

[Code ▾](#)

Clusterización

30-11-19

- 1 Particionamiento
- 2 Medidas de distancia
 - 2.1 Distancia euclídea
 - 2.2 Distancia de Manhattan
 - 2.3 Correlación
 - 2.3.1 Coeficiente de Pearson
 - 2.3.2 Coeficiente de Spearman
 - 2.3.3 Coeficiente Tau de Kendall
 - 2.3.4 Ejemplo de correlación
 - 2.4 Jackknife correlation
 - 2.4.1 Ejemplo
 - 2.5 Simple matching coefficient
 - 2.6 Índice Jaccard
 - 2.7 Distancia coseno
 - 2.8 Escala de las variables
- 3 K-means
 - 3.1 Definiendo el cluster (agrupamiento)
 - 3.2 Algoritmo de k-medias
 - 3.3 Ejemplo
 - 3.4 Ejemplos
 - 3.4.1 Ejemplo 1
 - 3.4.2 Ejemplo 2 - Agrupando dígitos
 - 3.5 ¿Cuántos clúster?
 - 3.6 Agrupando con datos mixtos
 - 3.7 Métodos populares de inicialización
- 4 k-medoids
 - 4.1 Idea intuitiva
 - 4.2 Ventajas y desventajas
 - 4.3 Ejemplo
- 5 CLARA
 - 5.1 Idea intuitiva
 - 5.2 Ejemplo
- 6 Agrupamiento de k-medoides más rápido
- 7 Fuzzy clustering
 - 7.1 Idea intuitiva
 - 7.2 Ejemplo
- 8 Model based clustering
 - 8.1 Idea intuitiva
 - 8.2 Ejemplo
 - 8.3 K-bumps
- 9 Agrupamiento espacial basado en densidad (DBSCAN)
 - 9.1 Idea intuitiva
 - 9.2 Ejemplo

- 10 Número óptimo de clusters
 - 10.1 Elbow method
 - 10.2 Average silhouette method
 - 10.3 Gap statistic method
- 11 Bibliografía

[Hide](#)

```
pacman::p_load(datasets, cluster, factoextra, purrr, mclust, dplyr, ISLR, ggplot2, tidyverse, g  
gpabr)
```

1 Particionamiento

- Los métodos de análisis de agrupamiento (análisis de clústeres) tienen por objeto dividir la muestra en grupos de individuos. Esta división se hace de modo que los grupos sean diferentes entre sí, pero los individuos pertenecientes al mismo grupo sean parecidos entre sí. Tales métodos pueden ser utilizados con diversas finalidades.
- El término clustering hace referencia a un amplio abanico de técnicas unsupervised cuya finalidad es encontrar patrones o grupos (clusters) dentro de un conjunto de observaciones. Las particiones se establecen de forma que, las observaciones que están dentro de un mismo grupo, son similares entre ellas y distintas a las observaciones de otros grupos. Se trata de un método unsupervised, ya que el proceso ignora la variable respuesta que indica a qué grupo pertenece realmente cada observación (si es que existe tal variable). Esta característica diferencia al clustering de las técnicas estadísticas conocidas como análisis discriminante, que emplean un set de entrenamiento en el que se conoce la verdadera clasificación.

Dada la utilidad del clustering en disciplinas muy distintas (genómica, marketing...), se han desarrollado multitud de variantes y adaptaciones de sus métodos y algoritmos. Pueden diferenciarse tres grupos principales:

- **Clustering basado en particiones (Partitioning Clustering):** Este tipo de algoritmos requieren que el usuario especifique de antemano el número de clusters que se van a crear (K-means, K-medoids, CLARA).
- **Clustering jerárquico (Hierarchical Clustering):** Este tipo de algoritmos no requieren que el usuario especifique de antemano el número de clusters. (agglomerative clustering, divisive clustering).
- Métodos que combinan o modifican los anteriores (hierarchical K-means, fuzzy clustering, model based clustering y density based clustering).

En el entorno de programación R existen múltiples paquetes que implementan algoritmos de clustering y funciones para visualizar sus resultados. A continuación mencionamos algunos:

- **stats:** contiene las funciones `dist()` para calcular matrices de distancias, `kmeans()`, `hclust()`, `cuttree()` para crear los clusters y `plot.hclust()` para visualizar los resultados.
- **cluster, mclust:** contienen múltiples algoritmos de clustering y métricas para evaluarlos.
- **factoextra:** extensión basada en `ggplot2` para crear visualizaciones de los resultados de clustering y su evaluación.
- **dendextend:** extensión para la customización de dendrogramas.

Ejemplos

- Una tienda de departamentos puede utilizar análisis de agrupación para orientar a sus clientes. Así, puede decidir cómo debe actuar con cada grupo. Por ejemplo, puede decidir enviar cartas con ciertos anuncios a un grupo, pero no a otro.
- Otro ejemplo son las reseñas de usuarios mostradas en sitios de compras en línea cuando un individuo busca un producto. En general la empresa escoge automáticamente dos o tres reseñas bastante distintas, pero que al mismo tiempo son representativas de las miles de reseñas hechas sobre ese producto.

Formalmente, el objetivo de um método de clustering es crear una partición C_1, \dots, C_K de los elementos muestrales $\{1, \dots, n\}$. Esto es, debemos tener al mismo tiempo

$$C_1 \cup C_2 \dots \cup C_k = \{1, \dots, n\}$$

y

$$C_i \cap C_j = \emptyset \forall i \neq j$$

Un concepto esencial a diversos métodos de agrupamiento es como medir disimilaridad (o similaridad) entre dos individuos con covariables x_i y x_j . Hay varias medidas posibles.

2 Medidas de distancia

- Todos los métodos de clustering tienen una cosa en común, para poder llevar a cabo las agrupaciones necesitan definir y cuantificar la similitud entre las observaciones. El término distancia se emplea dentro del contexto del clustering como cuantificación de la similitud o diferencia entre observaciones. Si se representan las observaciones en un espacio p dimensional, siendo p el número de variables asociadas a cada observación, cuando más se asemejen dos observaciones más próximas estarán, de ahí que se emplee el término distancia.

2.1 Distancia euclídea

La distancia euclídea entre dos puntos p y q se define como la longitud del segmento que une ambos puntos. En coordenadas cartesianas, la distancia euclídea se calcula empleando el teorema de Pitágoras. Por ejemplo, en un espacio de dos dimensiones en el que cada punto está definido por las coordenadas (x, y) , la distancia euclídea entre p y q viene dada por la ecuación:

$$d_{euc}(p, q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

Esta ecuación puede generalizarse para un espacio euclídeo n -dimensional donde cada punto está definido por un vector de n coordenadas: $p = (p_1, p_2, p_3, \dots, p_n)$ y $q = (q_1, q_2, q_3, \dots, q_n)$.

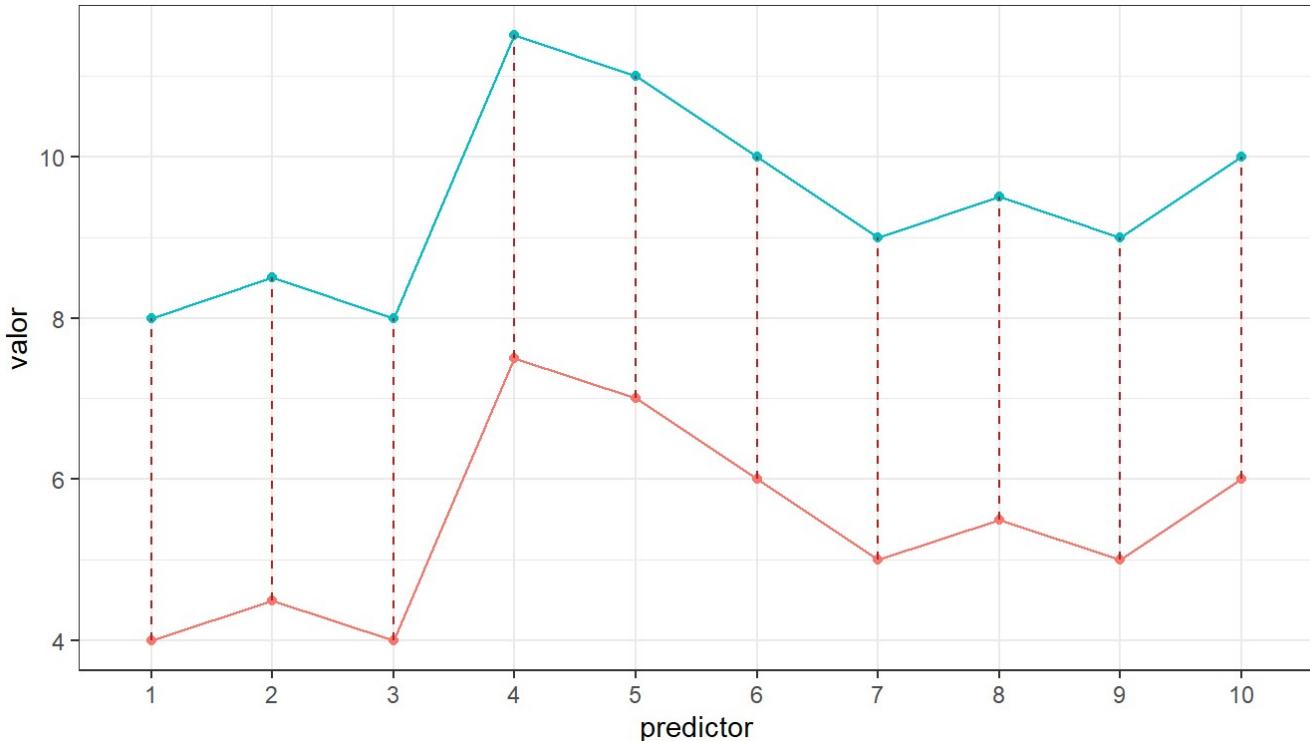
$$d_{euc}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Una forma de dar mayor peso a aquellas observaciones que están más alejadas es emplear la distancia euclídea al cuadrado. En el caso del clustering, donde se busca agrupar observaciones que minimicen la distancia, esto se traduce en una mayor influencia de aquellas observaciones que están más distantes.

La siguiente imagen muestra el perfil de dos observaciones definidas por 10 variables (espacio con 10 dimensiones).

[Hide](#)

```
library(ggplot2)
observacion_a <- c(4, 4.5, 4, 7.5, 7, 6, 5, 5.5, 5, 6)
observacion_b <- c(4, 4.5, 4, 7.5, 7, 6, 5, 5.5, 5, 6) + 4
datos <- data.frame(observacion = rep(c("a", "b"), each = 10),
                     valor = c(observacion_a, observacion_b),
                     predictor = 1:10)
ggplot(data = datos, aes(x = as.factor(predictor), y = valor,
                           colour = observacion)) +
  geom_path(aes(group = observacion)) +
  geom_point() +
  geom_line(aes(group = predictor), colour = "firebrick", linetype = "dashed") +
  labs(x = "predictor") +
  theme_bw() +
  theme(legend.position = "none")
```



La distancia euclidiana entre las dos observaciones equivale a la raíz cuadrada de la suma de las longitudes de los segmentos rojos que unen cada par de puntos. Tiene en cuenta por lo tanto el desplazamiento individual de cada una de las variables.

2.2 Distancia de Manhattan

- La distancia de Manhattan, también conocida como la métrica del taxista (taxicab metric), distancia rectilínea o distancia L1, define la distancia entre dos puntos p y q como el sumatorio de las diferencias absolutas entre cada dimensión. Esta medida se ve menos afectada por outliers (es más robusta) que la distancia euclídea debido a que no eleva al cuadrado las diferencias.

$$d_{man}(p, q) = \sum_{i=1}^n |(p_i - q_i)|$$

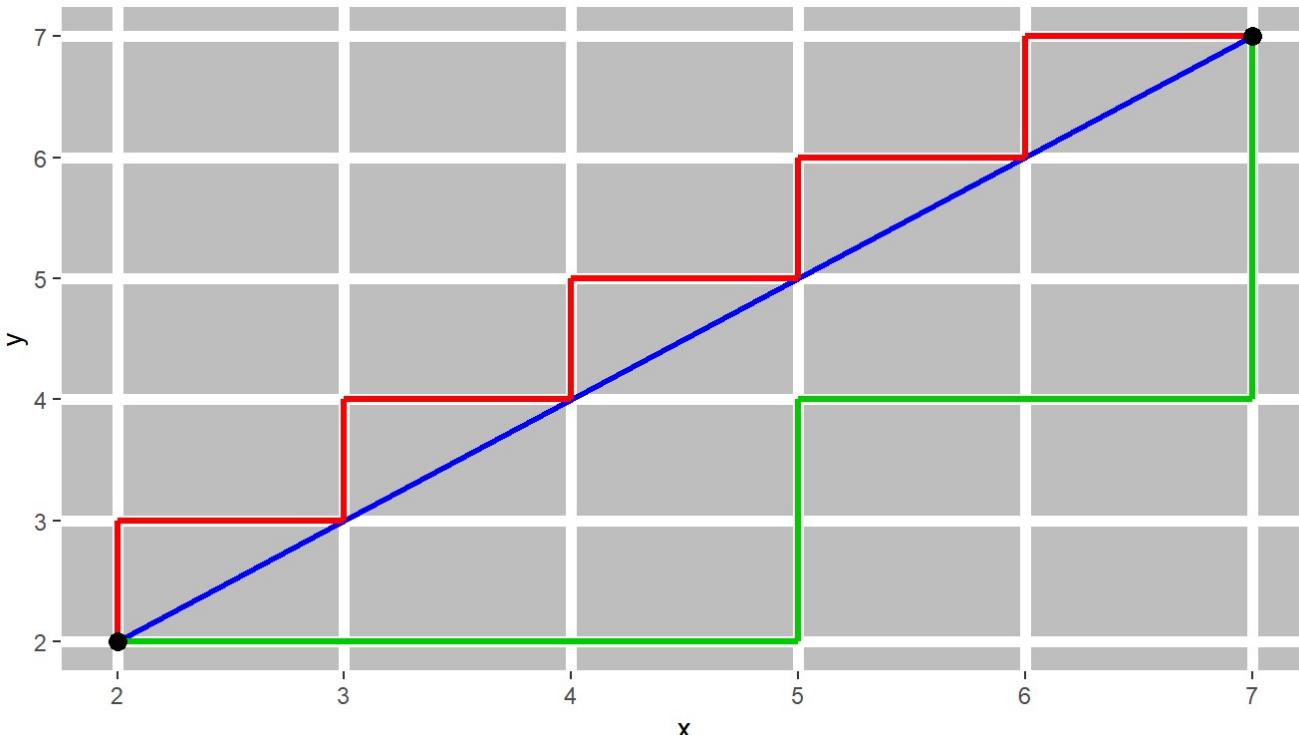
- La siguiente imagen muestra una comparación entre la distancia euclídea (segmento azul) y la distancia de manhattan (segmento rojo y verde) en un espacio bidimensional. Existen múltiples caminos para unir dos puntos con el mismo valor de distancia de manhattan, ya que su valor es igual al desplazamiento total en cada una de las dimensiones.

[Hide](#)

```
datos <- data.frame(observacion = c("a", "b"), x = c(2, 7), y = c(2, 7))
manhattan <- data.frame(
  x = rep(2:6, each = 2),
  y = rep(2:6, each = 2) + rep(c(0,1), 5),
  xend = rep(2:6, each = 2) + rep(c(0,1), 5),
  yend = rep(3:7, each = 2))

manhattan_2 <- data.frame(
  x = c(2, 5, 5, 7),
  y = c(2, 2, 4, 4),
  xend = c(5, 5, 7, 7),
  yend = c(2, 4, 4, 7))

ggplot(data = datos, aes(x = x, y = y)) +
  geom_segment(aes(x = 2, y = 2, xend = 7, yend = 7), color = "blue", size = 1.2) +
  geom_segment(data = manhattan, aes(x = x, y = y, xend = xend, yend = yend),
               color = "red", size = 1.2) +
  geom_segment(data = manhattan_2, aes(x = x, y = y, xend = xend, yend = yend),
               color = "green3", size = 1.2) +
  geom_point(size = 3) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(size = 2),
        panel.background = element_rect(fill = "gray",
                                         colour = "white",
                                         size = 0.5, linetype = "solid"))
```



2.3 Correlación

- La correlación es una medida de distancia muy útil cuando la definición de similitud se hace en términos de patrón o forma y no de desplazamiento o magnitud. ¿Qué quiere decir esto? En la imagen del apartado de la distancia euclídea, las dos observaciones tienen exactamente el mismo patrón, la única diferencia es que una de ellas está desplazada 4 unidades por encima de la otra. Si se emplea como medida de similitud 1 menos el valor de la correlación, ambas observaciones se consideran idénticas (su distancia es 0).

$$d_{cor}(p, q) = 1 - \text{correlacion}(p, q)$$

donde la correlación puede ser de distintos tipos (Pearson, Spearman, Kendall...) Correlación lineal.

