Análisis cluster - Práctica 3

José Luis Romero Béjar - Guillermo Arturo Cañadas De la Fuente

Junio, 2024

En esta práctica se ilustran dos ejemplos de análisis cluster mediante un **método jerárquico (Ward)** y mediante un método **no jerárquico (K-means)**.

Para realizar esta práctica se deben descargar los siguientes archivos, disponibles en la plataforma PRADO del curso:

- *AC_3_esp.Rmd*

Aspectos tratados:

- Paquetes de R necesarios.
- Preparación de los datos.
- Algunas distancias para análisis cluster.
- Algoritmo de agrupamiento jerárquico.
- Algoritmo de agrupamiento no jerárquico.

Este material es una adaptación de un curso de R de Bradley Boehmke, en concreto, la parte que hace referencia a análisis cluster.

Cargando paquetes y el conjunto de datos

Cargar e instalar paquetes de R

El siguiente módulo de código fuente se encarga de cargar, si ya están instalados, todos los paquetes que se utilizarán en esta sesión de R. Si bien un paquete R se puede cargar en cualquier momento cuando se vaya a utilizar, es recomendable optimizar sus llamadas con este fragmento de código al principio.

Cargar un paquete en una sesión de R **requiere que ya esté instalado**. Si no es así, el primer paso es ejecutar la sentencia:

install.packages("name_of_the_library")

```
########################################
# Loading necessary packages and reason #
########################################

# This is an example of the first installation of a package
# Only runs once if the package is not installed
# Once it is installed this sentence has to be commented (not to run again)
# install.packages("factoextra")

# Package required to call 'mutate' function
library(tidyverse)
```

```r
# Package required to call 'clusGap' function
library(cluster)

# Package required to call 'get_dist', 'fviz_cluster' and 'fviz_dist' functions
library(factoextra)

# Package required to call 'ggdendrogram' function
library(ggdendro)

# Package required to call 'grid.arrange' function
library(gridExtra)
```

Preparación de los datos

Para realizar un análisis cluster con R los datos deben ser preparados como sigue:

- Hay que asegurarse de que las **filas** son **registros de observaciones** y que las **columnas** son las **variables** de interés (estructura tipo data.frame).
- Hay que hacer un tratamiento adecuado de los **valores perdidos** (eliminar o sustituir su valor).
- Se debe decidir si **trabajar con datos estandarizados** para hacer comparables variables medidas en distintas escalas.

Para esta práctica se trabajará con el conjunto de datos *USArrests* del repositorio base de R. Este conjunto de datos contiene estadísticas sobre arrestos por cada 100.000 habitantes según el delito: atraco, asesinato y violación en cada uno de los 50 estados de USA en el año 1973. También incluye el porcentaje de población residente en áreas urbanas. Para este ejemplo se decide **omitir todos los valores perdidos** y **se estandarizan los datos**.

```r
# Data loading
df<-USArrests
# Visualization of a few records
head(df)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
# Decide to delete not available data
df<-na.omit(df)

# To prevent the cluster analysis from being influenced by any arbitrary variable, the data are standar
df<-as.data.frame(scale(df))

# Visualization of standardized data
head(df)
```

```
##                Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
```

```
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

Análisis cluster

Algunas distancias para análisis cluster

Para clasificar las observaciones en grupos es necesario elegir medidas de **similaridad**, o de **distancia** (disimilaridad), adecuadas que proporcionen información de como de parecidas son dos observaciones cualesquiera. De hecho esta elección influye en el tamaño y forma de los clusters. Esta elección es un **paso fundamental** en clustering.

Algunas **medidas de distancia clásicas** frecuentemente utilizadas son la distancia **Euclídea** o la distancia **Manhattan**.

Otras medidas de disimilaridad ampliamente utilizadas, por ejemplo, en el análisis de datos de expresión génica son las **distancias basadas en correlaciones**. Estas distancias se obtienen restando la correspondiente medida de correlación al valor 1. Entre estas medidas de distancia destacan: la distancia basada en la correlación de **Pearson**, en la correlación de **Spearman**, correlación de **Kendall**, etc.

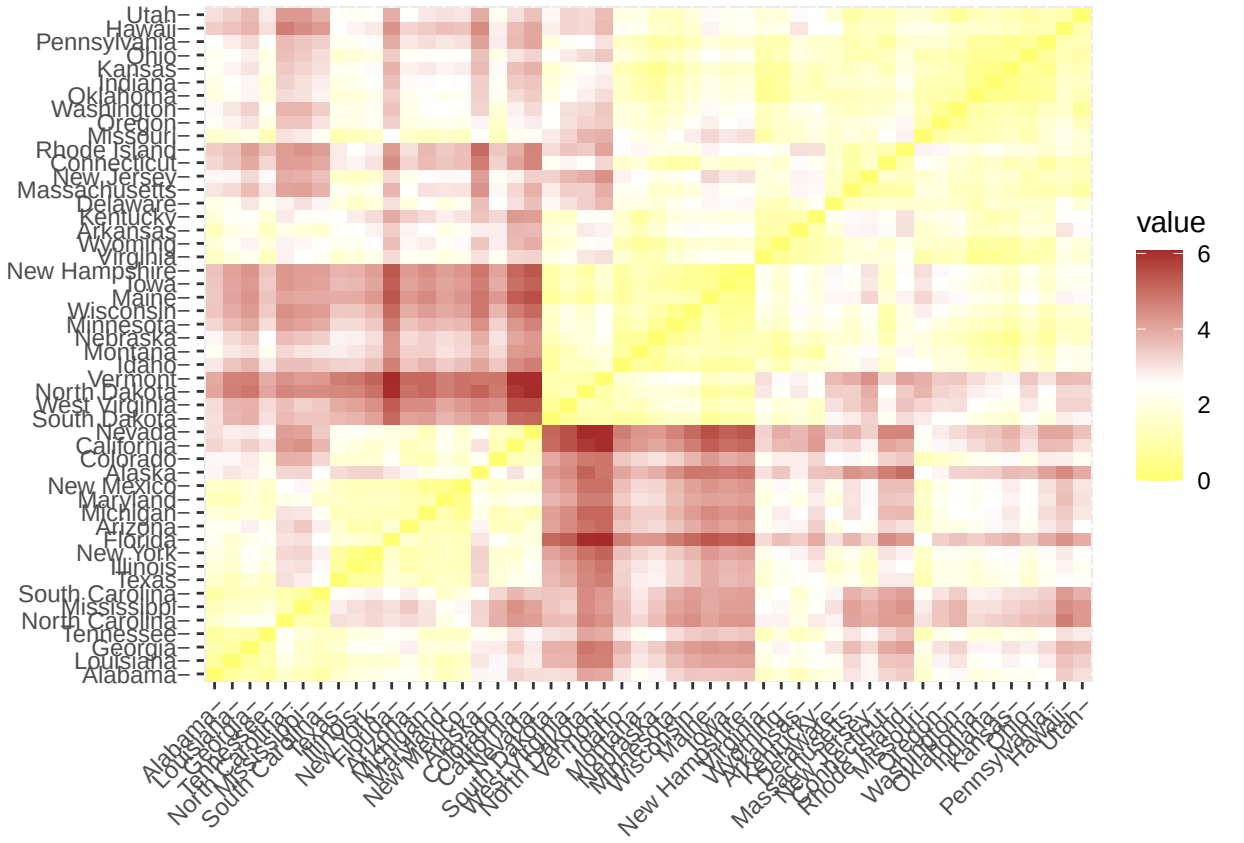
Pulsar este enlace para consultar una expresión formal de estas distancias.

Como se ha dicho anteriormente la elección de la distancia es muy importante. Casi todo el software habitual para análisis cluster utiliza la distancia euclídea, aunque dependiendo del tipo de datos y de las preguntas de investigación planteadas puede interesar otra medida de disimilaridad o distancia.

En R es fácil calcular y visualizar la matriz de distancias entre observaciones con las funciones *get_dist* y *fviz_dist*, respectivamente, incluidas en el paquete *factoextra*. De hecho, aunque por defecto, *get_dist* calcula la distancia euclídea también admite como parámetro todas las distancias mencionadas anteriormente.

La siguiente matriz de distancias muestra en **marrón aquellos estados que presentan grandes disimilitudes** (distancias), frente a aquellos que parecen **más cercanos en amarillo**. Se utiliza el color blanco para referirse a aquellos estados con distancias no tan extremas como para ser consideradas como bajas o altas.

