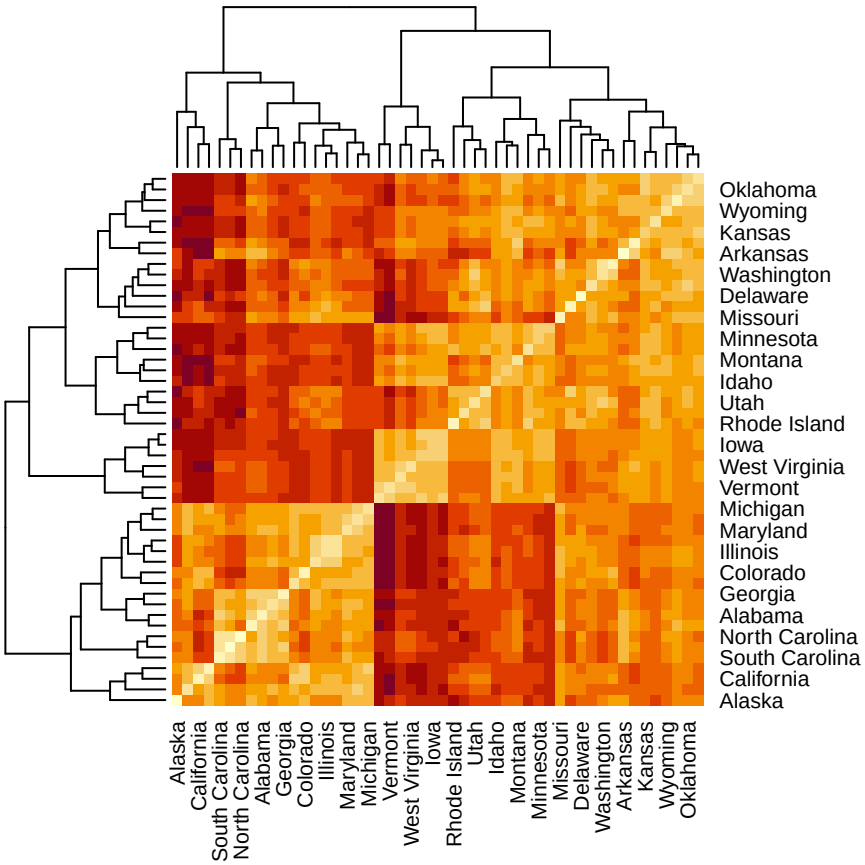
```
M <- as.matrix(D)[1:50, 1:50]
```

Las primeras distancias las vemos con

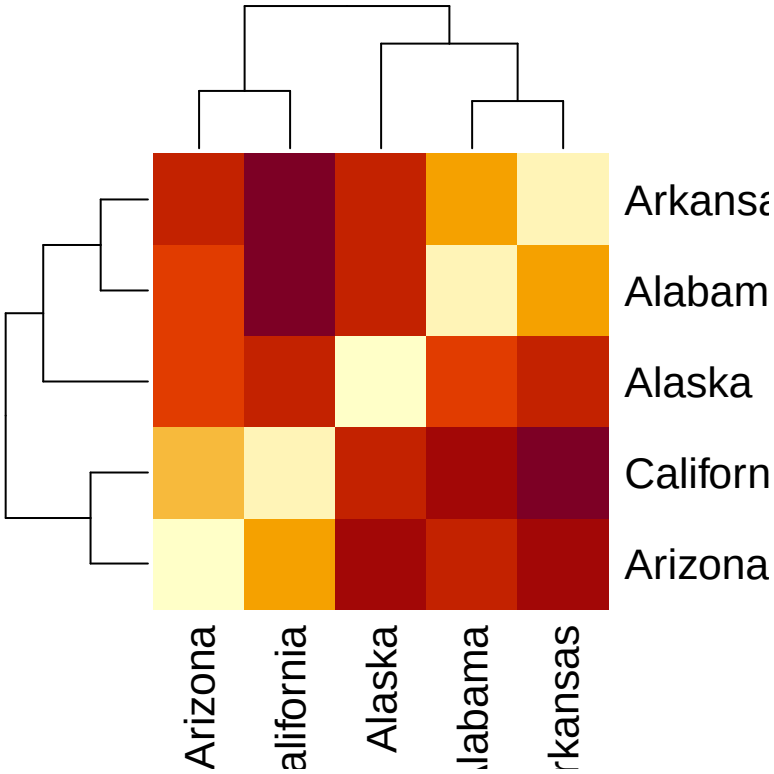
```
## completar aquí
```

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

```
heatmap(M)
```



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



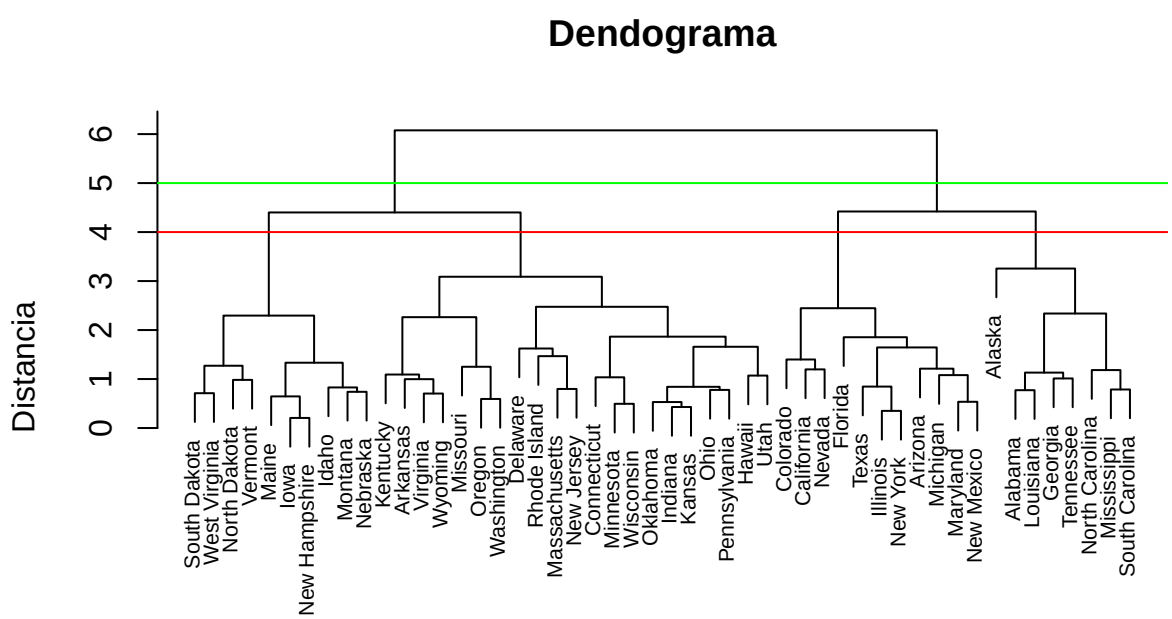

En este gráfico las distancias mayores se representan con colores oscuros. Los dendogramas nos indican las agrupaciones que se irían haciendo. Así, si solo consideramos los cinco primeros estados, los más similares son Arkansas y Alabama y serían los primeros en unirse en un grupo. Los siguientes son Arizona y California, etc.

Para hacer un CA jerarquizado en R incluiremos:

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

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

```r
plot(CA2, cex = 0.7, main = "Dendograma", ylab = "Distancia", xlab = "Observaciones")
abline(h = 4, col = "red")
abline(h = 5, col = "green")
```




En **CA2$merge** se describe cómo se ha realizado la unión de los grupos en el paso $i$ de la agrupación. Si un elemento $j$ en la fila es negativo, entonces la observación $-j$ se unió en esta etapa. Si j es positivo, entonces la unión se realizó con el grupo formado en la etapa (anterior) $j$ del algoritmo. Por lo tanto, las entradas negativas en la unió indican agrupaciones de elementos individuales, y las entradas positivas indican agrupaciones de elementos no individuales. Para ver cuáles son los estados que primero se unieron podemos hacer

```
## completar aquí
```

Oservamos que los dos estados más similares (y que primero se unen) son Iowa y New Hampshire, los siguientes Illinois y New York, etc. En el gráfico las líneas roja y verde nos muestran las divisiones para dos y cuatro grupos, respectivamente.