[Hide](#)

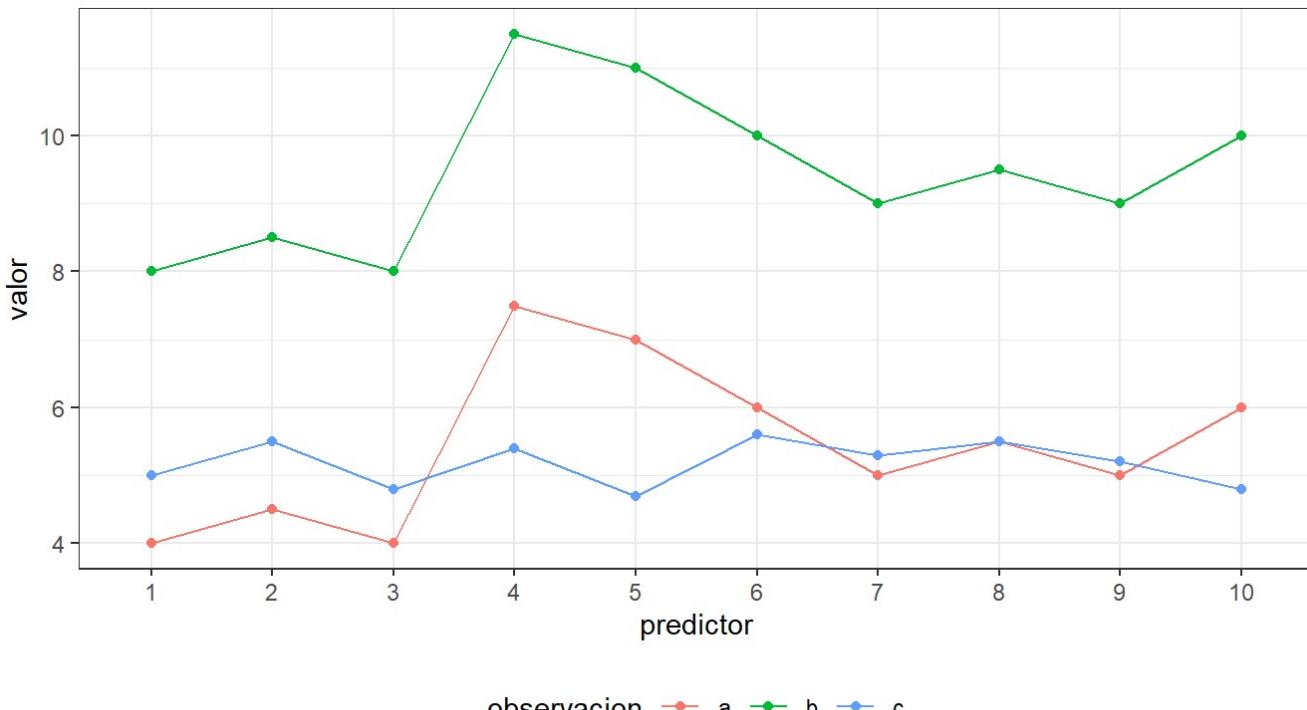
```

library(ggplot2)
observacion_a <- c(4, 4.5, 4, 7.5, 7, 6, 5, 5.5, 5, 6)
observacion_b <- c(4, 4.5, 4, 7.5, 7, 6, 5, 5.5, 5, 6) + 4
observacion_c <- c(5, 5.5, 4.8, 5.4, 4.7, 5.6, 5.3, 5.5, 5.2, 4.8)

datos <- data.frame(observacion = rep(c("a", "b", "c"), each = 10),
                     valor = c(observacion_a, observacion_b, observacion_c),
                     predictor = 1:10)

ggplot(data = datos, aes(x = as.factor(predictor), y = valor,
                         colour = observacion)) +
  geom_path(aes(group = observacion)) +
  geom_point() +
  labs(x = "predictor") +
  theme_bw() +
  theme(legend.position = "bottom")

```



Hide

```
dist(x = rbind(observacion_a, observacion_b, observacion_c), method = "euclidean")
```

```

##               observacion_a   observacion_b
## observacion_b      12.64911
## observacion_c      3.75100      13.98821

```

Hide

```
1 - cor(x = cbind(observacion_a, observacion_b, observacion_c), method = "pearson")
```

```
##               observacion_a  observacion_b  observacion_c
## observacion_a      0.0000000   0.0000000   0.9466303
## observacion_b      0.0000000   0.0000000   0.9466303
## observacion_c      0.9466303   0.9466303   0.0000000
```

Este ejemplo pone de manifiesto que no existe una única medida de distancia que sea mejor que las demás, sino que, dependiendo del contexto, una será más adecuada que otra.

- El coeficiente de correlación de Spearman es menos sensible que el de Pearson para los valores muy lejos.

2.3.1 Coeficiente de Pearson

El coeficiente de correlación de Pearson es la covarianza estandarizada, y su ecuación difiere dependiendo de si se aplica a una muestra, Coeficiente de Pearson muestral (r), o si se aplica la población Coeficiente de Pearson poblacional (ρ).

$$\rho = \frac{Cov(X, Y)}{\sigma_x \sigma_y}$$

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

Condiciones

- La relación que se quiere estudiar entre ambas variables es lineal (de lo contrario, el coeficiente de Pearson no la puede detectar).
- Las dos variables deben de ser cuantitativas.
- Normalidad: ambas variables se tienen que distribuir de forma normal. Varios textos defienden su robustez cuando las variables se alejan moderadamente de la normal.
- Homocedasticidad: La varianza de Y debe ser constante a lo largo de la variable X . Esto se puede identificar si en el scatterplot los puntos mantienen la misma dispersión en las distintas zonas de la variable X . Esta condición no la he encontrado mencionada en todos los libros.

Características

- Toma valores entre $[-1, +1]$, siendo $+1$ una correlación lineal positiva perfecta y -1 una correlación lineal negativa perfecta.
- Es una medida independiente de las escalas en las que se midan las variables.
- No varía si se aplican transformaciones a las variables.
- No tiene en consideración que las variables sean dependientes o independientes.
- El coeficiente de correlación de Pearson no equivale a la pendiente de la recta de regresión.
- Es sensible a outliers, por lo que se recomienda en caso de poder justificarlos, excluirlos del análisis.

Interpretación

Además del valor obtenido para el coeficiente, es necesario calcular su significancia. Solo si el p-value es significativo se puede aceptar que existe correlación y esta será de la magnitud que indique el coeficiente. Por muy cercano que sea el valor del coeficiente de correlación a $+1$ o -1 , si no es significativo, se ha de interpretar que la correlación de ambas variables es 0 ya que el valor observado se puede deber al azar. (Ver más

adelante como calcular la significancia).

2.3.2 Coeficiente de Spearman

El coeficiente de Spearman es el equivalente al coeficiente de Pearson pero con una previa transformación de los datos a rangos. Se emplea como alternativa cuando los valores son ordinales, o bien, cuando los valores son continuos pero no satisfacen la condición de normalidad requerida por el coeficiente de Pearson y se pueden ordenar transformándolos en rangos. Al trabajar con rangos, es menos sensible que Pearson a valores extremos. Existe una diferencia adicional con respecto a Pearson. El coeficiente de Spearman requiere que la relación entre las variables sea monótona, es decir, que cuando una variable crece la otra también lo hace o cuando una crece la otra decrece (que la tendencia sea constante). Este concepto no es exactamente el mismo que linealidad.

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},$$

Siendo d_i la distancia entre los rangos de cada observación ($x_i - y_i$) y n el número de observaciones.

2.3.3 Coeficiente Tau de Kendall

Trabaja con rangos, por lo que requiere que las variables cuya relación se quiere estudiar sean ordinales o que se puedan transformar en rangos. Al ser no paramétrico, es otra alternativa al Coeficiente de correlación de Pearson cuando no se cumple la condición de normalidad. Parece ser más aconsejable que el coeficiente de Spearman cuando el número de observaciones es pequeño o los valores se acumulan en una región por lo que el número de ligaduras al generar los rangos es alto.

$$\tau = \frac{C - D}{\frac{1}{2}n(n - 1)},$$

Siendo C el número de pares concordantes, aquellos en los que el rango de la segunda variable es mayor que el rango de la primera variable. D el número de pares discordantes, cuando el rango de la segunda es igual o menor que el rango de la primera variable.

Tau representa una probabilidad; es decir, es la diferencia entre la probabilidad de que las dos variables estén en el mismo orden en los datos observados y la probabilidad de que las dos variables estén en diferentes órdenes.

2.3.4 Ejemplo de correlación

Se dispone de un data set con información sobre diferentes coches. Se quiere estudiar si existe una correlación entre el peso de un vehículo (Weight) y la potencia de su motor (Horsepower).

R contiene funciones que permiten calcular los diferentes tipos de correlaciones y sus niveles de significancia: `cor()` y `cor.test()`. La segunda función es más completa ya que además de calcular el coeficiente de correlación indica su significancia (p-value) e intervalo de confianza.

Hide

```
library(MASS)
library(ggplot2)
data("Cars93")
```

En primer lugar se representan las dos variables mediante un diagrama de dispersión para intuir si existe relación lineal o monotónica. Si no la hay, no tiene sentido calcular este tipo de correlaciones.

[Hide](#)

```
ggplot(data = Cars93, aes(x = Weight, y = Horsepower)) +
  geom_point(colour = "red4") +
  ggtitle("Diagrama de dispersión") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

El diagrama de dispersión parece indicar una posible relación lineal positiva entre ambas variables.

Para poder elegir el coeficiente de correlación adecuado, se tiene que analizar el tipo de variables y la distribución que presentan. En este caso, ambas variables son cuantitativas continuas y pueden transformarse en rangos para ordenarlas, por lo que a priori los tres coeficientes podrían aplicarse. La elección se hará en función de la distribución que presenten las observaciones.

1. Análisis de normalidad

[Hide](#)

```
# Representación gráfica
par(mfrow = c(1, 2))
hist(Cars93$Weight, breaks = 10, main = "", xlab = "Weight", border = "darkred")
hist(Cars93$Horsepower, breaks = 10, main = "", xlab = "Horsepower",
     border = "blue")
```

[Hide](#)

```
qqnorm(Cars93$Weight, main = "Weight", col = "darkred")
qqline(Cars93$Weight)
qqnorm(Cars93$Horsepower, main = "Horsepower", col = "blue")
qqline(Cars93$Horsepower)
par(mfrow = c(1,1))
```

El análisis gráfico y el contraste de normalidad muestran que para la variable Horsepower no se puede asumir normalidad y que la variable Weight está en el límite. Siendo estrictos, este hecho excluye la posibilidad de utilizar el coeficiente de Pearson, dejando como alternativas el de Spearman o Kendall. Sin embargo, dado que la distribución no se aleja mucho de la normalidad y de que el coeficiente de Pearson tiene cierta robustez, a fines prácticos sí que se podría utilizar siempre y cuando se tenga en cuenta este hecho en los resultados. Otra posibilidad es tratar de transformar las variables para mejorar su distribución.

[Hide](#)

```
# Representación gráfica
par(mfrow = c(1, 2))
hist(log10(Cars93$Horsepower), breaks = 10, main = "", xlab = "Log10(Horsepower)",
     border = "blue")
qqnorm(log10(Cars93$Horsepower), main = "", col = "blue")
qqline(log10(Cars93$Horsepower))
```

[Hide](#)

```
par(mfrow = c(1, 1))
shapiro.test(log10(Cars93$Horsepower))
```

La transformación logarítmica de la variable Horsepower consigue una distribución de tipo normal.

2.Homocedasticidad

La homocedasticidad implica que la varianza se mantenga constante. Puede analizarse de forma gráfica representando las observaciones en un diagrama de dispersión y viendo si mantiene una homogeneidad en su dispersión a lo largo del eje X. Una forma cónica es un claro indicativo de falta de homocedasticidad. En algunos libros se menciona el test de Goldfeld-Quandt o el de Breusch-Pagan como test de hipótesis para la homocedasticidad en correlación y regresión.

Tal como muestra el diagrama de dispersión generado al inicio del ejercicio, sí hay un patrón cónico. Esto debe de tenerse en cuenta si se utiliza Pearson puesto que viola una de sus condiciones.

[Hide](#)

```
ggplot(data = Cars93, aes(x = Weight, y = Horsepower)) +
  geom_point(colour = "red4") +
  geom_segment(aes(x = 1690, y = 70, xend = 3100, yend = 300), linetype="dashed") +
  geom_segment(aes(x = 1690, y = 45, xend = 4100, yend = 100), linetype="dashed") +
  ggtitle("Diagrama de dispersión") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

3.Cálculo de correlación

Debido a la falta de homocedasticidad, los resultados generados por Pearson no son precisos, desde el punto de vista teórico Spearman o Kendall son más adecuados. Sin embargo, en la bibliografía emplean Pearson, así que se van a calcular tanto Pearson como Spearman.

[Hide](#)

```
cor(x = Cars93$Weight, y = log10(Cars93$Horsepower), method = "pearson")
```

[Hide](#)

```
cor(x = Cars93$Weight, y = log10(Cars93$Horsepower), method = "spearman")
```

Ambos test muestran una correlación alta (>0.8). Sin embargo para poder considerar que existe realmente correlación entre las dos variables es necesario calcular su significancia, de lo contrario podría deberse al azar.

4.Significancia de la correlación

Por muy alto que sea un coeficiente de correlación, si no es significativa se ha de considerar inexistente.

Hide

```
cor.test(x = Cars93$Weight,  
         y = log10(Cars93$Horsepower),  
         alternative = "two.sided",  
         conf.level = 0.95,  
         method      = "pearson")
```

Hide

```
cor.test(x = Cars93$Weight,  
         y = log10(Cars93$Horsepower),  
         alternative = "two.sided",  
         conf.level = 0.95,  
         method      = "spearman")
```

Ambos coeficientes de correlación son significativos ($p_value \approx 0$).

5.Coefficiente de determinación R2 (tamaño del efecto)

Hide

```
R2_pearson <- cor(x = Cars93$Weight,  
                     y = log10(Cars93$Horsepower),  
                     method = "pearson")  
R2_pearson <- R2_pearson^2  
R2_pearson
```

Hide

```
R2_spearman <- cor(x = Cars93$Weight,  
                     y = log10(Cars93$Horsepower),  
                     method = "spearman")  
R2_spearman <- R2_spearman^2  
R2_spearman
```

6.Conclusión

Existe una correlación significativa entre el peso del vehículo y la potencia de su motor ($r=0.8$, $p\text{-value} < 2.2e-16$), con un tamaño de efecto medio-alto ($R^2 = 0.66$).

2.4 Jackknife correlation

El coeficiente de correlación de Pearson resulta efectivo en ámbitos muy diversos. Sin embargo, tiene la desventaja de no ser robusto frente a outliers a pesar de que se cumpla la condición de normalidad. Si dos variables tienen un pico o un valle común en una única observación, por ejemplo, por un error de lectura, la correlación va a estar dominada por este registro a pesar de que entre las dos variables no haya correlación real alguna. Lo mismo puede ocurrir en la dirección opuesta. Si dos variables están altamente correlacionadas excepto para una observación, en la que los valores son muy dispares, entonces la correlación existente quedará enmascarada. Una forma de evitarlo es recurrir a la Jackknife correlation, que consiste en calcular

todos los posibles coeficientes de correlación entre dos variables si se excluye cada vez una de las observaciones. El promedio de todas las Jackknife correlations calculadas atenua en cierta medida el efecto del outlier.

$$\bar{\theta}_{(A,B)} = \text{Promedio Jackknife correlation (A,B)} = \frac{1}{n} \sum_{i=1}^n \hat{r}_i$$

donde n es el número de observaciones y \hat{r}_i es el coeficiente de correlación entre las variables A y B , habiendo excluido la observación i .

Además del promedio, se puede estimar su error estándar (SE) y así obtener intervalos de confianza para la Jackknife correlation y su correspondiente p-value.

$$SE = \sqrt{\frac{n-1}{n} \sum_{i=1}^n (\hat{r}_i - \bar{\theta})^2}$$

Intervalo de confianza del 95% ($Z = 1.96$)

$$\text{Promedio Jackknife correlation (A,B)} \pm 1.96 * SE$$

$$\bar{\theta} - 1.96 \sqrt{\frac{n-1}{n} \sum_{i=1}^n (\hat{r}_i - \bar{\theta})^2}, \quad \bar{\theta} + 1.96 \sqrt{\frac{n-1}{n} \sum_{i=1}^n (\hat{r}_i - \bar{\theta})^2}$$

P-value para la hipótesis nula de que $\bar{\theta} = 0$:

$$Z_{calculada} = \frac{\bar{\theta} - H_0}{SE} = \frac{\bar{\theta} - 0}{\sqrt{\frac{n-1}{n} \sum_{i=1}^n (\hat{r}_i - \bar{\theta})^2}}$$

$$p_{value} = P(Z > Z_{calculada})$$

Cuando se emplea este método, es conveniente calcular la diferencia entre el valor de correlación obtenido por Jackknife correlation ($\bar{\theta}$) y el que se obtiene si se emplean todas las observaciones (\bar{r}). A esta diferencia se le conoce como Bias. Su magnitud es un indicativo de cuánto está influenciada la estimación de la correlación entre dos variables debido a un valor atípico u outlier.

$$Bias = (n - 1) * (\bar{\theta} - \bar{r})$$

Si se calcula la diferencia entre cada correlación (\hat{r}_i) estimada en el proceso de Jackknife y el valor de correlación (\bar{r}) obtenido si se emplean todas las observaciones, se puede identificar qué observaciones son más influyentes.

Cuando el estudio requiere minimizar al máximo la presencia de falsos positivos, a pesar de que se incremente la de falsos negativos, se puede seleccionar como valor de correlación el menor de entre todos los calculados en el proceso de Jackknife.

$$\text{Correlacion} = \min\{\hat{r}_1, \hat{r}_2, \dots, \hat{r}_n\}$$

A pesar de que el método de Jackknife permite aumentar la robustez de la correlación de Pearson, si los outliers son muy extremos su influencia seguirá siendo notable.

2.4.1 Ejemplo

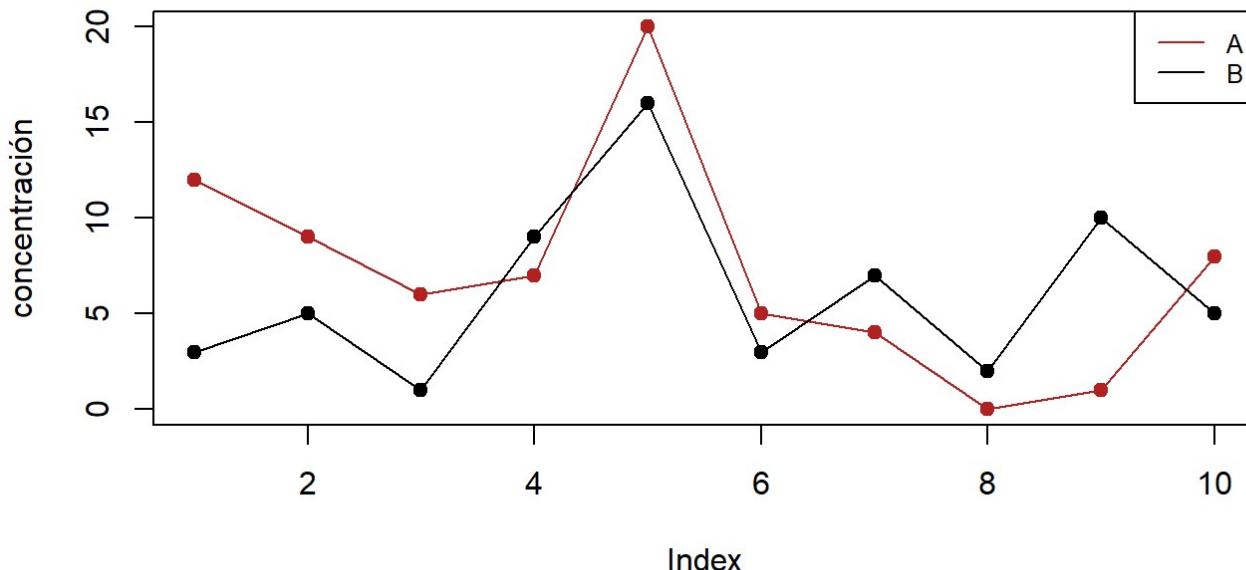
Un equipo de investigadores quiere estudiar si existe correlación en la presencia de dos sustancias (A y B) en el agua de los ríos. Para ello han realizado una serie de mediciones en las que se cuantifica la concentración de las dos sustancias en 10 muestras independientes de agua. Se sospecha que el instrumento de lectura sufre alguna avería que provoca que algunas lecturas se disparen, por esta razón se quiere emplear un método de correlación robusto. El objetivo de este ejemplo es ilustrar el método de Jackknife, por lo que se asume que se cumplen las condiciones para la correlación de Pearson.