```
distance<- get_dist(df)
fviz_dist(distance, gradient = list(low ="yellow", mid = "white", high = "brown"))
```

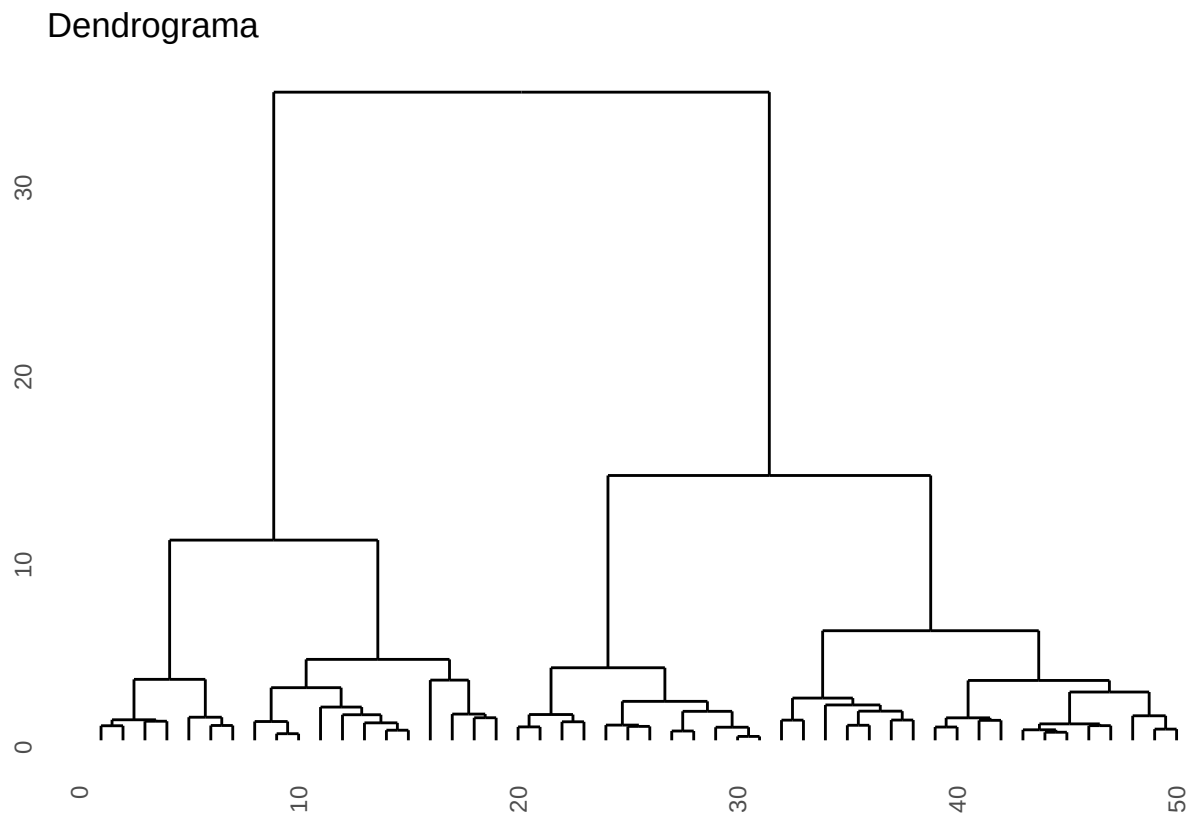


Clustering jerárquico: método de Ward

El **agrupamiento jerárquico** está interesado en encontrar una jerarquía basada en la cercanía o semejanza de los datos según la distancia considerada. En el caso **aglomerativo**, se parte de un grupo con las observaciones más cercanas. A continuación se calculan los siguientes pares más cercanos y de manera ascendente se van generando grupos. Esta construcción se puede observar de forma visual con la construcción de un **dendrograma**.

A continuación se ilustrará cómo los grupos están definidos por la cantidad de líneas verticales del dendrograma, y la selección del número de grupos óptimo se podrá estimar desde este mismo gráfico.