Para formar dos grupos podemos hacer:

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

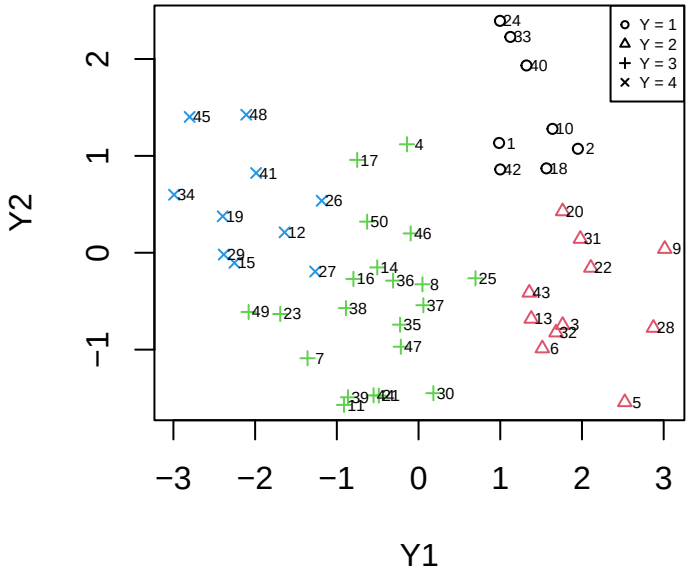
```
## [1] 49
```

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:

```r
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)
```




En estos gráficos 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.

Para calcular los centroides de los cuatro grupos finales podemos hacer

```r
## completar aquí
```