[Hide](#)

```
# Datos simulados de dos variables A y B
a <- c(12, 9, 6, 7, 2, 5, 4, 0, 1, 8)
b <- c(3, 5, 1, 9, 5, 3, 7, 2, 10, 5)

# Se introduce un outlier
a[5] <- 20
b[5] <- 16
datos <- data.frame(a,b)
plot(datos$a, type = "o", lty = 1, pch = 19, col = "firebrick",
      ylab = "concentración", main = "Con outlier")
lines(datos$b, type = "o", pch = 19, lty = 1)
legend("topright", legend = c("A", "B"),
       col = c("firebrick", "black"), lty = c(1,1), cex = 0.8)
```

Con outlier


[Hide](#)

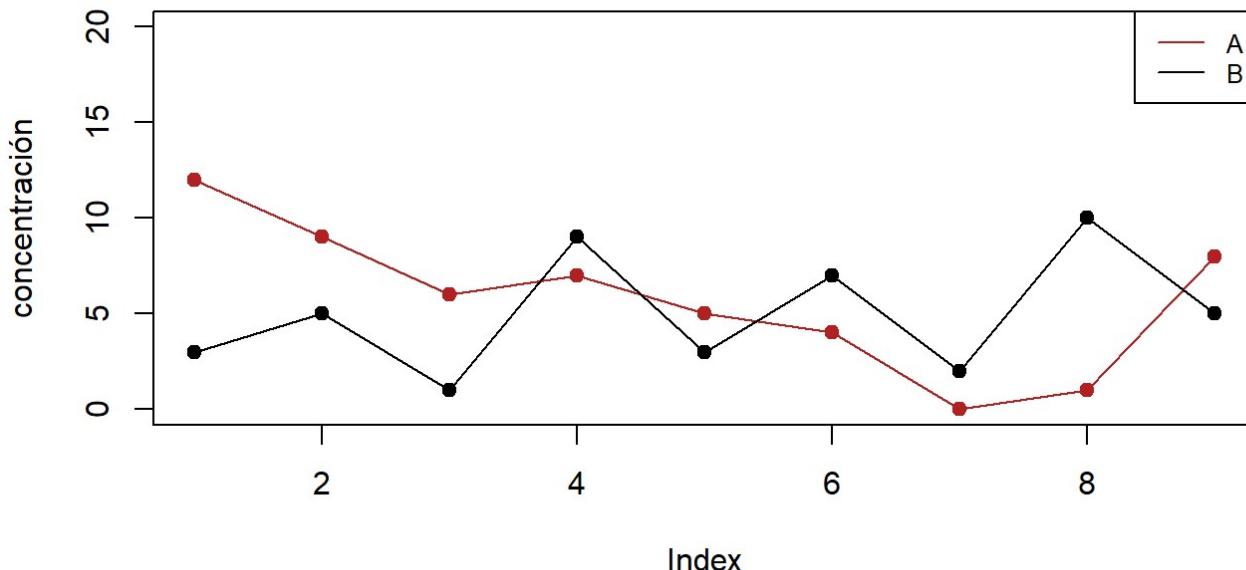
```
cor(datos$a, datos$b, method = "pearson")
```

```
## [1] 0.5249277
```

Hide

```
# Se elimina el outlier
a <- a[-5]
b <- b[-5]
datos_sin_outlier <- data.frame(a,b)
plot(datos_sin_outlier$a, type = "o", pch = 19, col = "firebrick", ylim = c(0,20),
      ylab = "concentración", main = "Sin outlier")
lines(datos_sin_outlier$b, type = "o", pch = 19, lty = 1)
legend("topright", legend = c("A", "B"), col = c("firebrick", "black"),
       lty = c(1,1), cex = 0.8)
```

Sin outlier



Hide

```
cor(datos_sin_outlier$a, datos_sin_outlier$b, method = "pearson")
```

```
## [1] -0.1790631
```

La observación numero 5 tiene una gran influencia en el resultado de la correlación, siendo de 0.52 en su presencia y de -0.18 si se excluye.

Hide

```
# FUNCIÓN PARA APLICAR JACKKNIFE A LA CORRELACIÓN DE PEARSON
correlacion_jackknife <- function(matriz, method = "pearson") {
  n <- nrow(matriz) # número de observaciones
  valores_jackknife <- rep(NA, n)

  for (i in 1:n) {
    # Loop para excluir cada observación y calcular la correlación
    valores_jackknife[i] <- cor(matriz[-i, 1], matriz[-i, 2], method = method)
  }

  promedio_jackknife <- mean(valores_jackknife)
  standar_error <- sqrt(((n - 1)/n)*sum((valores_jackknife-promedio_jackknife)^2))
  bias <- (n - 1) * (promedio_jackknife - cor(matriz[, 1], matriz[, 2],
                                                method = method))

  return(list(valores_jackknife = valores_jackknife,
                promedio = promedio_jackknife,
                se = standar_error,
                bias = bias))
}

correlacion <- correlacion_jackknife(datos)
correlacion$promedio
```

```
## [1] 0.4854695
```

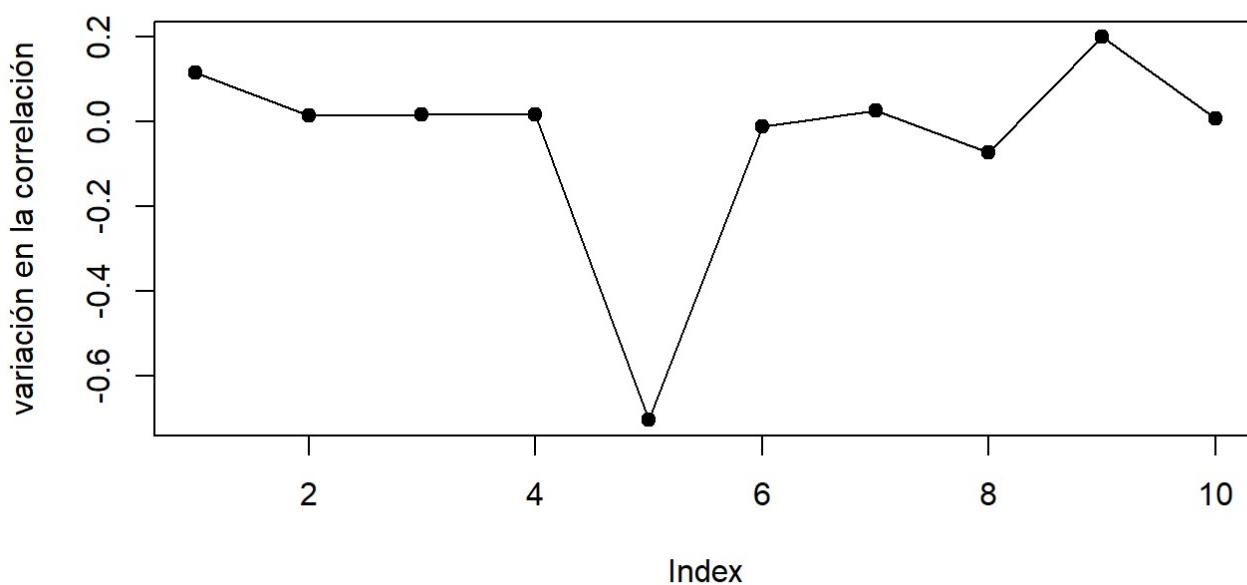
Hide

```
correlacion$valores_jackknife
```

```
## [1] 0.6409823 0.5394608 0.5410177 0.5414076 -0.1790631 0.5121559
## [7] 0.5504217 0.4528914 0.7237978 0.5316224
```

Hide

```
plot((correlacion$valores_jackknife - cor(datos$a, datos$b, method = "pearson")),
      type = "o", pch = 19, ylab = "variación en la correlación")
```



El método Jackknife correlation solo ha sido capaz de amortiguar una pequeña parte de la influencia del outlier, sin embargo, si ha permitido identificar qué observación está afectando en mayor medida.

2.5 Simple matching coefficient

Cuando las variables con las que se pretende determinar la similitud entre observaciones son de tipo binario, a pesar de que es posible codificarlas de forma numérica como 1 o 0, no tiene sentido aplicar operaciones aritméticas sobre ellas (media, suma...). Por ejemplo, si la variable sexo se codifica como 1 para mujer y 0 para hombre, carece de significado decir que la media de la variable sexo en un determinado set de datos es 0.5. En situaciones como esta, no se pueden emplear medidas de similitud basadas en distancia euclídea, manhattan, correlación...

Dado dos objetos A y B, cada uno con n atributos binarios, el simple matching coefficient (SMC) define la similitud entre ellos como:

$$SMC = \frac{\text{número coincidencias}}{\text{número total de atributos}} = \frac{M_{00} + M_{11}}{M_{00} + M_{01} + M_{10} + M_{11}}$$

donde M_{00} y M_{11} son el número de variables para las que ambas observaciones tienen el mismo valor (ambas 0 o ambas 1), y M_{01} y M_{10} el número de variables que no coinciden. El valor de distancia simple matching distance (SMD) se corresponde con $1 - SMC$.

2.6 Índice Jaccard

El índice Jaccard o coeficiente de correlación Jaccard es similar al simple matching coefficient (SMC). La diferencia radica en que el SMC tiene el término M_{00} en el numerador y denominador, mientras que el índice de Jaccard no. Esto significa que SMC considera como coincidencias tanto si el atributo está presente en ambos sets como si el atributo no está en ninguno de los sets, mientras que Jaccard solo cuenta como coincidencias cuando el atributo está presente en ambos sets.

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

o en términos matemáticos de sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

La distancia de Jaccard ($1 - J$) supera a la simple matching distance en aquellas situaciones en las que la coincidencia de ausencia no aporta información. Para ilustrar este hecho, supóngase que se quiere cuantificar la similitud entre dos clientes de un supermercado en base a los artículos comprados. Es de esperar que cada cliente solo adquiera unos pocos artículos de los muchos disponibles, por lo que el número de artículos no comprados por ninguno (M_{00}) será muy alto. Como la distancia de Jaccard ignora las coincidencias de tipo M_{00} , el grado de similitud dependerá únicamente de las coincidencias entre los artículos comprados.

2.7 Distancia coseno

El coseno del ángulo que forman dos vectores puede interpretarse como una medida de similitud de sus orientaciones, independientemente de sus magnitudes. Si dos vectores tienen exactamente la misma orientación (el ángulo que forman es 0°) su coseno toma el valor de 1, si son perpendiculares (forman un ángulo de 90°) su coseno es 0 y si tienen orientaciones opuestas (ángulo de 180°) su coseno es de -1.

$$\cos(\alpha) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

Si los vectores se normalizan restándoles la media ($\mathbf{X} - \bar{\mathbf{X}}$), la medida recibe el nombre de coseno centrado y es equivalente a la correlación de Pearson.

Hide

```
a <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6)
b <- c(5, 5.5, 4.8, 5.4, 4.7, 5.6, 5.3, 5.5, 5.2, 4.8)

# Correlación de Pearson
cor(x = a, y = b, method = "pearson")
```

```
## [1] 0.02427991
```

Hide

```
# Coseno
coseno <- function(x, y) {
  resultado <- x %*% y / (sqrt(x %*% x) * sqrt(y %*% y))
  return(as.numeric(resultado))
}

coseno(a, b)
```

```
## [1] 0.9802813
```

Hide

```
# Coseno tras centrar los vectores
a <- a - mean(a)
b <- b - mean(b)
coseno(a,b)
```

```
## [1] 0.02427991
```

2.8 Escala de las variables

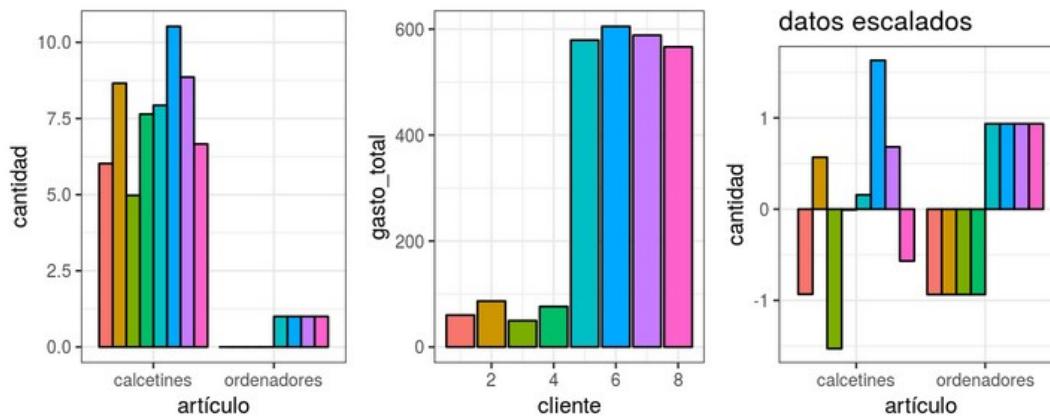
Al igual que en otros métodos estadísticos (PCA, ridge regression, lasso...), la escala en la que se miden las variables y la magnitud de su varianza pueden afectar en gran medida a los resultados obtenidos por clustering. Si una variable tiene una escala mucho mayor que el resto, determinará en gran medida el valor de distancia/similitud obtenido al comparar las observaciones, dirigiendo así la agrupación final. Escalar y centrar las variables de forma que todas ellas tengan media 0 y desviación estándar 1 antes de calcular la matriz de distancias asegura que todas las variables tengan el mismo peso cuando se realice el clustering. Sebastian Raschka hace un análisis muy explicativo de los distintos tipos de escalado y normalización.

$$\frac{x_i - \text{mean}(x)}{\text{sd}(x)}$$

Para ilustrar este hecho, supóngase que una tienda online quiere clasificar a los compradores en función de los artículos que adquieren, por ejemplo, calcetines y ordenadores. La siguiente imagen muestra el número de artículos comprados por 8 clientes a lo largo de un año, junto con el gasto total de cada uno.

Hide

```
knitr:::include_graphics("images/cliente.png")
```



Si se intenta agrupar a los clientes por el número de artículos comprados, dado que los calcetines se compran con mucha más frecuencia que los ordenadores, van a tener más peso al crear los clusters. Por el contrario, si la agrupación se hace en base al gasto total de los clientes, como los ordenadores son mucho más caros, van a determinar en gran medida la clasificación. Escalando y centrándolas las variables se consigue igualar la influencia de calcetines y ordenadores.

Cabe destacar que, si se aplica la estandarización descrita, existe una relación entre la distancia euclídea y la

correlación de Pearson que hace que los resultados obtenidos por clustering en ambos casos sean equivalentes.

$$d_{euc}(p, q \text{ estandarizados}) = \sqrt{2n(1 - cor(p, q))}$$

Ejemplo

El set de datos `USArrests` contiene información sobre el número de delitos (asaltos, asesinatos y secuestros) junto con el porcentaje de población urbana para cada uno de los 50 estados de USA. Empleando estas variables se pretende calcular una matriz de distancias que permita identificar los estados más similares

Dos de las funciones en R que permiten calcular matrices de distancia empleando variables numéricas son `dist()` y `get_dist()`. Esta última incluye más tipos de distancias.