```
dendrogram <- hclust(dist(df, method = 'euclidean'), method = 'ward.D')
ggdendrogram(dendrogram, rotate = FALSE, labels = FALSE, theme_dendro = TRUE) +
  labs(title = "Dendrograma")
```



En el **eje horizontal** del dendrograma tenemos *cada uno de los datos* que componen el conjunto de entrada, mientras que en el **eje vertical** se representa la *distancia euclídea* que existe entre cada grupo a medida que éstos se van jerarquizando.

Cada **línea vertical** del diagrama representa *un agrupamiento*. Conforme se va subiendo en el dendograma termina con un solo gran grupo determinado por la línea horizontal superior. De modo que, **al ir descendiendo en la jerarquía**, se observa que de un solo grupo pasamos a 2, luego a 3, luego a 6, y así sucesivamente.

Una forma de determinar el número ***K* de grupos adecuado** es cortando el dendrograma a aquella altura del diagrama que mejor representa los datos de entrada.

Clustering no jerárquico: algoritmo K-means

Pulsar este enlace para ver una descripción detallada de este algoritmo.

R implementa el **algoritmo K-means** con la función del mismo nombre. Esta función recibe como parámetros de entrada los datos y el número de agrupamientos a realizar (parámetro *centers*). Para abordar el problema de la **elección de los puntos semilla** iniciales incorpora el parámetro *nstart* que prueba múltiples configuraciones iniciales e informa sobre la mejor. Por ejemplo, si *nstart = 25*, generará 25 configuraciones iniciales. El uso de este parámetro es recomendable.

Para este primer ejemplo la función *kmeans* construye dos clusters.

```
k2 <- kmeans(df, centers = 2, nstart = 25)

# Displaying all the fields of the object k2
str(k2)
```

```
## List of 9
##  $ cluster     : Named int [1:50] 1 1 1 2 1 1 2 2 1 1 ...
##   ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ centers     : num [1:2, 1:4] 1.005 -0.67 1.014 -0.676 0.198 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##  $ totss       : num 196
##  $ withinss    : num [1:2] 46.7 56.1
##  $ tot.withinss: num 103
##  $ betweenss   : num 93.1
##  $ size        : int [1:2] 20 30
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

La salida que proporciona la función *kmeans* es una lista de información, de la que destacan las siguientes:

- *cluster*: es un vector de enteros, de 1 a K (K=2 en este caso), que indica el cluster en el que ha sido ubicado cada observación.
- *centers*: una matriz con los sucesivos centros de los clusters.
- *totss*: la suma total de cuadrados.
- *withinss*: vector de suma de cuadrados dentro de cada cluster (un componente por cluster).
- *tot.withinss*: suma total de cuadrados de los cluster, i.e. sum(withinss).
- *betweens*: suma de cuadrados entre grupos, i.e. totss-tot.withinss.
- *size*: el número de observaciones en cada cluster.

Al mostrar la variable *k2* se ve como las agrupaciones dan como resultado 2 tamaños de agrupación de 30 y 20. También se ven los centros de cada grupo (medias) en las cuatro variables (Asesinato, Asalto, UrbanPop, Violación). Y por último la asignación de grupo para cada observación (es decir, Alabama se asignó al grupo 2, Arkansas se asignó al grupo 1, etc.).