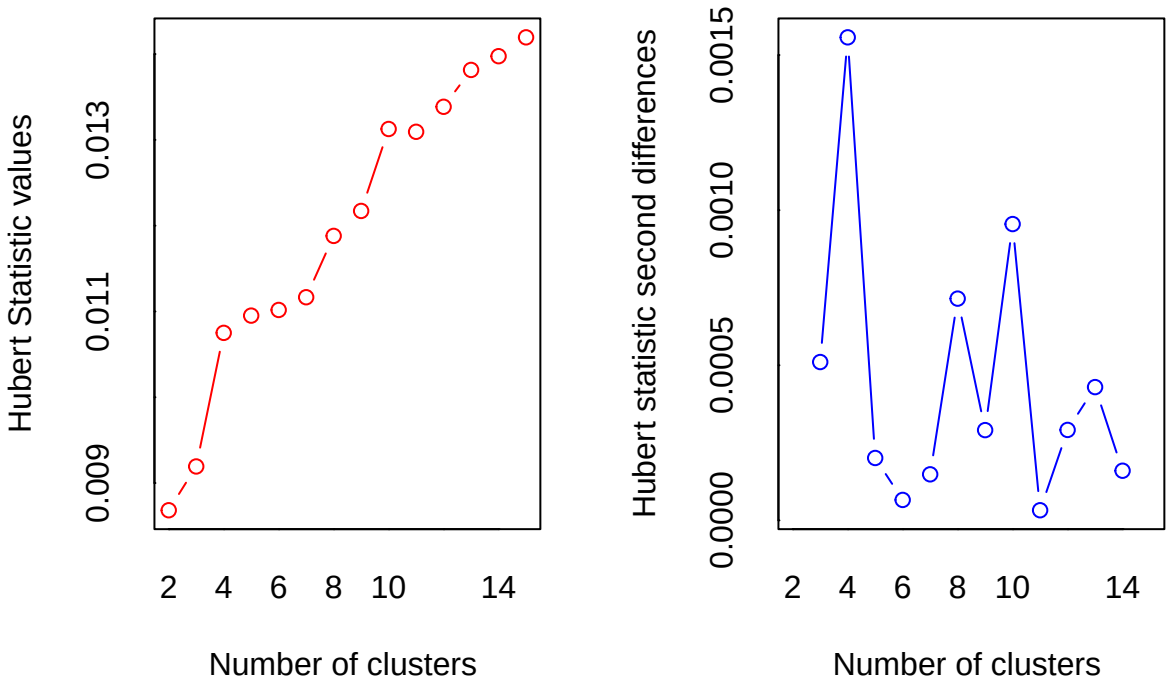
Los resultados son

| *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 |

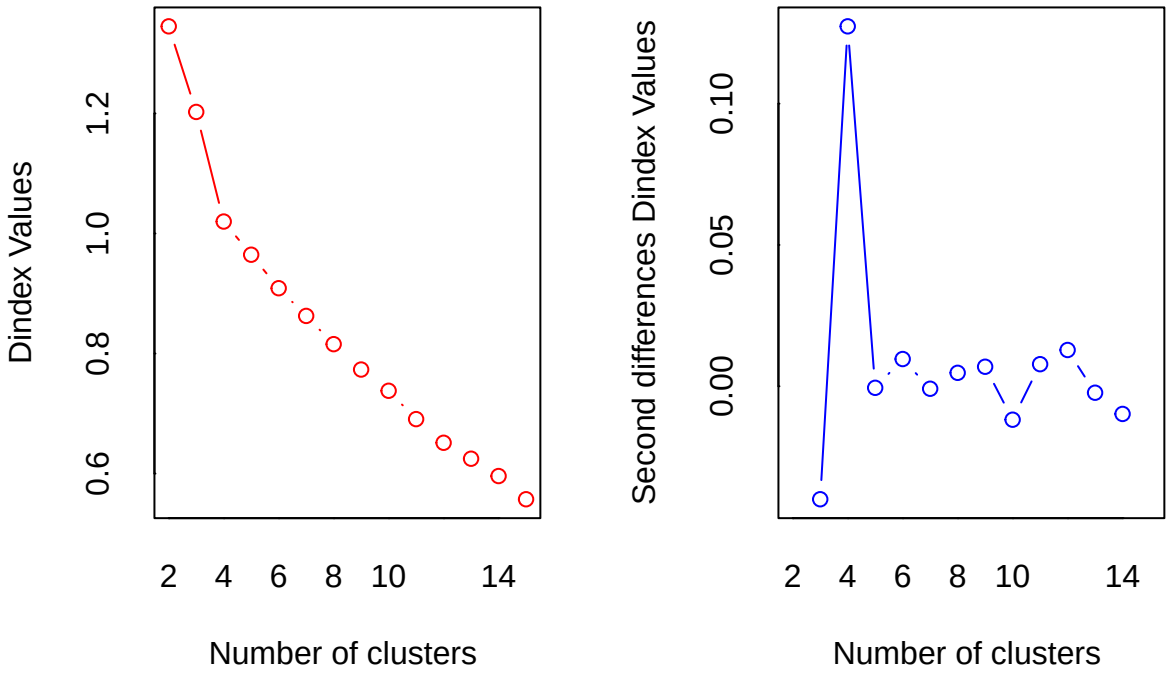
## 4. Número de grupos

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 casos, nos pueden ayudar. Aunque no lo hemos visto en teoría, para calcularlos podemos hacer:

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




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * 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
##
##
## *******************************************************************

##                        KL      CH Hartigan     CCC  Scott  Marriot   TrCovW
## Number_clusters   4.0000  2.0000   4.0000  2.0000  4.000      4.0   3.0000
## Value_Index     393.5832 41.8949  13.4849 -0.3483 48.647 261453.4 376.9233
##                  TraceW Friedman   Rubin Cindex     DB Silhouette   Duda
## Number_clusters  4.0000  15.0000  4.0000   3.00 5.0000     2.0000 2.0000
## Value_Index     17.8644   7.3522 -0.5494   0.38 0.8645     0.4048 0.5525
##                 PseudoT2  Beale Ratkowsky    Ball PtBiserial Frey McClain
```

```
## Number_clusters   2.0000 2.0000    2.0000  3.0000      2.000    1  2.0000
## Value_Index      13.7698 1.8468    0.4404 24.1946      0.633   NA  0.5287
##                     Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 15.0000      0  5.0000      0 15.0000
## Value_Index      0.3253      0  1.4783      0  0.0907
```

Por lo tanto el número óptimo es $k = 2$ aunque hay seis métodos que recomiendan $k = 4$. El comando también proporciona dos métodos gráficos. En ambos hay que buscar los picos de crecimiento o decrecimiento (codo) y nos conducen a $k = 4$.