[Hide](#)

```
data(USArrests)

# Se escalan las variables
datos <- scale(USArrests, center = TRUE, scale = TRUE)

# Distancia euclídea
mat_dist <- dist(x = datos, method = "euclidean")
round(as.matrix(mat_dist)[1:5, 1:5], 2)
```

	Alabama	Alaska	Arizona	Arkansas	California
## Alabama	0.00	2.70	2.29	1.29	3.26
## Alaska	2.70	0.00	2.70	2.83	3.01
## Arizona	2.29	2.70	0.00	2.72	1.31
## Arkansas	1.29	2.83	2.72	0.00	3.76
## California	3.26	3.01	1.31	3.76	0.00

[Hide](#)

```
# Distancia basada en la correlación de pearson
library(factoextra)
mat_dist <- get_dist(x = datos, method = "pearson")
round(as.matrix(mat_dist)[1:5, 1:5], 2)
```

	Alabama	Alaska	Arizona	Arkansas	California
## Alabama	0.00	0.71	1.45	0.09	1.87
## Alaska	0.71	0.00	0.83	0.37	0.81
## Arizona	1.45	0.83	0.00	1.18	0.29
## Arkansas	0.09	0.37	1.18	0.00	1.59
## California	1.87	0.81	0.29	1.59	0.00

[Hide](#)

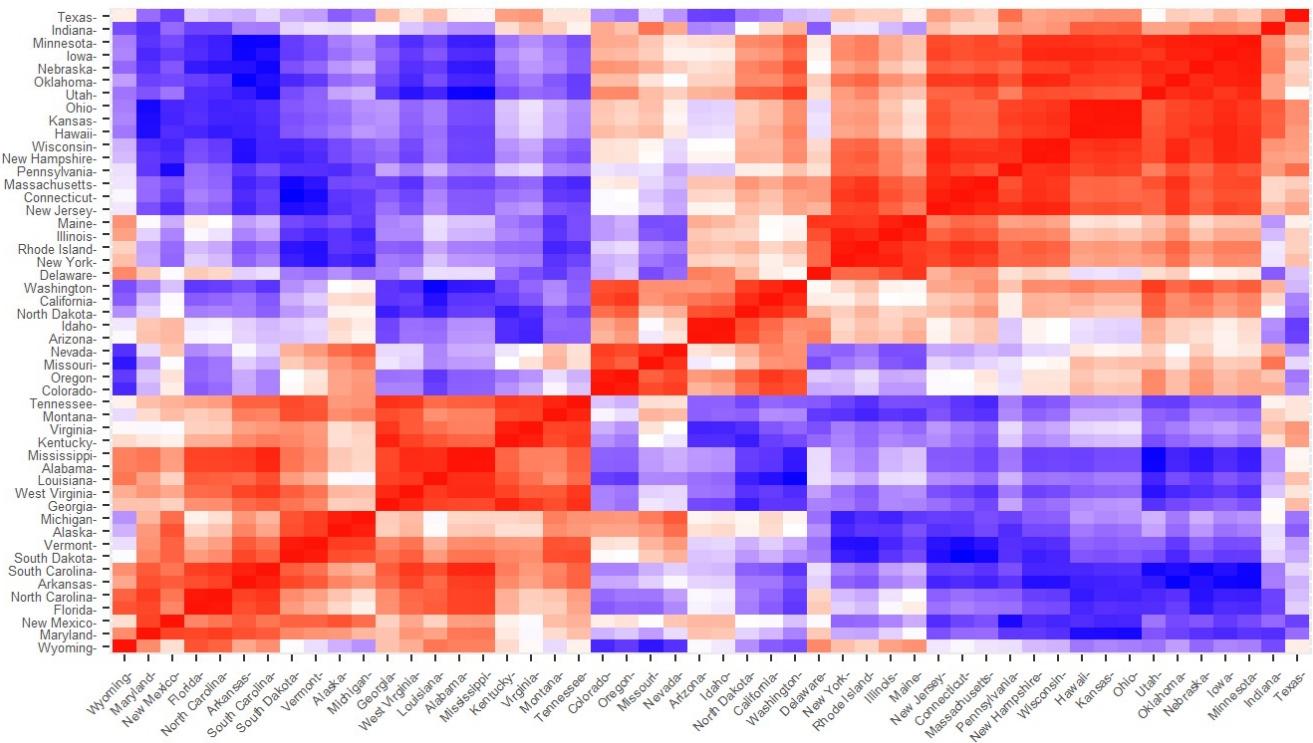
```
# Esto es equivalente a 1 - correlación pearson
round(1 - cor(x = t(datos), method = "pearson"), 2)[1:5, 1:5]
```

#	Alabama	Alaska	Arizona	Arkansas	California
## Alabama	0.00	0.71	1.45	0.09	1.87
## Alaska	0.71	0.00	0.83	0.37	0.81
## Arizona	1.45	0.83	0.00	1.18	0.29
## Arkansas	0.09	0.37	1.18	0.00	1.59
## California	1.87	0.81	0.29	1.59	0.00

La función `fviz_dist` del paquete `factoextra` resulta muy útil para generar representaciones gráficas (heatmap) de matrices de distancia.

Hide

```
fviz_dist(dist.obj = mat_dist, lab_size = 5) +
  theme(legend.position = "none")
```



3 K-means

Estos métodos se conocen comúnmente como agrupamiento. La agrupación k -means es uno de los algoritmos de agrupación más utilizados para dividir las observaciones en un conjunto de k grupos (es decir, k agrupaciones), donde k es preespecificado por el analista.

k -means, al igual que otros algoritmos de agrupación, intenta clasificar las observaciones en grupos (o agrupaciones) mutuamente excluyentes, de modo que las observaciones dentro del mismo grupo sean lo más similares posible (es decir, alta similitud intraclasa), mientras que las observaciones de diferentes grupos son lo más diferente posible (es decir, baja similitud entre clases). En la agrupación de k -medias, cada grupo está representado por su centro (es decir, centroide) que corresponde a la media de los valores de observación asignados al grupo.

3.1 Definiendo el cluster (agrupamiento)

- La idea básica detrás de la agrupación de k -means es construir agrupaciones de modo que se minimice la variación total dentro de la agrupación. Hay varios algoritmos de k -means disponibles para hacer esto. El algoritmo estándar es el algoritmo de Hartigan-Wong (Hartigan y Wong 1979,), que define la variación total dentro del clúster como la suma de las distancias euclidianas entre los valores de característica de la observación i y el centroide correspondiente:

$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2,$$

donde

- x_i es una observación que pertenece al grupo C_k ;
- μ_k es el valor medio de los puntos asignados al cluster C_k .

Cada observación (x_i) es asignado a un grupo dado tal que la suma de las distancias al cuadrado (SS) de cada observación a sus centros de grupo asignados μ_k es minimizado.

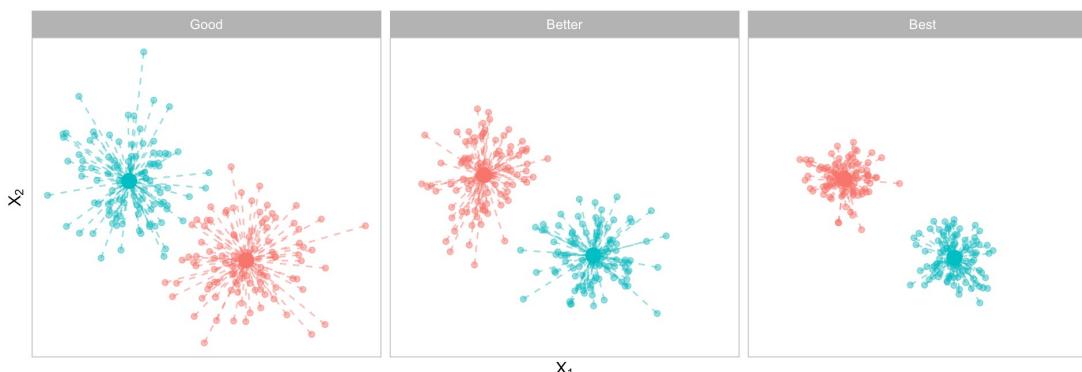
Definimos la variación total dentro del clúster de la siguiente manera:

$$SS_{within} = \sum_{k=1}^k W(C_k) = \sum_{k=1}^k \sum_{x_i \in C_k} (x_i - \mu_k)^2 \quad (20.2)$$

La medida SS_{within} mide la compacidad (es decir, la bondad) de los grupos resultantes y queremos que sea lo más pequeño posible como se ilustra en la Figura siguiente:

Hide

```
knitr:::include_graphics("images/kmeans-clusters-good-better-best-1.png")
```



Hide

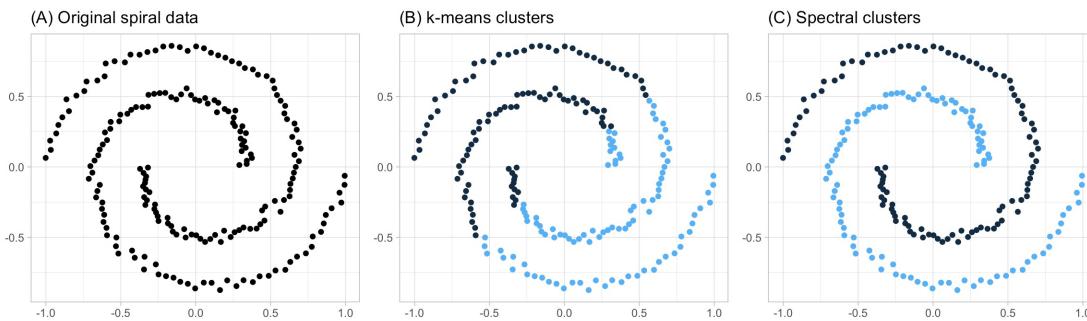
#La variación total dentro del grupo captura las distancias totales entre el centroide de un grupo y las observaciones individuales asignadas a ese grupo. Cuanto más compactas sean estas distancias, más definidos y aislados estarán los grupos.

Los supuestos subyacentes de k -means requieren que los puntos estén más cerca de su propio centro de clúster que de otros. Esta suposición puede ser ineficaz cuando los grupos tienen geometrías complicadas ya que k -means requiere límites convexos. Por ejemplo, considere los datos en la Figura (A) siguiente. Estos datos están claramente agrupados; sin embargo, sus agrupaciones no tienen lindos límites convексos. En

consecuencia, la agrupación de k -medias no captura los grupos apropiados como ilustra la parte (B) de la figura siguiente. Sin embargo, los métodos de agrupamiento espectral aplican el mismo kernel que el SVM para permitir que el k -means descubra límites no convexos (Figura siguiente parte (C)). Vea J. Friedman, Hastie y Tibshirani (2001) (<https://web.stanford.edu/~hastie/Papers/ESLII.pdf>) para una discusión exhaustiva sobre la agrupación espectral y el paquete `kernlab` para una implementación en R.

Hide

```
knitr:::include_graphics("images/non-linear-boundaries-1.png")
```



Hide

#Los supuestos de k -means le prestan ineeficacia para capturar agrupaciones geométricas complejas; sin embargo, la agrupación espectral le permite agrupar datos que están conectados pero no necesariamente agrupados dentro de límites convexas.

3.2 Algoritmo de k -medias

El primer paso cuando se utiliza la agrupación de k -medias es indicar el número de grupos (k) que se generarán en la solución final. Desafortunadamente, a menos que nuestro conjunto de datos sea muy pequeño, no podemos evaluar todas las combinaciones posibles de conglomerados porque hay casi k^n formas de dividir n observaciones en k conglomerados. En consecuencia, necesitamos estimar una solución local óptima para nuestro k específico (Hartigan y Wong 1979). Para hacerlo, el algoritmo comienza seleccionando aleatoriamente k observaciones del conjunto de datos para que sirvan como centros iniciales para los grupos (es decir, centroides). A continuación, cada una de las observaciones restantes se asigna a su centroide más cercano, donde se define el más cercano utilizando la distancia entre el objeto y la media del grupo (en función de la medida de distancia seleccionada). Esto se llama el paso de asignación del clúster.

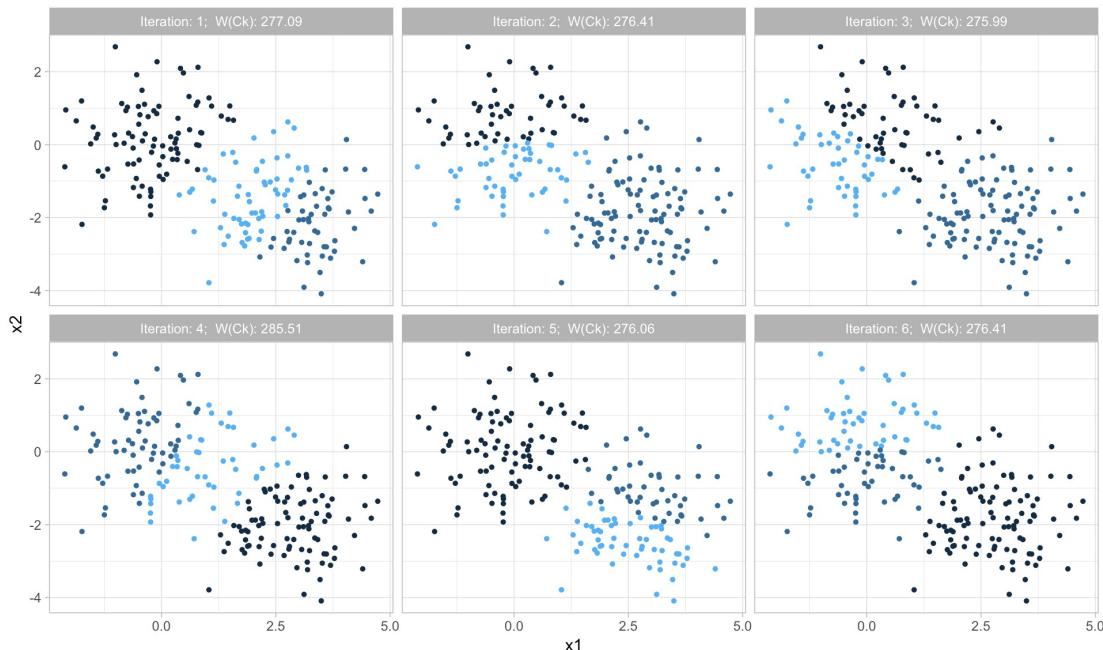
A continuación, el algoritmo calcula el nuevo centro (es decir, el valor medio) de cada grupo. La actualización del centroide se usa para definir este paso. Ahora que los centros han sido recalculados, cada observación se verifica nuevamente para ver si podría estar más cerca de un grupo diferente. Todos los objetos se reasignan nuevamente utilizando las medias del clúster actualizados. Los pasos de asignación del clúster y actualización del centroide se repiten iterativamente hasta que las asignaciones al clúster dejan de cambiar (es decir, cuando se logra la convergencia). Es decir, los grupos formados en la iteración actual son los mismos que los obtenidos en la iteración anterior.

Debido a la aleatorización de las observaciones iniciales de k utilizadas como centroides iniciales, podemos obtener resultados ligeramente diferentes cada vez que aplicamos el procedimiento. En consecuencia, la mayoría de los algoritmos usan varios inicios aleatorios y eligen la iteración con la menor $W(C_k)$. La siguiente figura ilustra la variación de $W(C_k)$ para diferentes inicios aleatorios.

Una buena regla para la cantidad de inicios aleatorios para aplicar es de 10 a 20.

Hide

```
knitr:::include_graphics("images/random-starts-1.png")
```



Hide

#Cada aplicación del algoritmo k -means puede lograr ligeras diferencias en los resultados finales basados en el inicio aleatorio.

El algoritmo k -means se puede resumir de la siguiente manera:

1. Especifique el número de grupos (k) que se crearán (esto lo hace el analista).
2. Seleccione k observaciones al azar del conjunto de datos para usar como centroides de conglomerado iniciales.
3. Asigne cada observación a su centroide más cercano en función de la medida de distancia seleccionada.
4. Para cada uno de los grupos de k , actualice el centroide del grupo calculando los nuevos valores medios de todos los puntos de datos en el grupo. El centroide para el i -ésimo grupo es un vector de longitud p que contiene las medias de todas las características de p para las observaciones en el grupo i .
5. Minimizar iterativamente SS_{within} . Es decir, se repite los pasos 3–4 hasta que las asignaciones del clúster dejen de cambiar (más allá de algún umbral) o se alcance el número máximo de iteraciones. Una buena regla general es realizar 10–20 iteraciones.

Hide

```
knitr:::include_graphics("images/cluster.gif")
```



3.3 Ejemplo

Supongamos que queremos agrupar a los visitantes en un sitio web utilizando solo su edad (espacio unidimensional) de la siguiente manera ($n = 19$):

15, 15, 16, 19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65

- *Grupos (cluster) iniciales (centroide aleatorio o promedio):*

Tenemos $k = 2$, $c_1 = 16$, $c_2 = 22$. Distancia 1 = $\sqrt{(x_i - c_1)^2}$ y Distancia 2 = $\sqrt{(x_i - c_2)^2}$.

Hide

```
knitr:::include_graphics("images/kmeans1.JPG")
```

Iteration 1:

$$c_1 = 15.33$$

$$c_2 = 36.25$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	15.33
15	16	22	1	7	1	
16	16	22	0	6	1	
19	16	22	9	3	2	36.25
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

```
knitr:::include_graphics("images/kmeans2.JPG")
```

Iteration 2:

$$c_1 = 18.56$$

$$c_2 = 45.90$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	18.56
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	
19	15.33	36.25	3.67	17.25	1	
19	15.33	36.25	3.67	17.25	1	
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	45.9
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

```
knitr:::include_graphics("images/kmeans3.JPG")
```

Iteration 3:

$$c_1 = 19.50$$

$$c_2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	19.50
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	
20	18.56	45.9	1.44	25.9	1	
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	47.89
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	
43	18.56	45.9	24.44	2.9	2	
44	18.56	45.9	25.44	1.9	2	
60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

```
knitr:::include_graphics("images/kmeans4.JPG")
```

Iteration 4:

$$c_1 = 19.50$$

$$c_2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	19.50
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	
19	19.5	47.89	0.50	28.89	1	
19	19.5	47.89	0.50	28.89	1	
20	19.5	47.89	0.50	27.89	1	
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	47.89
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

No se ha observado ningún cambio entre las iteraciones 3 y 4. Al usar agrupamiento, se identificaron 2 grupos 15-28 y 35-65. La elección inicial de centroides puede afectar a los clústeres de salida, por lo que el algoritmo a menudo se ejecuta varias veces con diferentes condiciones de inicio para obtener una visión justa de lo que

deberían ser los conglomerados.

Veamos una animación [<http://www.saedsayad.com/flash/Kmeans.html> (<http://www.saedsayad.com/flash/Kmeans.html>)]

3.4 Ejemplos

3.4.1 Ejemplo 1

Echemos un vistazo a un ejemplo en R usando el conjunto de datos **Chatterjee-Price Attitude** del paquete `datasets`. El conjunto de datos consiste en una encuesta de empleados administrativos de una gran organización financiera. Los datos se agregan a partir de cuestionarios de aproximadamente 35 empleados para cada uno de los 30 departamentos (seleccionados al azar). Las cifras dan el porcentaje de respuestas favorables a siete preguntas en cada departamento.

[Hide](#)

```
# Cargar la librería
# library(datasets)

# Inspeccionando la estructura de los datos
str(attitude)
```

```
## 'data.frame':    30 obs. of  7 variables:
## $ rating      : num  43 63 71 61 81 43 58 71 72 67 ...
## $ complaints: num  51 64 70 63 78 55 67 75 82 61 ...
## $ privileges: num  30 51 68 45 56 49 42 50 72 45 ...
## $ learning    : num  39 54 69 47 66 44 56 55 67 47 ...
## $ raises      : num  61 63 76 54 71 54 66 70 71 62 ...
## $ critical    : num  92 73 86 84 83 49 68 66 83 80 ...
## $ advance     : num  45 47 48 35 47 34 35 41 31 41 ...
```

[Hide](#)

```
# Summarise data
summary(attitude)
```

```

##      rating      complaints      privileges      learning
## Min.   :40.00   Min.   :37.0   Min.   :30.00   Min.   :34.00
## 1st Qu.:58.75  1st Qu.:58.5  1st Qu.:45.00  1st Qu.:47.00
## Median :65.50  Median :65.0  Median :51.50  Median :56.50
## Mean    :64.63  Mean    :66.6  Mean    :53.13  Mean    :56.37
## 3rd Qu.:71.75  3rd Qu.:77.0  3rd Qu.:62.50  3rd Qu.:66.75
## Max.    :85.00  Max.    :90.0  Max.    :83.00  Max.    :75.00
##      raises      critical      advance
## Min.   :43.00   Min.   :49.00  Min.   :25.00
## 1st Qu.:58.25  1st Qu.:69.25  1st Qu.:35.00
## Median :63.50  Median :77.50  Median :41.00
## Mean    :64.63  Mean    :74.77  Mean    :42.93
## 3rd Qu.:71.00  3rd Qu.:80.00  3rd Qu.:47.75
## Max.    :88.00  Max.    :92.00  Max.    :72.00

```

Como hemos visto, estos datos proporcionan el porcentaje de respuestas favorables para cada departamento. Por ejemplo, en el resultado del resumen anterior, podemos ver que para la variable **privileges** entre los 30 departamentos, el porcentaje mínimo de respuestas favorables fue de 30 y el máximo fue de 83. En otras palabras, un departamento tuvo solo un 30% de respuestas favorables cuando llegó para evaluar los “privileges” y un departamento tuvo un 83% de respuestas favorables a la hora de evaluar los “privilegios” y muchos otros niveles favorables de respuesta intermedios.