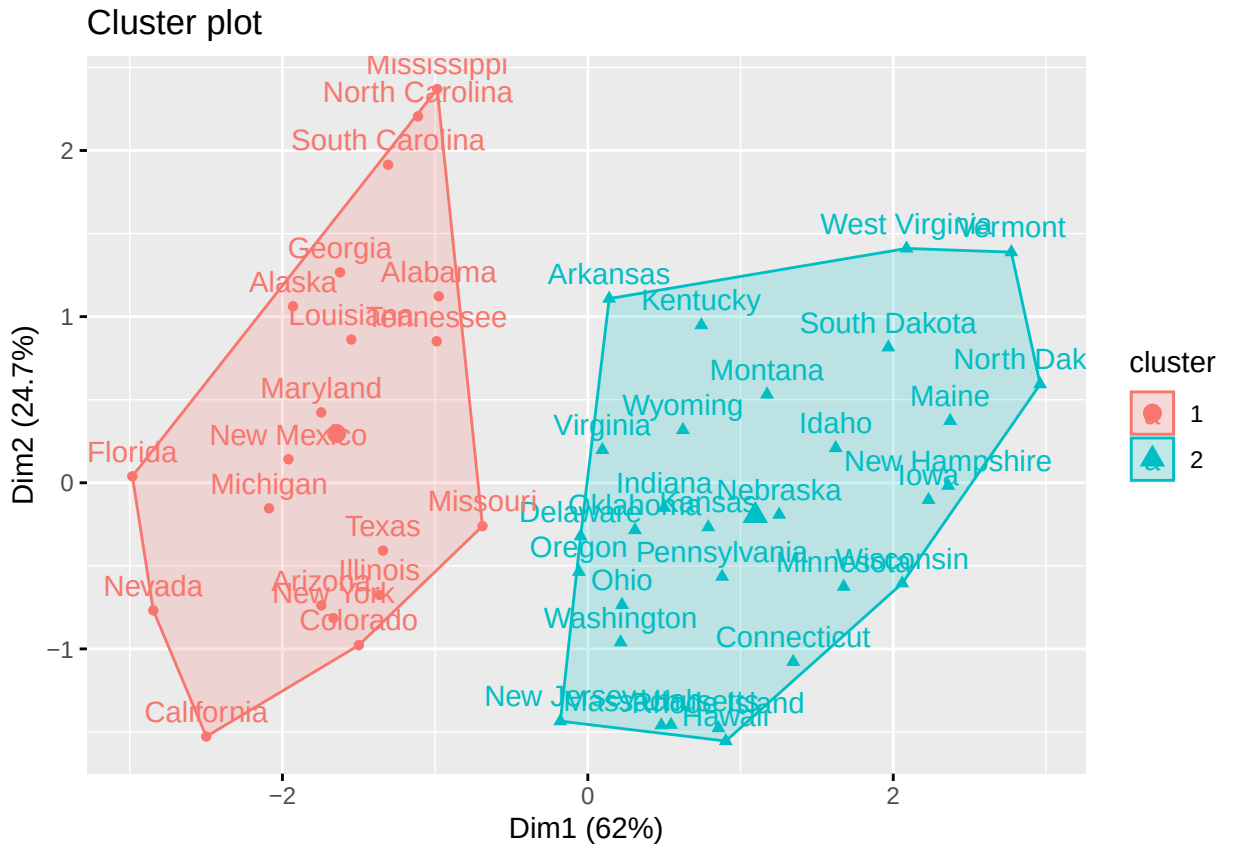
```
k2
```

```
## K-means clustering with 2 clusters of sizes 20, 30
##
## Cluster means:
##      Murder    Assault   UrbanPop       Rape
## 1  1.004934  1.0138274  0.1975853  0.8469650
## 2 -0.669956 -0.6758849 -0.1317235 -0.5646433
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              1              2              2              1              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              2              2              1              2              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              2              2              1              2              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              2              1              1
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              1              2              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              2              2
```

```
##       Oklahoma          Oregon    Pennsylvania    Rhode Island South Carolina
##              2               2               2               2              1
##   South Dakota       Tennessee           Texas            Utah        Vermont
##              2               1               1               2              2
##       Virginia      Washington   West Virginia       Wisconsin        Wyoming
##              2               2               2               2              2
##
## Within cluster sum of squares by cluster:
## [1] 46.74796 56.11445
##  (between_SS / total_SS =  47.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Una forma visual de resumir los resultados de forma elegante y con una interpretación directa es mediante el uso de la función *fviz_cluster*.

```
fviz_cluster(k2,data=df)
```

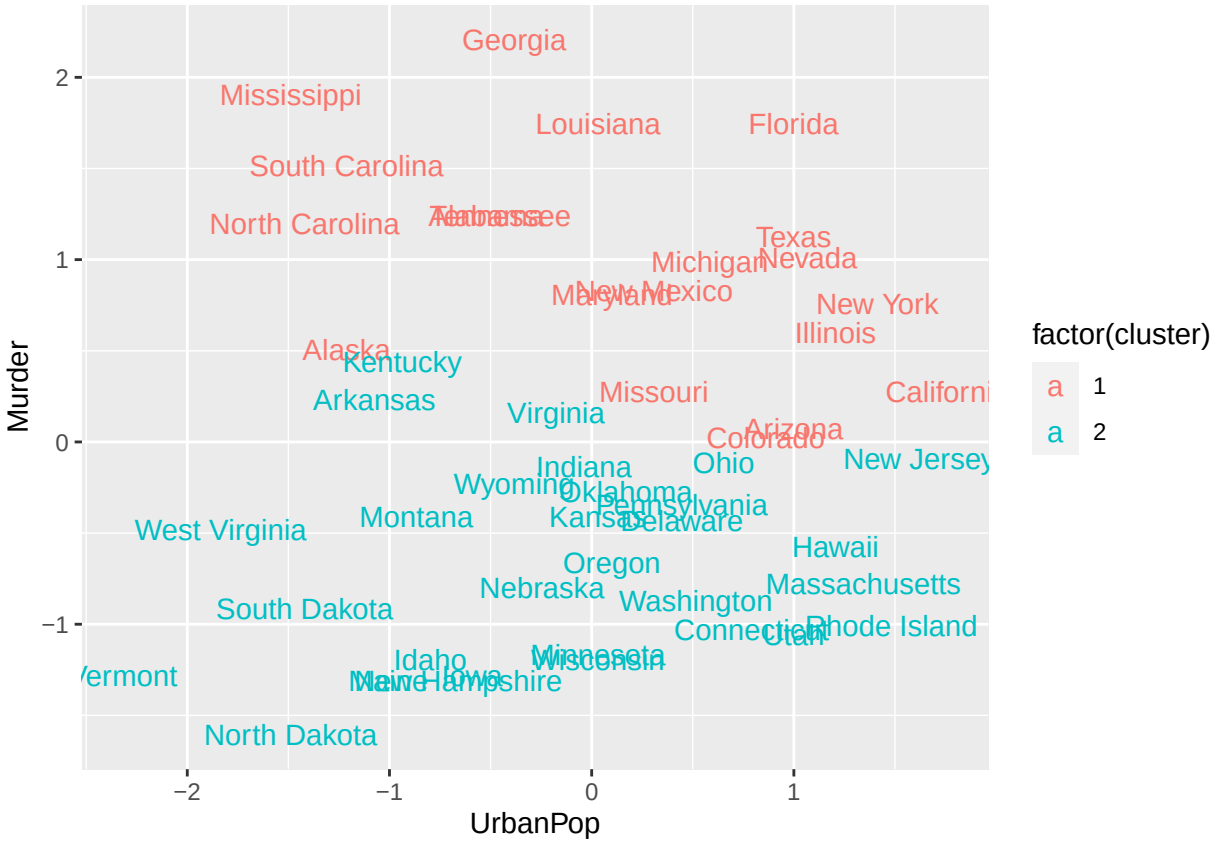


Observación:

Si hay más de dos dimensiones (variables), esta función realizará, en primer lugar, un análisis de componentes principales (ACP) y dibujará los puntos de acuerdo a las dos primeras componentes principales obtenidas (las que explican la mayor parte de la varianza). Es por esto que en el gráfico anterior, *Dim1* y *Dim2* se refieren a estas dos componentes principales.

De forma alternativa se puede utilizar un diagrama de dispersión por pares para ilustrar los grupos en comparación con las variables originales.