A la luz del ejemplo, tomaremos un subconjunto del conjunto de datos de actitud y consideraremos solo dos variables en nuestro ejercicio de agrupación usando *k*-Means. Imaginemos que nos gustaría agrupar el conjunto de datos de actitud con las respuestas de los 30 departamentos cuando se trata de “privileges” y “learning”, y nos gustaría entender si hay puntos en común entre ciertos departamentos cuando se trata de estas dos variables.

Hide

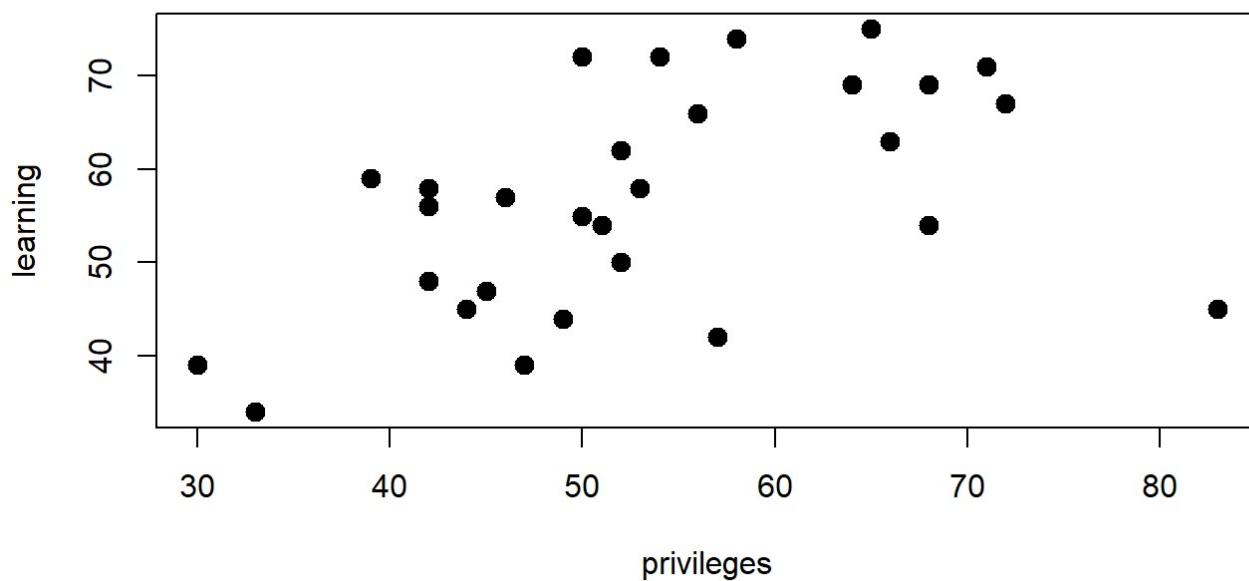
```

# Subconjuntos de los datos attitude
dat = attitude[,c(3,4)]

# Grafica del subconjunto de datos
plot(dat, main = "% de respuestas favorables de Learning (aprendizaje) y Privilege (privilegio)", pch =20, cex =2)

```

% de respuestas favorables de Learning (aprendizaje) y Privilege (privilegio)



Con el subconjunto de datos y el gráfico anterior, podemos ver cómo la puntuación de cada departamento se comporta según las variables Privilegio y Aprendizaje comparados entre si. En el sentido más simple, podemos aplicar clusters de k -Means a este conjunto de datos e intentar asignar cada departamento a un número específico de clusters que sean "similares".

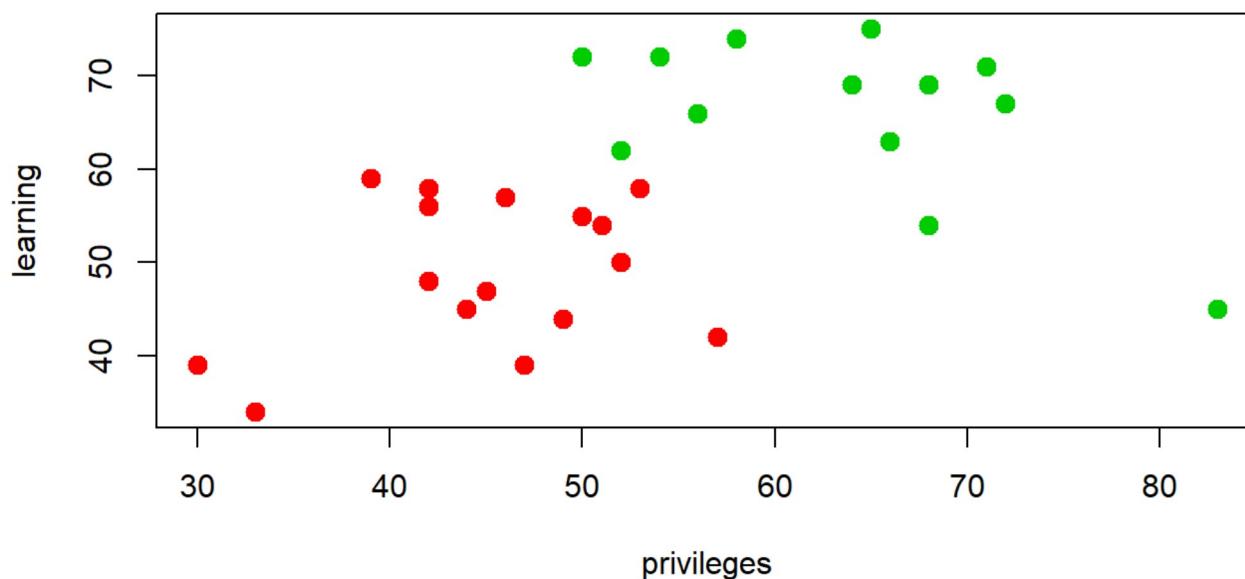
Usemos la función `kmeans` del paquete de base del R:

Hide

```
# Realizando k-Means con 2 clusters
set.seed(7)
km1 = kmeans(dat, 2, nstart=100)

# Graficando los resultados
windows()
plot(dat, col =(km1$cluster +1) , main="k-Means con 2 clusters", pch=20, cex=2)
```

k-Means con 2 clusters



Como se mencionó anteriormente, una de las decisiones clave que se deben tomar al realizar el agrupamiento de *k*-Means es decidir el número de clústeres que se utilizarán. En la práctica, no hay una respuesta fácil y es importante probar diferentes formas y números de clusters para decidir que opciones es la solución más útil, aplicable o interpretable.

En la figura anterior, elegimos aleatoriamente el número de clústeres para que sean 2 solo con fines ilustrativos.

Sin embargo, una solución que se usa a menudo para identificar el número óptimo de conglomerados que se denomina método del Elbow (codo) e implica la observación de un conjunto de números posibles de conglomerados en relación a la forma como ellos minimizan la suma de cuadrados dentro del cluster. En otras palabras, el método Elbow examina la desemejanza dentro del clúster como una función del número de clusters. A continuación hay una representación visual del método:

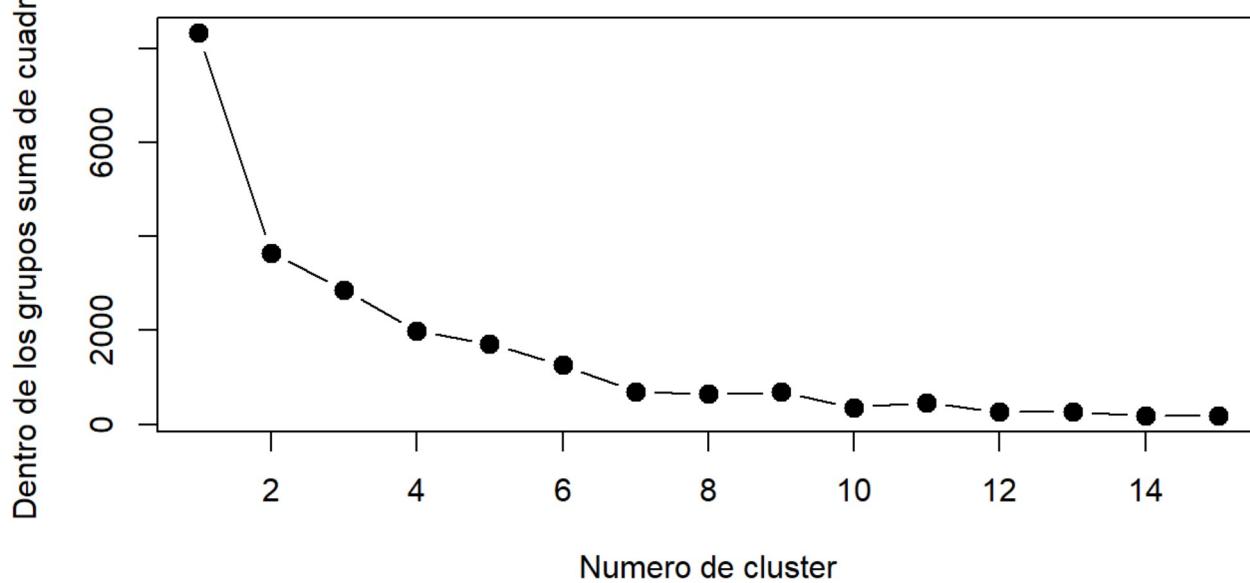
Hide

```
#Verificar la cantidad óptima de clusters dados los datos

mydata <- dat
wss      <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                         centers=i)$withinss)

windows()
plot(1:15, wss, type="b", xlab="Numero de cluster",
      ylab="Dentro de los grupos suma de cuadrados",
      main="Evaluar el numero optimo de conglomerados con el metodo del codo",
      pch=20, cex=2)
```

Evaluar el numero optimo de conglomerados con el metodo del codo



Con el método de codo, el valor del criterio de solución (dentro de los grupos suma de cuadrados) tenderá a disminuir sustancialmente con cada aumento sucesivo en la cantidad de conglomerados. De forma simplista, se identifica una cantidad óptima de conglomerados una vez que se observa un “pliegue” en el trazado de la línea. Como puede comprender, identificar el punto en el que existe un “pliegue” no es un enfoque muy objetivo y es muy propenso a los procesos heurísticos.

Pero del ejemplo anterior, podemos decir que después de 6 conglomerados la diferencia observada en la desemejanza dentro del conglomerado no es sustancial. En consecuencia, podemos decir con cierta confianza razonable que la cantidad óptima de conglomerados que se utilizarán es 6.

Suponiendo que esta afirmación es válida, podemos continuar y aplicar el número identificado de clusters en el algoritmo de k -Means y trazar los resultados:

Hide

```
# Realizar k-Means con la cantidad óptima de clústeres identificados a partir del método del Codo
set.seed(7)
km2 = kmeans(dat, 6, nstart=100)

# Examine el resultado del algoritmo de agrupamiento
km2
```

```
## K-means clustering with 6 clusters of sizes 4, 2, 2, 8, 6, 8
##
## Cluster means:
##   privileges learning
## 1    54.50000   71.000
## 2    75.50000   49.500
## 3    31.50000   36.500
## 4    46.87500   57.375
## 5    67.66667   69.000
## 6    47.62500   45.250
##
## Clustering vector:
## [1] 3 4 5 6 1 6 4 4 5 6 4 6 6 2 1 1 5 5 4 2 3 4 6 4 6 5 1 6 5 4
##
## Within cluster sum of squares by cluster:
## [1] 71.0000 153.0000 17.0000 244.7500 133.3333 255.3750
## (between_SS / total_SS =  89.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"       "totss"        "withinss"
## [5] "tot.withinss" "betweenss"     "size"         "iter"
## [9] "ifault"
```

[Hide](#)

```
km2$withinss #
```

```
## [1] 71.0000 153.0000 17.0000 244.7500 133.3333 255.3750
```

[Hide](#)

```
sum(km2$withinss)
```

```
## [1] 874.4583
```

[Hide](#)

```
km2$tot.withinss #Suma total de cuadrados dentro del clúster
```

```
## [1] 874.4583
```

[Hide](#)

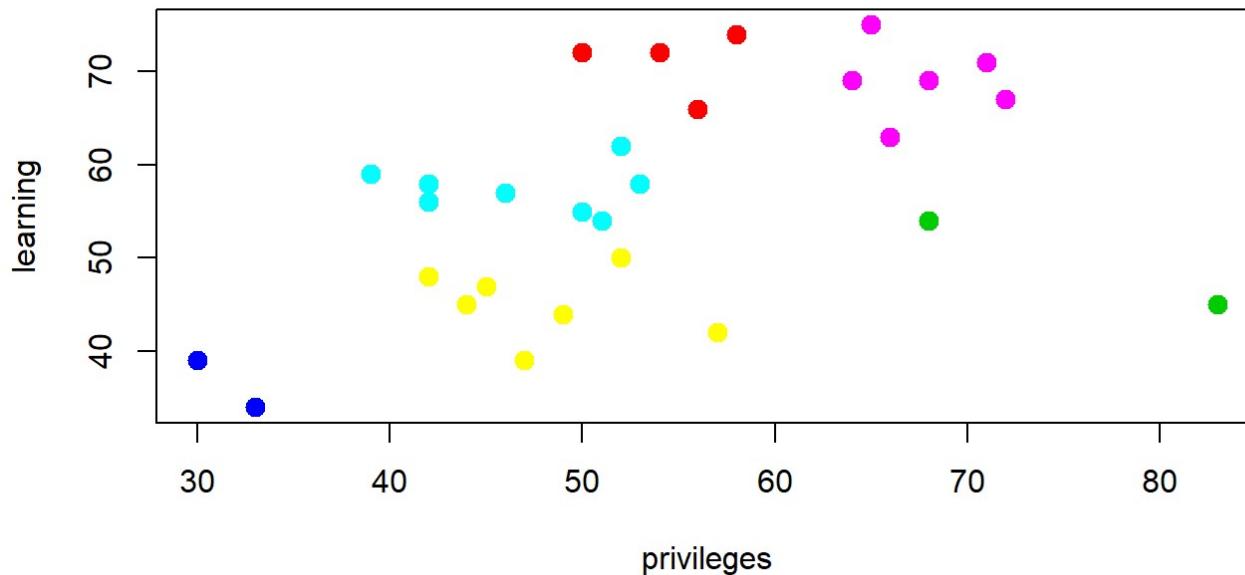
```
km2$size
```

```
## [1] 4 2 2 8 6 8
```

[Hide](#)

```
# Graficando los resultados  
plot(dat, col =(km2$cluster +1) , main="Resultados de k-Means con 6 clusters", pch=20,  
cex=2)
```

Resultados de k-Means con 6 clusters



A partir de los resultados anteriores podemos ver que hay un conjunto relativamente bien definido de grupos de departamentos que son relativamente distintos cuando se trata de responder favorablemente en torno a Privilegios y Aprendizaje en la encuesta. Es natural pensar en los próximos pasos a partir de este tipo de resultados. Uno podría comenzar a idear estrategias para comprender por qué ciertos departamentos califican estas dos medidas diferentes de la manera en que lo hacen y qué hacer al respecto.

3.4.2 Ejemplo 2 - Agrupando dígitos

Vamos a ilustrar un ejemplo mediante la agrupación de k -means en las variables de píxeles MNIST y ver si podemos identificar grupos únicos de dígitos sin usar la variable de respuesta. Aquí, declaramos $k = 10$ solo porque ya sabemos que hay 10 dígitos únicos representados en los datos. También usamos 10 comienzos aleatorios (`nstart=10`). El resultado de nuestro modelo contiene muchas de las métricas que ya hemos discutido, como la variación total dentro del clúster (`withinss`), suma total de cuadrados dentro del clúster (`tot.withinss`), el tamaño de cada grupo (`clusts`), y la iteración de nuestros 10 inicios aleatorios utilizados (`iter`). También incluye el `cluster` al que se asigna cada observación y los `centros` de cada grupo.

[Hide](#)

```
# Helper packages
library(dplyr)      # para manipulación de datos
library(ggplot2)      # para visualización de datos
library(stringr)      # para funcionalidad de cadena

# Paquetes para los cluster
library(cluster)     # para algoritmos generales de agrupamiento
library(factoextra)   # para visualizar resultados de clúster
```

[Hide](#)

```
#Vamos a cargar el conjunto de datos MNIST
mnist <- dslabs::read_mnist()
```

[Hide](#)

```
features <- mnist$train$images

# Use el modelo k-means con 10 centros y 10 inicio aleatorios
gc() #para evitar problemas de memoria
```

```
##          used   (Mb) gc trigger   (Mb) max used   (Mb)
## Ncells  1015506  54.3    1991948 106.4  1991948 106.4
## Vcells 29283509 223.5   60710699 463.2 56779536 433.2
```

[Hide](#)

```
library(tictoc) #paquete para medir el tiempo y en mi computador (32gb de RAM y procesador i7) demora 3.2min
tic()
mnist_clustering <- kmeans(features, centers = 10, nstart = 10, iter.max=30000000)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 3000000)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 3000000)
```

[Hide](#)

```
toc()
```

```
## 201.69 sec elapsed
```

[Hide](#)

```
# Imprimir contenido de la salida del modelo
str(mnist_clustering)
```

```
## List of 9
## $ cluster      : int [1:60000] 2 4 10 6 1 7 8 2 8 1 ...
## $ centers       : num [1:10, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
## ..- attr(*, "dimnames")=List of 2
## ...$ : chr [1:10] "1" "2" "3" "4" ...
## ...$ : NULL
## $ totss         : num 2.06e+11
## $ withinss      : num [1:10] 2.05e+10 1.80e+10 9.24e+09 8.97e+09 1.58e+10 ...
## $ tot.withinss: num 1.53e+11
## $ betweenss     : num 5.27e+10
## $ size          : int [1:10] 8851 6506 3098 3192 5720 5613 4680 5972 7446 8922
## $ iter          : int 8
## $ ifault        : int 0
## - attr(*, "class")= chr "kmeans"
```

Notamos que los centros son una matriz de 10x784. Esta matriz contiene el valor promedio de cada una de las 784 características para los 10 grupos. Podemos graficar esto como es mostrado en la figura de abajo, que nos muestra cuál es el dígito típico en cada grupo. Claramente vemos dígitos reconocibles a pesar de que k -means no tenía una idea de la variable de respuesta.

Hide

```
# Extrayendo los centros de los clúster
mnist_centers <- mnist_clustering$centers

# Graficando los dígitos típicos del clúster
par(mfrow = c(2, 5), mar=c(0.5, 0.5, 0.5, 0.5))
layout(matrix(seq_len(nrow(mnist_centers)), 2, 5, byrow = FALSE))
for(i in seq_len(nrow(mnist_centers))) {
  image(matrix(mnist_centers[i, ], 28, 28) [, 28:1],
        col = gray.colors(12, rev = TRUE), xaxt="n", yaxt="n")
}
```

[Hide](#)

#Centros de grupos para los 10 grupos identificados en los datos de entrenamiento de M NIST

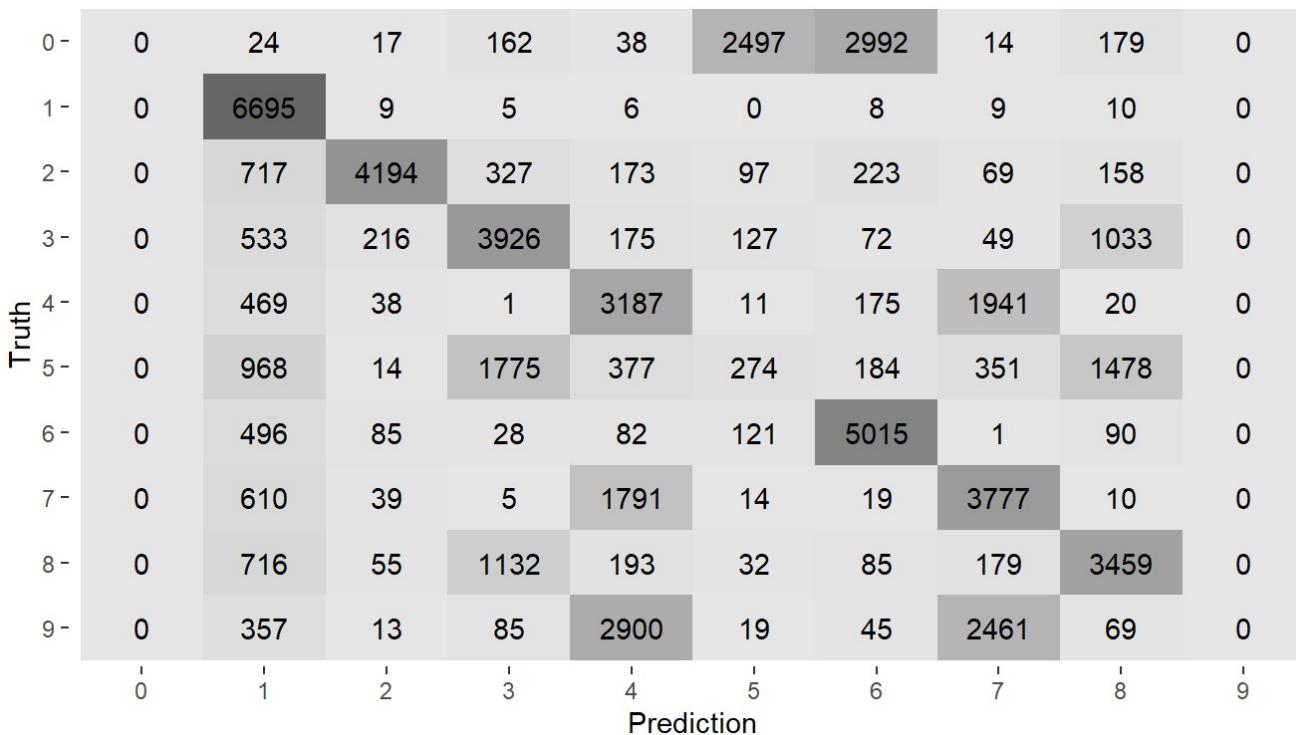
Podemos comparar los dígitos del clúster con las etiquetas de dígitos reales para ver qué tan bien está funcionando nuestro agrupamiento. Para hacerlo, comparamos el dígito más común en cada grupo (es decir, con la moda) con las etiquetas de entrenamiento reales. La Figura 20.6 ilustra los resultados. Vemos que k -means hace un trabajo decente al agrupar algunos de los dígitos. De hecho, la mayoría de los dígitos se agrupan más a menudo con dígitos similares que con dígitos diferentes.

[Hide](#)

```
# Creando la función moda
mode_fun <- function(x) {
  which.max(tabulate(x))
}

mnist_comparison <- data.frame(
  cluster = mnist_clustering$cluster,
  actual = mnist$train$labels
) %>%
  group_by(cluster) %>%
  mutate(mode = mode_fun(actual)) %>%
  ungroup() %>%
  mutate_all(factor, levels = 0:9)

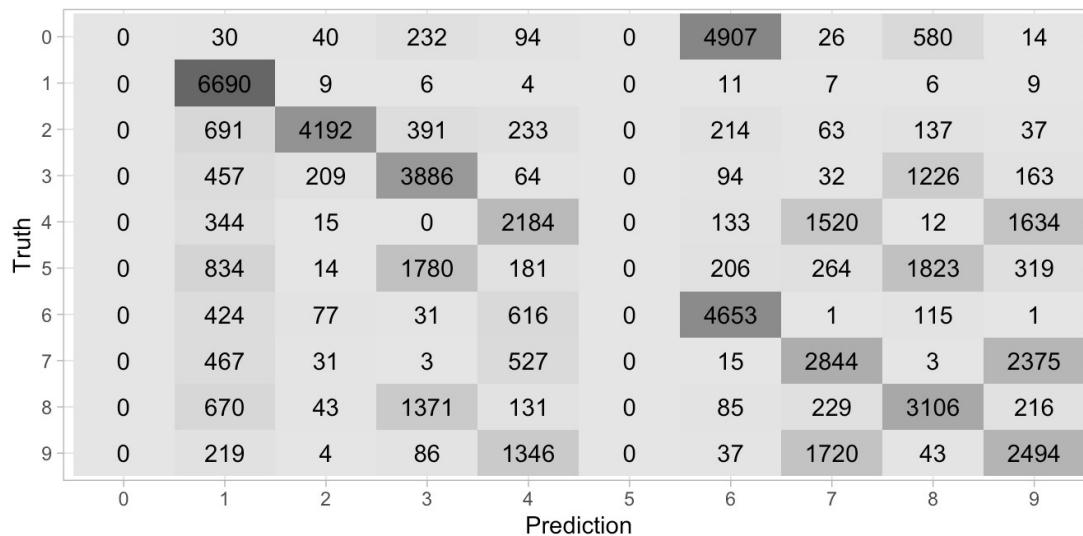
# Crear la matriz de confusión y resultados del plot
yardstick::conf_mat(
  mnist_comparison,
  truth = actual,
  estimate = mode
) %>%
  autoplot(type = 'heatmap')
```



Sin embargo, también vemos que algunos dígitos se agrupan a menudo con diferentes dígitos (por ejemplo, los 6 se agrupan a menudo con 0 y los 9 se agrupan a menudo con 7). También vemos que 0 y 5 nunca son el dígito dominante en un grupo. En consecuencia, nuestra agrupación está agrupando muchos dígitos que tienen cierta semejanza (3s, 5s y 8s a menudo se agrupan) y dado que esta es una tarea no supervisada, no existe un mecanismo para supervisar el algoritmo de lo contrario.

[Hide](#)

```
knitr:::include_graphics("images/mnist-clustering-confusion-matrix-1.png")
```



Hide

#Matriz de confusión que ilustra cómo el algoritmo k -medias agrupa los dígitos (eje x) y las etiquetas reales (eje y)

Elegir el número de grupos requiere un delicado equilibrio. Los valores más grandes de k pueden mejorar la homogeneidad de los grupos, sin embargo, se corre el riesgo de sobreajuste.

3.5 ¿Cuántos clúster?

Al agrupar los datos de MNIST, el número de grupos que especificamos se basó en el conocimiento previo de los datos. Sin embargo, a menudo no tenemos este tipo de información a priori y la razón por la que estamos realizando análisis de conglomerados es para identificar qué conglomerados pueden existir. Entonces, ¿cómo hacemos para determinar el número correcto de k ?

En el mejor de los casos (o quizás deberíamos decir el caso más fácil), k está predeterminado. Esto ocurre a menudo cuando tenemos recursos deterministas para asignar. Por ejemplo, una compañía puede emplear k vendedores y les gustaría dividir a sus clientes en uno de los k segmentos para que puedan ser asignados a uno de los vendedores. En este caso, k está predeterminado por recursos externos o conocimiento.

Un caso más común es que k es desconocido, sin embargo, a menudo todavía podemos aplicar el conocimiento a priori para agrupaciones potenciales. Por ejemplo, tal vez necesite agrupar las respuestas de la encuesta de experiencia del cliente para una empresa de ventas de automóviles. Puede comenzar estableciendo k en el número de marcas de automóviles que tiene la compañía. Si no tiene ningún conocimiento a priori para establecer k , entonces una regla general comúnmente utilizada es $k = \sqrt{n/2}$, donde n es el número de observaciones para agrupar. Sin embargo, esta regla puede dar como resultado valores muy grandes de k para conjuntos de datos más grandes. (por ejemplo, esto nos haría usar $k = 173$ para el conjunto de datos MNIST).

Cuando el objetivo del procedimiento de agrupamiento es determinar qué grupos distintos naturales existen en los datos, sin ningún conocimiento a priori, existen múltiples métodos estadísticos que podemos aplicar. Sin

embargo, muchas de estas medidas sufren la maldición de la dimensionalidad, ya que requieren múltiples iteraciones y la agrupación de grandes conjuntos de datos no es eficiente, especialmente cuando se agrupa en varias ocasiones.

- Vea Charrad et al. (2015) (<https://cran.r-project.org/web/packages/NbClust/index.html>) para una revisión exhaustiva de la gran variedad de medidas de rendimiento del clúster. El paquete NbClust implementa muchos de estos métodos, proporcionándole más de 30 índices para determinar el k óptima.

Uno de los métodos más populares es el método del codo. Recuerde que la idea básica detrás de los métodos de partición de conglomerados, como la agrupación de k -medias, es definir los conglomerados de modo que la variación total dentro del conglomerado se minimice (Ecuación (20.2)). La suma total de cuadrados dentro del clúster mide la compacidad del clúster y queremos que sea lo más pequeño posible. Por lo tanto, podemos usar el siguiente enfoque para definir los clústeres óptimos:

1. Calcule la agrupación de k -medias para diferentes valores de k . Por ejemplo, variando k de 1 a 20 grupos.
2. Para cada k , calcule la suma total de cuadrados dentro del clúster (WSS).
3. Grafique la curva de WSS de acuerdo con el número de grupos k .
4. La ubicación de una curva (es decir, el codo) en la gráfica generalmente se considera como un indicador del número apropiado de grupos.

Cuando se utilizan conjuntos de datos de tamaño pequeño a moderado, este proceso se puede realizar convenientemente con `factoextra::fviz_nbclust()`. Sin embargo, esta función requiere que especifique un valor máximo de k y entrenará modelos k -means para grupos de 1 – k . Cuando se trata de grandes conjuntos de datos, como MNIST, esto no es razonable, por lo que querrá implementar manualmente el procedimiento (por ejemplo, con un bucle `for` y especificar los valores de k para evaluar).

A continuación se evalúa la agrupación de los datos de `my_basket` de 1 a 25 grupos. El argumento `method = 'wss'` especifica que nuestros criterios de búsqueda están utilizando el método de codo discutido anteriormente y dado que estamos evaluando cantidades en diferentes canastas de productos, usamos la medida de distancia no paramétrica basada en correlación de Spearman. Los resultados muestran que el “codo” parece suceder cuando $k = 5$.

Hide

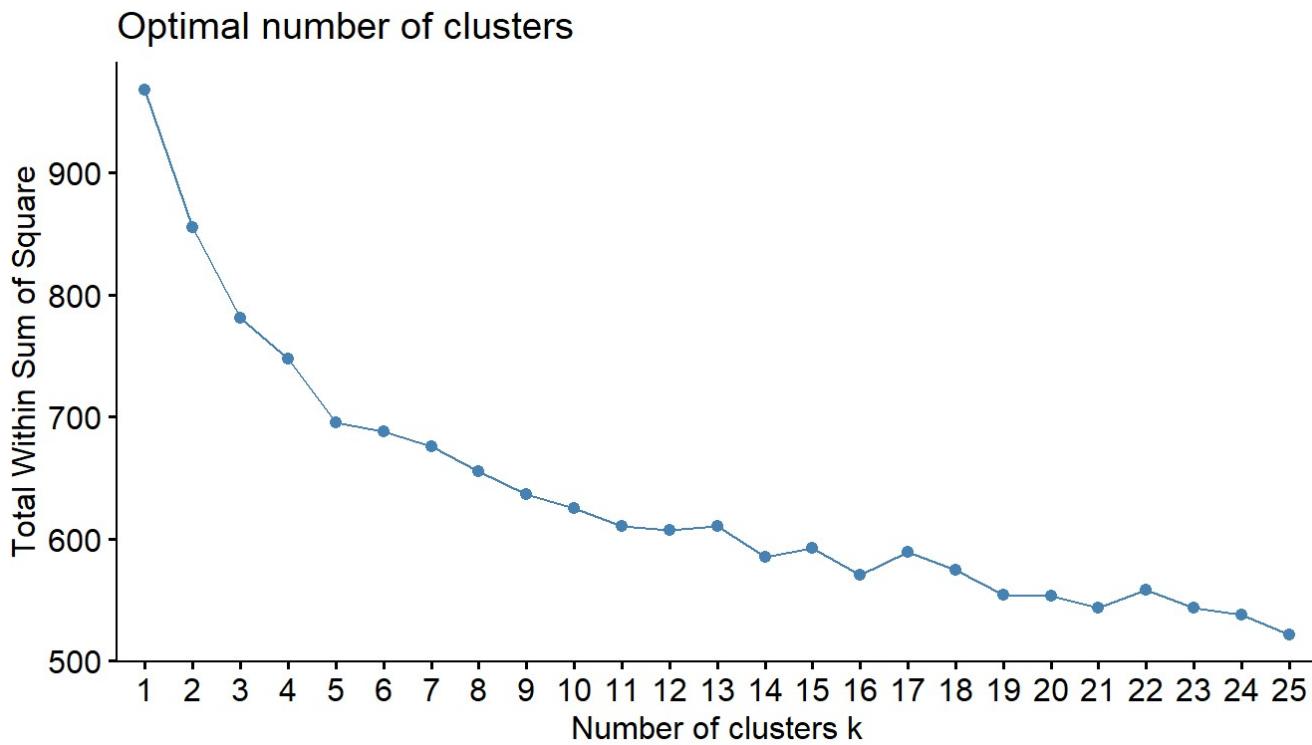
```
url      <- "https://koalaverse.github.io/homlr/data/my_basket.csv"
my_basket <- readr::read_csv(url)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

Hide

```
fviz_nbclust(  
  my_basket,  
  kmeans,  
  k.max = 25,  
  method = "wss",  
  diss = get_dist(my_basket, method = "spearman")  
)
```



Hide

#Usando el método del codo para identificar el número preferido de grupos en el conjunto de datos de mi cesta.

`fviz_nbclust()` también implementa otros métodos populares, como el método Silhouette (Rousseeuw 1987) y la estadística Gap (Tibshirani, Walther y Hastie 2001). Afortunadamente, las aplicaciones que requieren el conjunto óptimo exacto de grupos son bastante raras. En la mayoría de las aplicaciones, es suficiente elegir un k en función de la conveniencia en lugar de los estrictos requisitos de rendimiento. Pero si es necesario, el método del codo y otras métricas de rendimiento pueden orientarlo en la dirección correcta.

3.6 Agrupando con datos mixtos

Sabemos que la mayoría de los conjuntos de datos de la vida real contienen una mezcla de variables numéricas, categóricas y ordinales, y si una observación es similar a otra observación debería depender de estos atributos de tipo de datos. Hay algunas opciones para realizar la agrupación con datos mixtos y lo demostraremos en el conjunto completo de datos de alojamiento de Ames (menos la variable de respuesta `Sale_Price`). Para realizar la agrupación de k -medias en datos mixtos, podemos convertir cualquier variable categórica ordinal a numérica y codificar en caliente las restantes variables categóricas nominales.

Hide

```
# Full ames data set --> recodificar variables ordinales a numéricas
ames_full <- AmesHousing::make_ames() %>%
  mutate_if(str_detect(names(.), 'Qual|Cond|QC|Qu'), as.numeric)

# One-hot encode --> retener solo las características y no el precio de venta
full_rank <- caret::dummyVars(Sale_Price ~ ., data = ames_full,
                               fullRank = TRUE)
ames_1hot <- predict(full_rank, ames_full)