```
df %>%
  as_tibble() %>%
  mutate(cluster = k2$cluster,
         state = row.names(USArrests)) %>%
  ggplot(aes(UrbanPop, Murder, color = factor(cluster), label = state)) +
  geom_text()
```



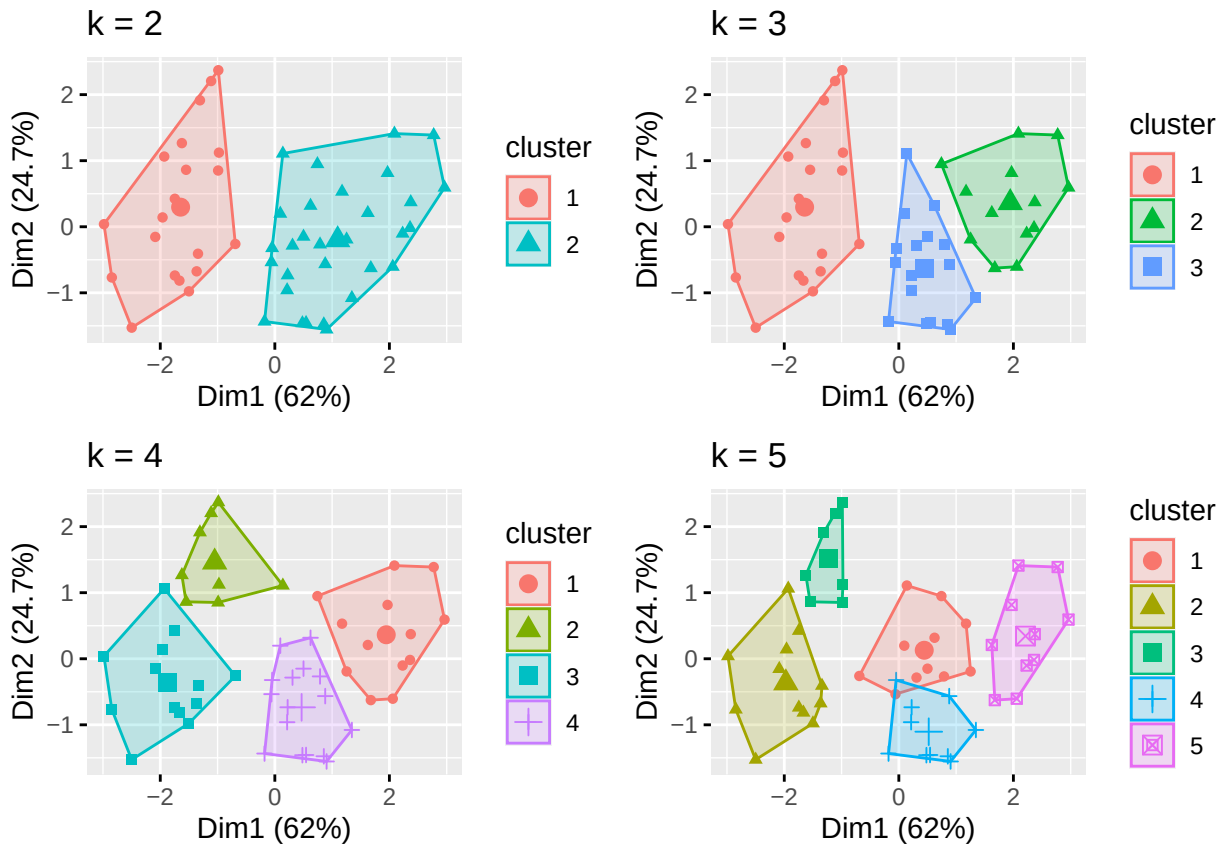
Para usar el algoritmo K-means, el número **K de clusters debe ser fijado de antemano**. Es por esto que es recomendable ejecutar el mismo proceso con otros valores de K (en este ejemplo para K= 3, 4 y 5) para comparar y examinar las diferencias entre los resultados.

```
set.seed(123)

k3 <- kmeans(df, centers = 3, nstart = 25)
k4 <- kmeans(df, centers = 4, nstart = 25)
k5 <- kmeans(df, centers = 5, nstart = 25)

# Plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df) + ggtitle("k = 5")
```

```
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



Aunque esta visualización nos permite deducir donde ocurren las verdaderas diferencias (o no ocurren, como en los clusters 1 y 4 en el gráfico para K=5), **no nos dice cuál es el número óptimo de clusters**.

Determinación del número óptimo de clusters

Tal y como se ha indicado anteriormente, cuando se aplica un método no jerárquico como K-means para realizar un análisis cluster, el investigador debe informar a priori del número de clusters deseado. En este sentido, este investigador estará interesado en **proporcionar de partida un número óptimo de grupos** a formar.