# Escalando los datos
ames_1hot_scaled <- scale(ames_1hot)

# Nueva dimensión
dim(ames_1hot_scaled)
```

```
## [1] 2930 240
```

Hide

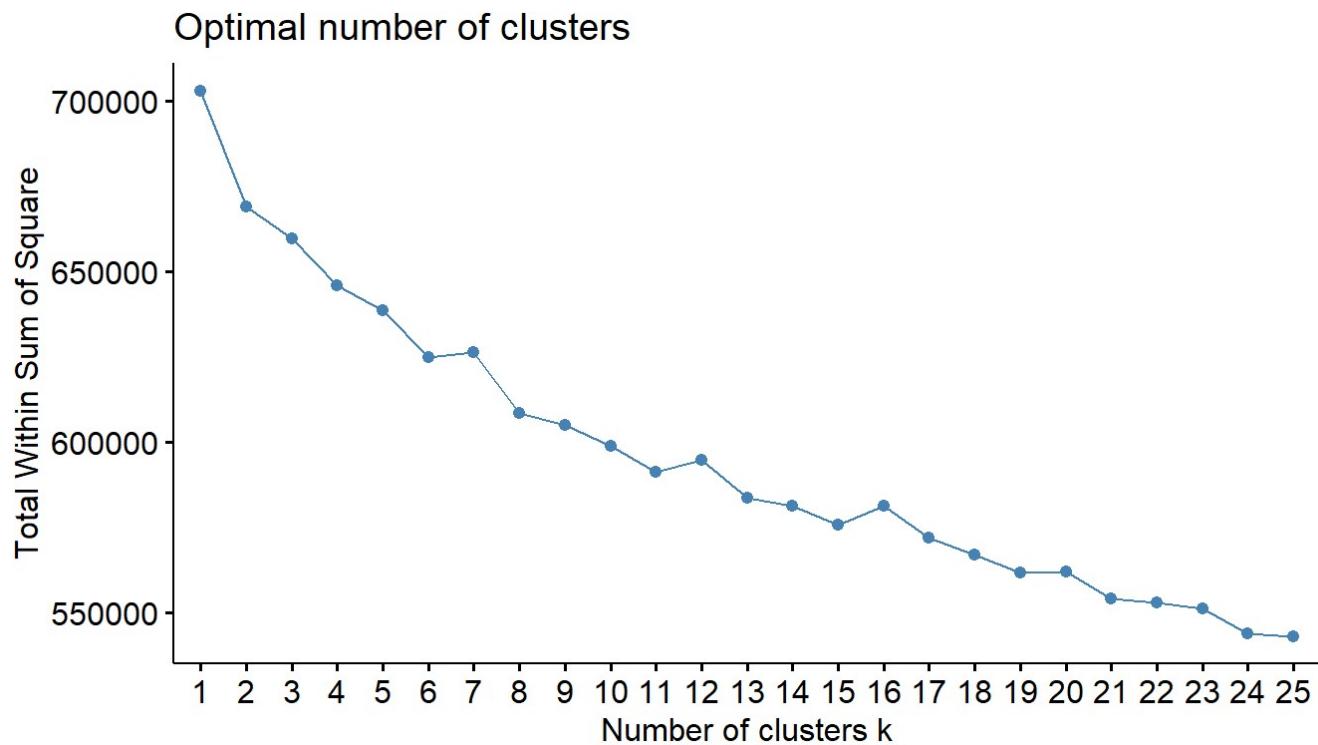
```
## [1] 2930 240
```

Ahora que todas nuestras variables están representadas numéricamente, podemos realizar la agrupación de k -medias como lo hicimos en las secciones anteriores. Usando el método del codo, no parece haber un número definitivo de grupos para usar.

Hide

```
set.seed(123)

fviz_nbclust(
  ames_1hot_scaled,
  kmeans,
  method = "wss",
  k.max = 25,
  verbose = FALSE
)
```

[Hide](#)

#Número sugerido de agrupaciones para datos de Ames codificados utilizando la agrupación de k -medias y el criterio de codo.

Desafortunadamente, este es un problema común. A medida que se expande el número de características, el rendimiento de k -means tiende a romperse y los enfoques de k -means y el agrupamiento jerárquico se vuelven lentos e ineficaces. Esto sucede, generalmente, a medida que sus datos se vuelven más escasos. Una opción adicional para datos muy mezclados es usar la medida de distancia de Gower (Gower 1971 (<http://members.cbio.mines-paristech.fr/~jvert/svn/bibli/local/Gower1971general.pdf>)), que aplica un cálculo de distancia particular que funciona bien para cada tipo de datos. Las métricas utilizadas para cada tipo de datos incluyen:

- **Cuantitativo (intervalo)**: distancia de Manhattan normalizada por rango;
- **ordinal**: la variable se clasifica primero, luego se usa la distancia de Manhattan
- **nominal**: las variables con k categorías se convierten primero en k columnas binarias (es decir, codificación one-hot) y luego se utiliza el coeficiente Dice. Para calcular la métrica de los datos para dos observaciones (X, Y), el algoritmo analiza todas las variables categóricas codificadas en un solo punto y las califica como:
 - a - número de dummies 1 para ambas observaciones
 - b - Número de dummies 1 para X y 0 para Y
 - c - Número de dummies 0 para X y 1 para Y
 - d - Número de dummies 0 para ambos

y luego usa la siguiente fórmula:

$$D = \frac{2a}{2a + b + c}$$

Podemos usar la función `cluster::daisy()` para crear una matriz de distancia de Gower a partir de nuestros

datos, esta función realiza las transformaciones de datos categóricas para que pueda proporcionar los datos en el formato original.

[Hide](#)

```
# Original data minus Sale_Price
ames_full <- AmesHousing::make_ames() %>% select(-Sale_Price)

# Compute Gower distance for original data
gower_dst <- daisy(ames_full, metric = "gower")
```

Ahora podemos alimentar los resultados en cualquier algoritmo de agrupación que acepte una matriz de distancia. Esto incluye principalmente `cluster::pam()`, `cluster::diana()` y `cluster::agnes()` (`stats::kmeans()` y `cluster::clara()` no aceptan matrices de distancia como entradas).

- `cluster::diana()` y `cluster::agnes()` son algoritmos de agrupamiento jerárquico.

3.7 Métodos populares de inicialización

En su artículo clásico MacQueen (1967), *J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297, Berkeley, CA, USA, 1967*, propuso un método de inicialización simple que elige K semillas al azar. Este es el método más simple y ha sido ampliamente utilizado en la literatura. Los otros métodos populares de inicialización de K-means que se han utilizado con éxito para mejorar el rendimiento de agrupamiento se dan a continuación.

1. Hartigan and Wong [J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979]. Usando el concepto de densidad de vecinos más cercanos, este método sugiere que los puntos que están bien separados y tienen una gran cantidad de puntos dentro de su esfera multidimensional circundante pueden ser buenos candidatos para los puntos iniciales. La distancia euclidiana promedio por pares entre puntos se calcula utilizando la ecuación (1). Los puntos subsiguientes se eligen en el orden de su densidad decreciente y simultáneamente manteniendo la separación de d_1 de todas las semillas anteriores.

$$d_1 = \frac{1}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^N \|x_i - x_j\| \quad (1)$$

2. Milligan. [G. W. Milligan. A Monte Carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199, 1981]: Utilizando los resultados del agrupamiento jerárquico aglomerativo (con la ayuda del dendrograma), este método utiliza los resultados obtenidos del método de Ward. El método de Ward elige los centroides iniciales mediante el uso de la suma de los errores al cuadrado para evaluar la distancia entre dos grupos. El método de Ward es un enfoque codicioso y mantiene el crecimiento aglomerativo lo más pequeño posible.
3. Bradley y Fayyad [P. S. Bradley and U.M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, volume 66. San Francisco, CA, USA, 1998]: Elija submuestras aleatorias de los datos y aplique el agrupamiento K-means a todas estas submuestras utilizando semillas aleatorias. Luego se recopilan los centroides de cada una de estas submuestras y se crea un nuevo conjunto de datos que consta de solo estos centroides. Este nuevo

conjunto de datos se agrupa utilizando estos centroides como semillas iniciales. El mínimo SSE obtenido garantiza el mejor conjunto de semillas elegido.

4. K-Means++ [D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.]: El algoritmo K-means++ selecciona cuidadosamente los centroides iniciales para la agrupación de K-means. El algoritmo sigue un enfoque simple basado en la probabilidad donde inicialmente se selecciona al azar el primer centroide. El siguiente centroide seleccionado es el que está más alejado del centroide seleccionado actualmente. Esta selección se decide en base a una puntuación de probabilidad ponderada. La selección continúa hasta que tengamos K centroides y luego la agrupación de K-mean se realiza utilizando estos centroides.

4 k-medoids

4.1 Idea intuitiva

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.

Una definición más exacta del término medoid es: elemento dentro de un cluster cuya distancia (diferencia) promedio entre él y todos los demás elementos del mismo cluster es lo menor posible. Se corresponde con el elemento más central del cluster y por lo tanto puede considerarse como el más representativo. El hecho de utilizar medoids en lugar de centroides hace de *k-medoids* un método más robusto que *k-means*, viéndose menos afectado por outliers. A modo de idea intuitiva puede considerarse como la analogía entre media y mediana.

1. El algoritmo más empleado para aplicar *k-medoids* se conoce como PAM (Partitioning Around Medoids) y sigue los siguientes pasos:
2. Seleccionar k observaciones aleatorias como medoids iniciales. También es posible identificarlas de forma específica.
3. Calcular la matriz de distancia entre todas las observaciones si esta no se ha calculado anteriormente.
4. Asignar cada observación a su medoid más cercano.
5. Para cada uno de los clusters creados, comprobar si seleccionando otra observación como medoid se consigue reducir la distancia promedio del cluster, si esto ocurre, seleccionar la observación que consigue una mayor reducción como nuevo medoid.

Si al menos un medoid ha cambiado en el paso 4, volver al paso 3, de lo contrario, se termina el proceso.

A diferencia del algoritmo *k-means*, en el que se minimiza la suma total de cuadrados intra-cluster (suma de las distancias al cuadrado de cada observación respecto a su centroide), el algoritmo PAM minimiza la suma de las diferencias de cada observación respecto a su medoid.

Por lo general, el método de *k-medoids* se utiliza cuando se conoce o se sospecha de la presencia de outliers. Si esto ocurre, es recomendable utilizar como medida de similitud la distancia de Manhattan, ya que es menos sensible a outliers que la euclídea.

4.2 Ventajas y desventajas

- k -medoids es un método de clustering más robusto que k -means, por lo es más adecuado cuando el set de datos contiene outliers o ruido.
- Al igual que k -means, necesita que se especifique de antemano el número de clusters que se van a crear. Esto puede ser complicado de determinar si no se dispone de información adicional sobre los datos. Muchas de las estrategias empleadas en k -means para identificar el numero óptimo, pueden aplicarse en k -medoids.
- Para sets de datos grandes necesita muchos recursos computacionales. En tal situación se recomienda aplicar el método CLARA.

4.3 Ejemplo

El set de datos `USArrests` contiene información sobre el número de delitos (asaltos, asesinatos y secuestros) junto con el porcentaje de población urbana para cada uno de los 50 estados de USA. Se pretende estudiar si existe una agrupación subyacente de los estados mediante clustering. Dado que se sospecha de la presencia de outliers se recurre a k -medoids.

El proceso a seguir en R para aplicar el método de k -medoids es igual al seguido en k -means como vimos en el ejemplo de arriba, pero en este caso empleando la función `pam()` del paquete `cluster`.

Hide

```
data("USArrests")
str(USArrests)
```

```
## 'data.frame':    50 obs. of  4 variables:
##   $ Murder   : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##   $ Assault  : int  236 263 294 190 276 204 110 238 335 211 ...
##   $ UrbanPop : int  58 48 80 50 91 78 77 72 80 60 ...
##   $ Rape     : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

Como la magnitud de los valores difiere notablemente entre variables, se procede a escalarlas antes de aplicar el clustering.

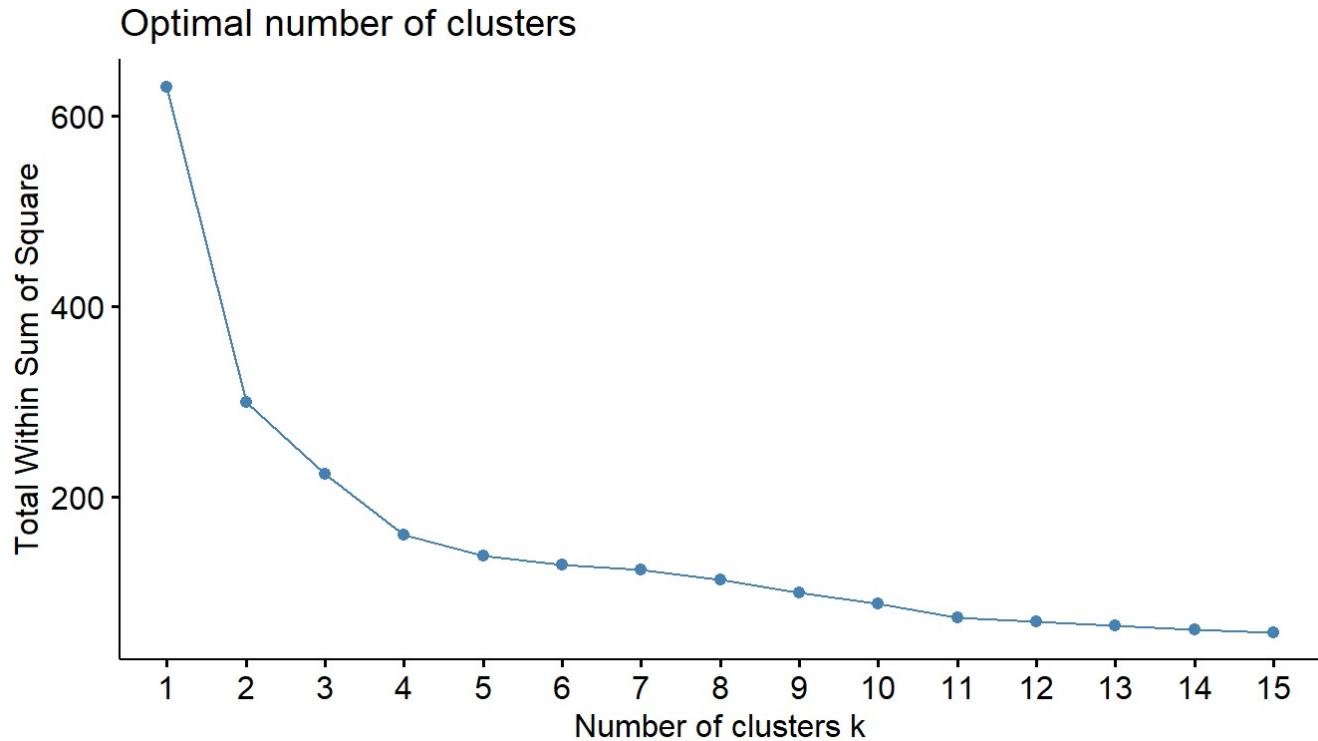
Hide

```
datos <- scale(USArrests)
```

Se evalúa la reducción de varianza total intra-cluster para un rango de valores k con el objetivo de identificar el número óptimo de clusters (elbow method, método del codo). En este caso, dado que se sospecha de la presencia de outliers, se emplea la distancia de Manhattan como medida de similitud.

Hide

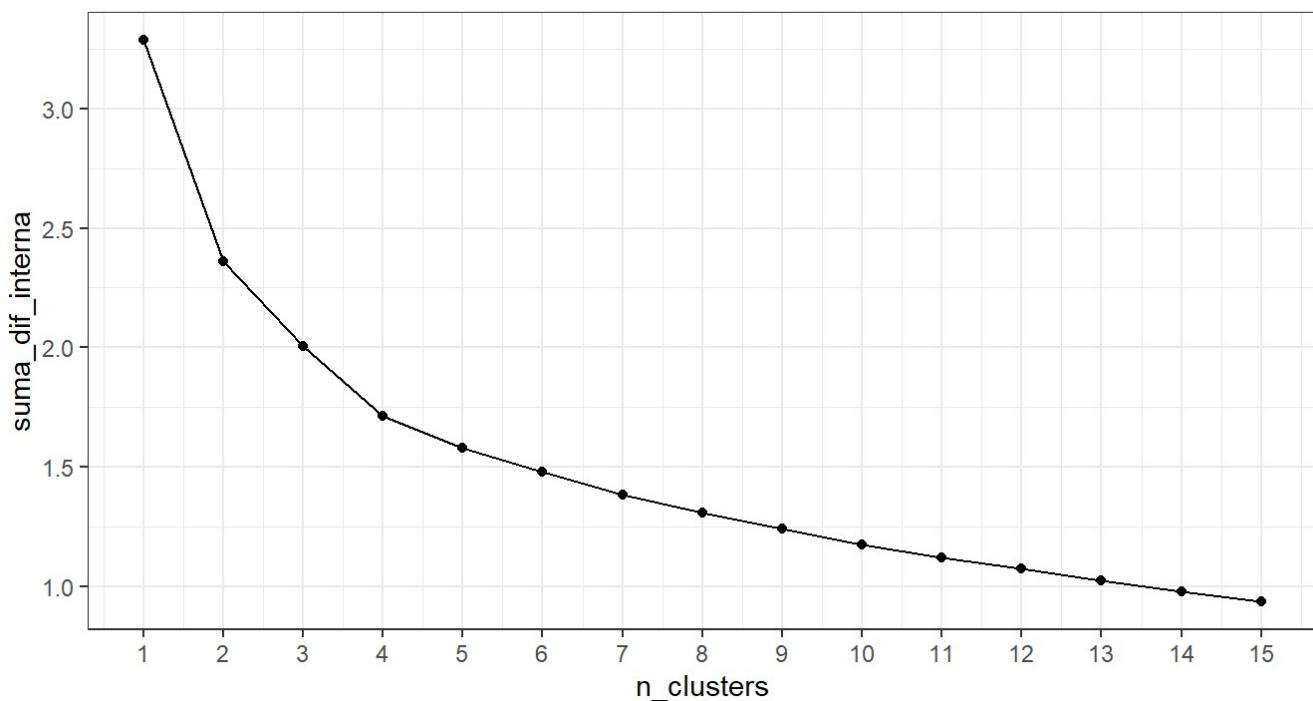
```
library(cluster)
library(factoextra)
fviz_nbclust(x = datos, FUNcluster = pam, method = "wss", k.max = 15, diss = dist(datos,
method = "manhattan"))
```

[Hide](#)

```
# Misma 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 cluster_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 = datos)
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 = "Evolucion de la suma total de diferencias intra-cluster") +
  theme_bw()
```

Evolucion de la suma total de diferencias intra-cluster



Al igual que ocurría al aplicar k -means a estos datos, a partir de 4 clusters la reducción en la suma total de diferencias internas parece estabilizarse, indicando que $k = 4$ es una buena opción.