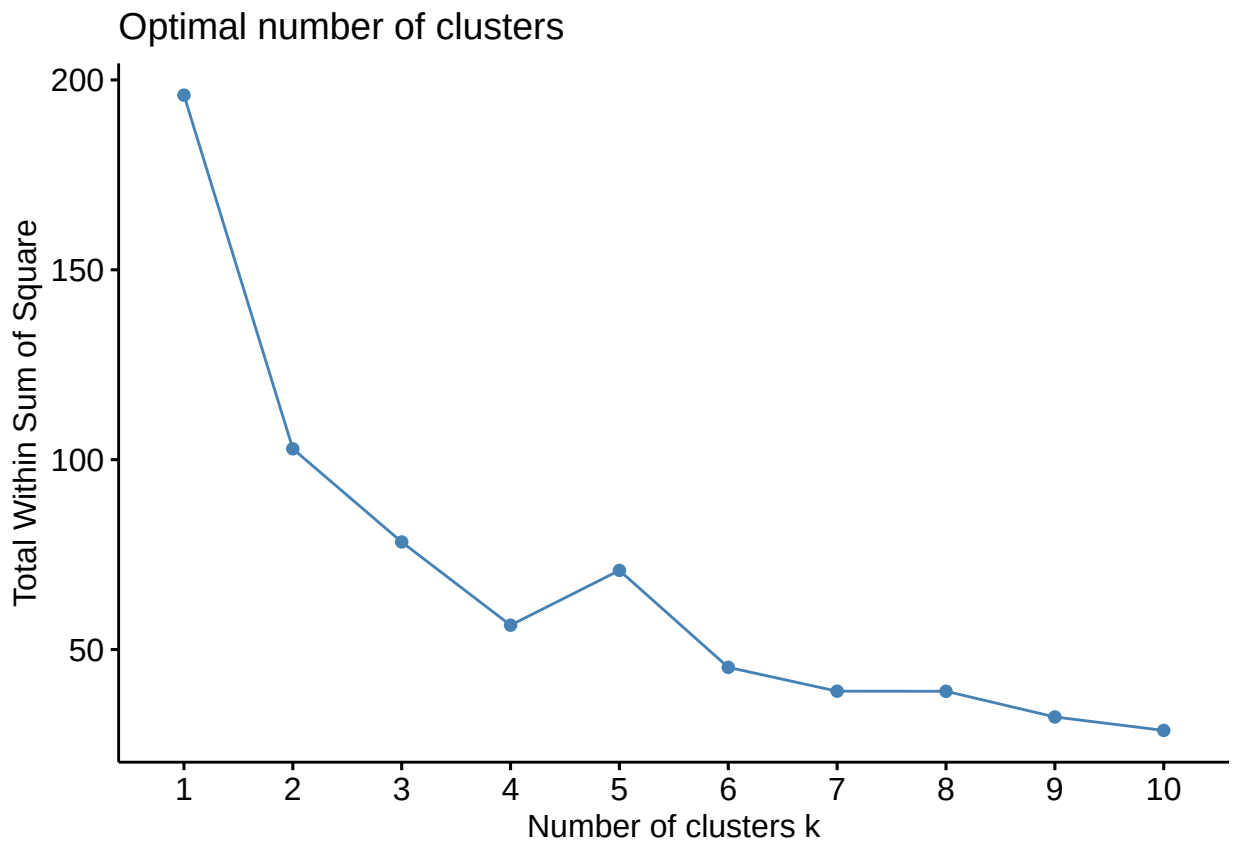
A continuación se presentan tres de los métodos más utilizados para determinar este número óptimo de grupos: método de Elbow, método de Silhouette y el stadístico Gap.

Método de Elbow Teniendo en mente que la idea tras una división en K clusters, es obtener estas agrupaciones de modo que la varianza total intra-grupos sea mínima (total within-cluster variation o total within-cluster sum of square), se puede utilizar el siguiente algoritmo para identificar el número óptimo de clusters:

- Ejecutar un algoritmo de clustering (como K-means) para diferentes valores del K (por ejemplo K=1,...,10).
- Para cada K se calcula la variación total intra-cluster (total within-cluster sum of square, que aquí denotamos por *wss*).
- Se dibuja la curva de *wss* de acuerdo al número de clusters K.
- La localización en esta curva de un 'codo' es tomado como el indicador más apropiado del número de clusters.

Aunque podemos programar este algoritmo en R, el método de Elbow está implementado en la función *fviz_nbclust*.

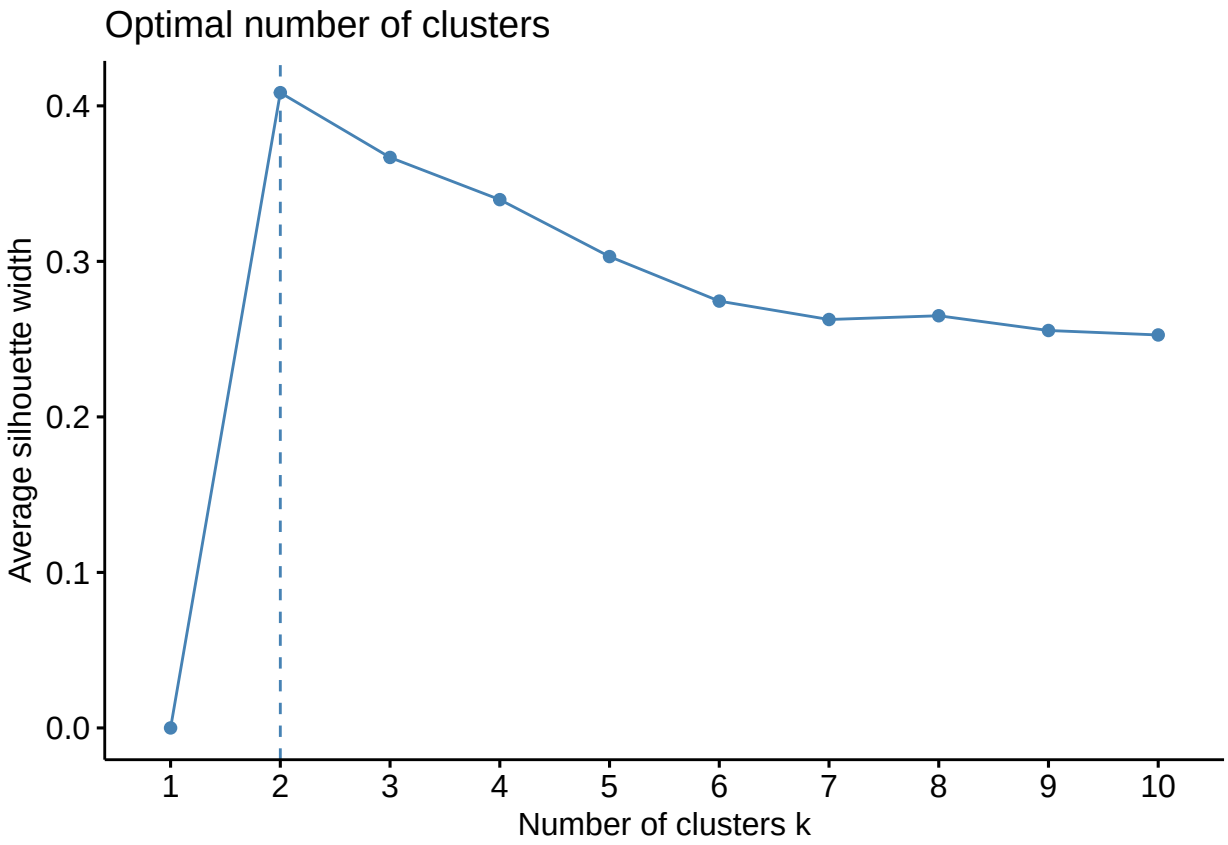
```
set.seed(123)
fviz_nbclust(df, kmeans, method = "wss")
```



Método de Silhouette Este enfoque, *silueta promedio*, mide la calidad de un cluster. Es decir, determina como de adecuado es un objeto dentro de su cluster. El número óptimo de clusters según este enfoque es, de entre un rango de valores posibles para K, aquel que maximiza la silueta promedio.

Como antes, este algoritmo se puede programar en R, pero la función *fviz_nbclust* también lo incluye.

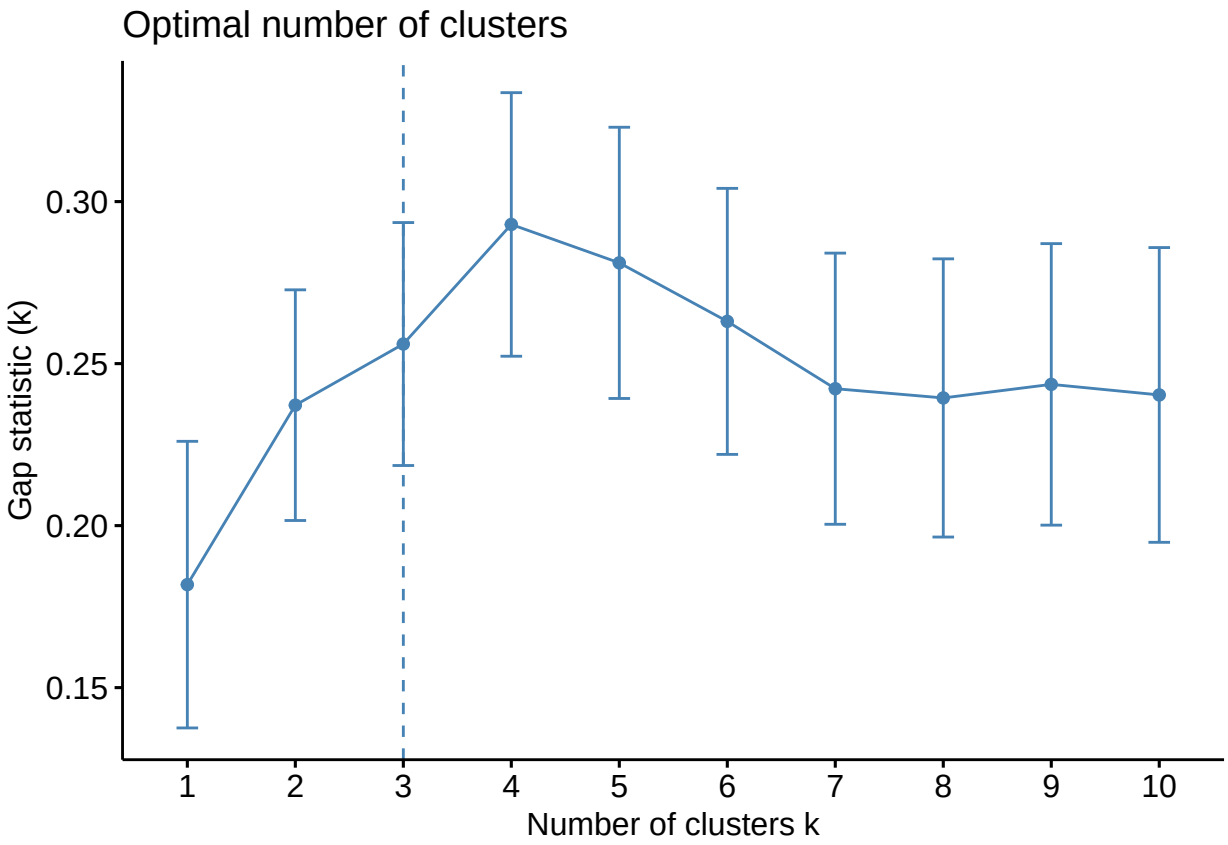
```
set.seed(123)
fviz_nbclust(df, kmeans, method = "silhouette")
```



Método estadístico de brecha (GAP) Este método compara la variación total intracluster para diferentes valores de K. Utiliza simulación Montecarlo en su algoritmo.

La función *clusGap* proporciona el estadístico GAP y su error estándar para una salida. Con la función *fviz_gap_stat* se obtiene una representación gráfica que sugiere un número de clusters.

```
set.seed(123)
gap_stat <- clusGap(df, FUN = kmeans, nstart = 25,K.max = 10, B = 50)
fviz_gap_stat(gap_stat)
```



Análisis de resultados

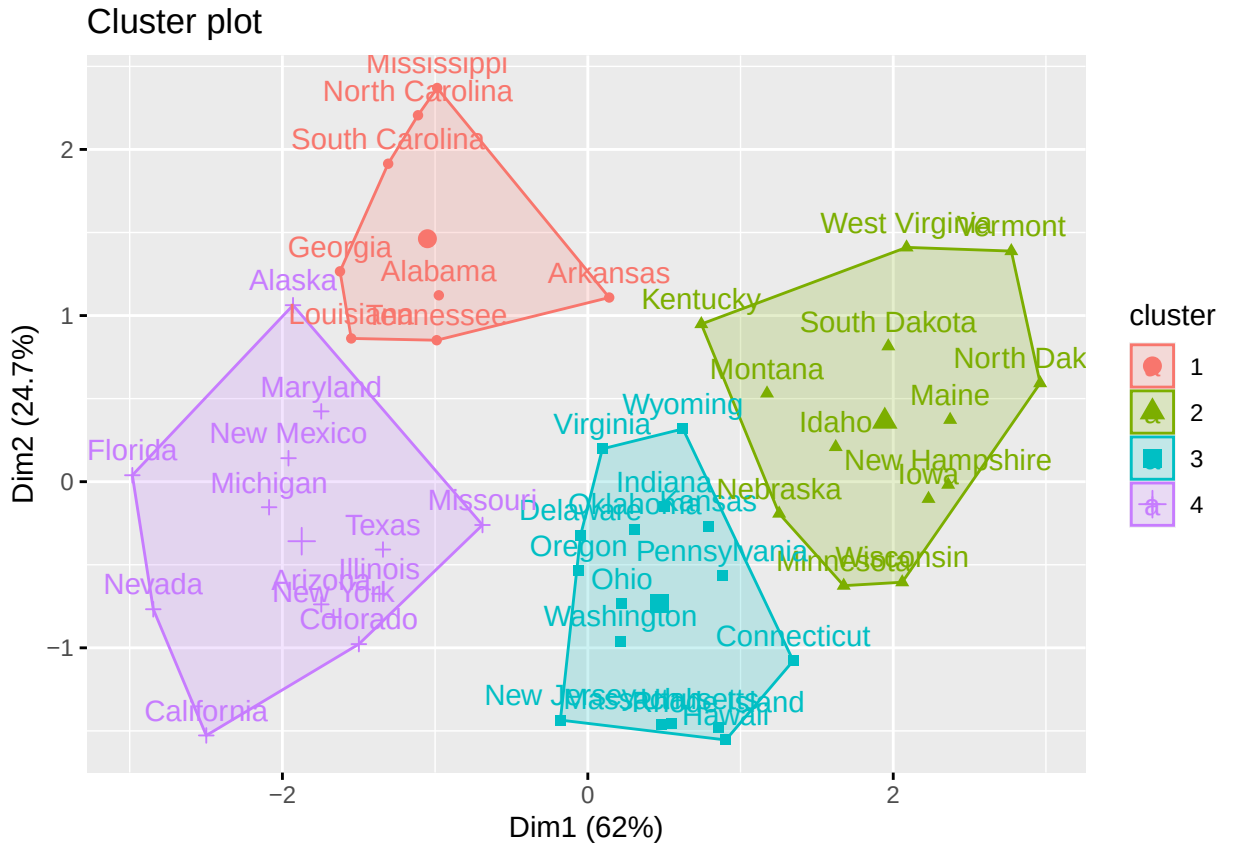
Tras el análisis pormenorizado del número óptimo de clusters, realizado en la sección anterior, parecen ser K=4 el número de grupos más adecuado.

```
set.seed(123)
final <- kmeans(df, 4, nstart = 25)
print(final)
```

```
## K-means clustering with 4 clusters of sizes 8, 13, 16, 13
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1  1.4118898  0.8743346 -0.8145211  0.01927104
## 2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 3 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 4  0.6950701  1.0394414  0.7226370  1.27693964
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              4              4              1              4
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              4              3              3              4              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              2              4              3              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              2              1              2              4
```

```
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              4              2              1              4
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              4              2              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              4              4              1              2              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              2              1              4              3              2
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              2              2              3
##
## Within cluster sum of squares by cluster:
## [1]  8.316061 11.952463 16.212213 19.922437
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# Final output
fviz_cluster(final, data = df)
```



Finalmente, es posible extraer los clusters y añadirlos a nuestros datos iniciales para proporcionar algunos estadísticos descriptivos a nivel de cluster.

```r
USArrests %>%
  mutate(Cluster = final$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

```
## # A tibble: 4 x 5
##   Cluster Murder Assault UrbanPop  Rape
##     <int>  <dbl>   <dbl>    <dbl> <dbl>
## 1       1  13.9    244.      53.8  21.4
## 2       2   3.6     78.5     52.1  12.2
## 3       3   5.66   139.      73.9  18.8
## 4       4  10.8    257.      76    33.2
```