Hide

```
set.seed(123)
pam_clusters <- pam(x = datos, k = 4, metric = "manhattan")
pam_clusters
```

```

## Medoids:
##           ID Murder Assault UrbanPop      Rape
## Alabama    1  1.2425641   0.7828393 -0.5209066 -0.003416473
## Michigan   22  0.9900104   1.0108275   0.5844655  1.480613993
## Oklahoma  36 -0.2727580  -0.2371077   0.1699510 -0.131534211
## Iowa      15 -1.2829727 -1.3770485 -0.5899924 -1.060387812
## Clustering vector:
##           Alabama      Alaska     Arizona     Arkansas California
##             1            2            2            3            2
##           Colorado Connecticut Delaware Florida Georgia
##             2            4            3            2            1
##           Hawaii   Idaho Illinois Indiana Iowa
##             3            4            2            3            4
##           Kansas Kentucky Louisiana Maine Maryland
##             3            3            1            4            2
##           Massachusetts Michigan Minnesota Mississippi Missouri
##             3            2            4            1            3
##           Montana Nebraska Nevada New Hampshire New Jersey
##             3            3            2            4            3
##           New Mexico New York North Carolina North Dakota Ohio
##             2            2            1            4            3
##           Oklahoma Oregon Pennsylvania Rhode Island South Carolina
##             3            3            3            3            1
##           South Dakota Tennessee Texas Utah Vermont
##             4            1            2            3            4
##           Virginia Washington West Virginia Wisconsin Wyoming
##             3            3            4            4            3
## Objective function:
##       build      swap
## 1.730682 1.712075
##
## Available components:
## [1] "medoids"     "id.med"      "clustering"  "objective"  "isolation"
## [6] "clusinfo"    "silinfo"     "diss"        "call"       "data"

```

El objeto devuelto por `pam()` contiene entre otra información: las observaciones que finalmente se han seleccionado como medoids y el cluster al que se ha asignado cada observación (clustering).

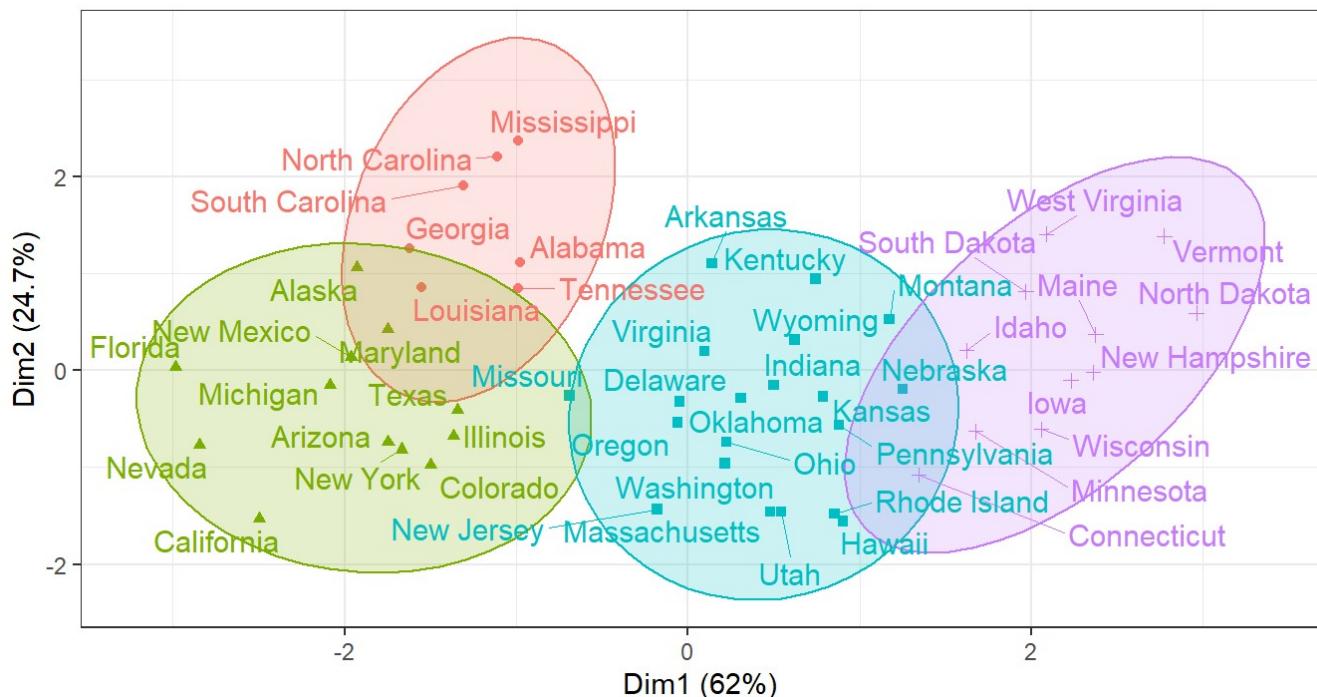
[Hide](#)

```

fviz_cluster(object = pam_clusters, data = datos, ellipse.type = "t",
             repel = TRUE) +
  theme_bw() +
  labs(title = "Resultados clustering PAM") +
  theme(legend.position = "none")

```

Resultados clustering PAM



```
# Como en k-medoids no hay centroides, no se muestran en la representación ni
# tampoco las distancias desde este al resto de observaciones
```

La función `fviz_cluster()` no permite resaltar las observaciones que actúan como medoids, sin embargo, al tratarse de un objeto `ggplot2`, es sencillo conseguirlo.

Hide

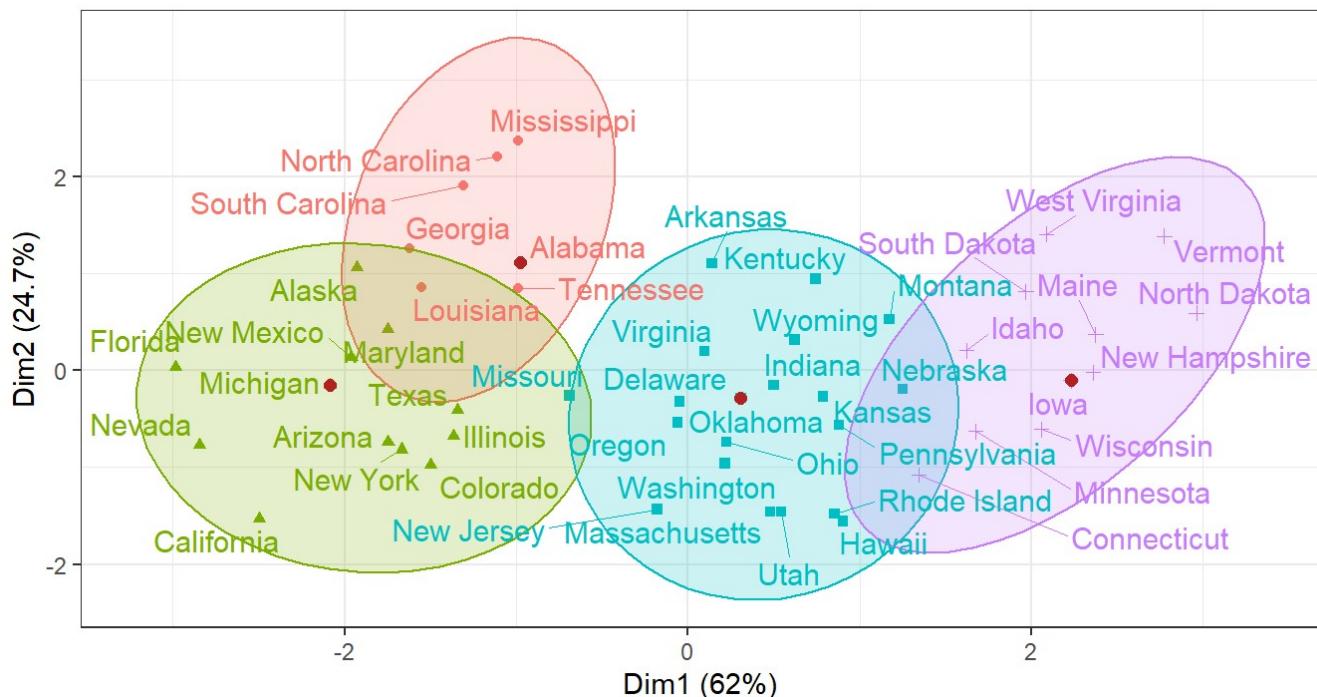
```
# Como hay más de 2 variables, se están representando las 2 primeras componentes
# de un PCA. Se tienen que calcular el PCA y extraer las proyecciones almacenadas
# en el elemento x.
medoids <- prcomp(datos)$x

# Se seleccionan únicamente las proyecciones de las observaciones que son medoids
medoids <- medoids[rownames(pam_clusters$medoids), c("PC1", "PC2")]
medoids <- as.data.frame(medoids)

# Se emplean los mismos nombres que en el objeto ggplot
colnames(medoids) <- c("x", "y")

# Creación del gráfico
fviz_cluster(object = pam_clusters, data = datos, ellipse.type = "t",
             repel = TRUE) + theme_bw() +
  # Se resaltan las observaciones que actúan como medoids
  geom_point(data = medoids, color = "firebrick", size = 2) +
  labs(title = "Resultados clustering PAM") +
  theme(legend.position = "none")
```

Resultados clustering PAM



5 CLARA

5.1 Idea intuitiva

Una de las limitaciones del método k -medoids-clustering es que su algoritmo requiere mucha memoria RAM, lo que impide que se pueda aplicar cuando el set de datos contiene varios miles de observaciones. CLARA (Clustering Large Applications, agrupación de grandes aplicaciones) es un método que combina la idea de k -medoids con el resampling (remuestreo) para que pueda aplicarse a grandes volúmenes de datos.

En lugar de intentar encontrar los medoids empleando todos los datos a la vez, CLARA selecciona una muestra aleatoria de un tamaño determinado y le aplica el algoritmo de PAM (K -medoids) para encontrar los clusters óptimos acorde a esa muestra. Utilizando esos medoids se agrupan las observaciones de todo el set de datos. La calidad de los medoids resultantes se cuantifica con la suma total de las distancias entre cada observación del set de datos y su correspondiente medoid (suma total de distancias intra-clusters). CLARA repite este proceso un número predeterminado de veces con el objetivo de reducir el bias de muestreo. Por último, se seleccionan como clusters finales los obtenidos con aquellos medoids que han conseguido menor suma total de distancias. A continuación, se describen los pasos del algoritmo CLARA.

1. Se divide aleatoriamente el set de datos en n partes de igual tamaño, donde n es un valor que determina el analista.
2. Para cada una de las n partes:
 1. Aplicar el algoritmo PAM e identificar cuáles son los k medoids.
 2. Utilizando los medoids del paso anterior, agrupa todas las observaciones del set de datos.
 3. Calcula la suma total de las distancias entre cada observación del set de datos y su correspondiente medoid (suma total de distancias intra-clusters).
3. Seleccionar como clustering final aquel que ha conseguido *menor* suma total de distancias intra-clusters en el paso sub 2.3.

5.2 Ejemplo

Para ilustrar la aplicación del método CLARA se simula un set de datos bidimensional (dos variables) con 500 observaciones, de las cuales 200 pertenecen a un grupo y 300 a otro (número de grupos reales = 2).

Hide

```
set.seed(1234)
grupo_1 <- cbind(rnorm(n = 200, mean = 0, sd = 8), rnorm(n = 200, mean = 0, sd = 8))
grupo_2 <- cbind(rnorm(n = 300, mean = 30, sd = 8), rnorm(n = 300, mean = 30, sd = 8))
datos2 <- rbind(grupo_1, grupo_2)
colnames(datos2) <- c("x", "y")
head(datos2)
```

```
##           x         y
## [1,] -9.656526 3.881815
## [2,]  2.219434 5.574150
## [3,]  8.675529 1.484111
## [4,] -18.765582 5.605868
## [5,]   3.432998 2.493448
## [6,]   4.048447 6.083699
```

La función `clara()` del paquete `cluster` permite aplicar el algoritmo CLARA. Entre sus argumentos destaca: una matriz numérica x donde cada fila es una observación, el número de clusters k , la medida de distancia empleada en el argumento `metric` (euclídea o manhattan), si los datos se tienen que estandarizar se coloca en el argumento `stand` el valor de TRUE, el número de muestras (`samples`) en las que se divide el set de datos (recomendable 50) y si se utiliza el algoritmo PAM, se coloca en el argumento `pamLike` el valor de TRUE.

Hide

```
#library(cluster)
#library(factoextra)
clara_clusters <- clara(x = datos2, k = 2, metric = "manhattan", stand = TRUE, samples = 50, pamLike = TRUE)
clara_clusters
```

```

## Call: clara(x = datos2, k = 2, metric = "manhattan", stand = TRUE, samples = 50, pamLike = TRUE)
## Medoids:
##          x           y
## [1,] -0.2780831 1.269004
## [2,] 30.3298450 29.233511
## Objective function: 0.8629488
## Clustering vector: int [1:500] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
## Cluster sizes:      202 298
## Best sample:
## [1] 17 30 33 34 63 71 80 84 85 100 115 141 149 162 165 167 168
## [18] 171 188 214 223 249 259 261 291 302 318 320 333 386 391 402 412 417
## [35] 419 422 437 439 440 450 469 472 498 499
##
## Available components:
## [1] "sample"      "medoids"     "i.med"       "clustering"   "objective"
## [6] "clusinfo"    "diss"        "call"        "silinfo"     "data"

```

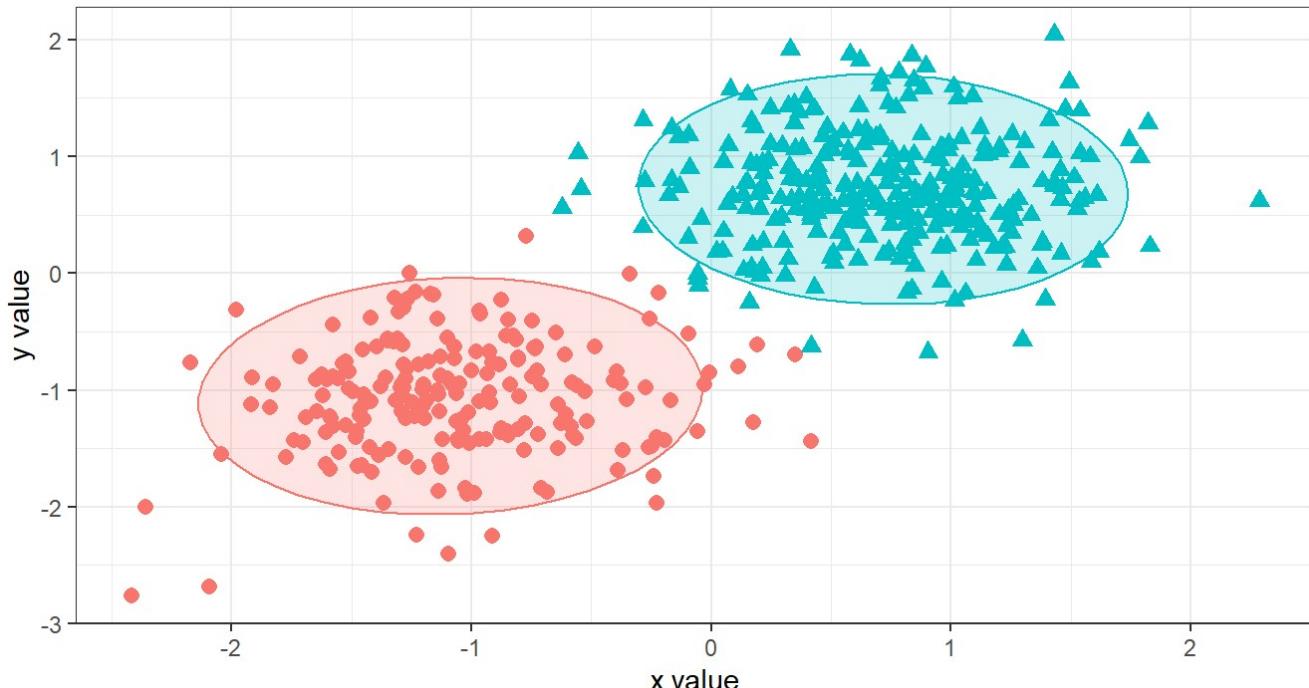
[Hide](#)

```

fviz_cluster(object = clara_clusters, ellipse.type = "t", geom = "point",
              pointsize = 2.5) + theme_bw() +
  labs(title = "Resultados del clustering usando CLARA") +
  theme(legend.position = "none")

```

Resultados del clustering usando CLARA

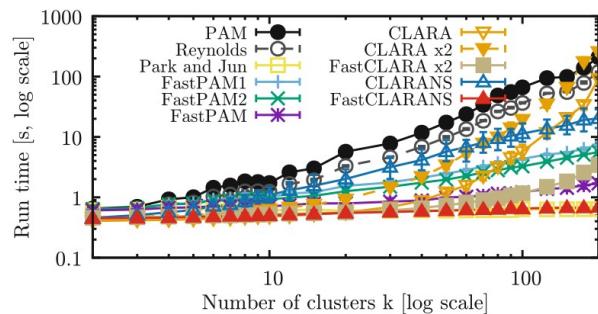


6 Agrupamiento de k-medoides más rápido

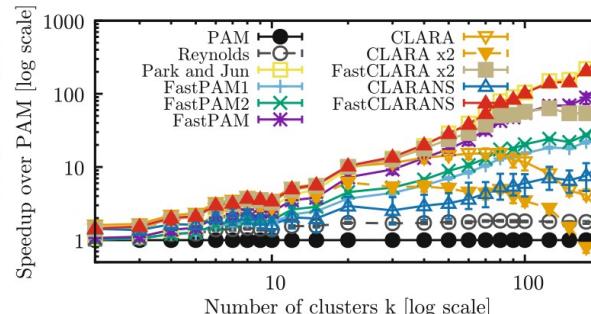
La agrupación de datos no euclidianos es difícil, y uno de los algoritmos más utilizados además de la

agrupación jerárquica es el algoritmo popular llamado Particionamiento alrededor de los medoides (PAM), también conocido simplemente como k-medoides. En la geometría euclídea, la media, como se usa en k-means, es un buen estimador para el centro del grupo, pero esto no existe para las diferencias arbitrarias. PAM utiliza el medoide en su lugar, el objeto con la menor diferencia con todos los demás en el clúster. Esta noción de centralidad se puede utilizar con cualquier (des)-similitud y, por lo tanto, es de gran relevancia para diferentes aplicaciones.

Esta propuesta fue publicada el 23 de setiembre de este año, y titula “Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms” de los autores Erich SchubertEmail y Peter J. Rousseeuw, tenemos una versión publicada y cuya descarga es gratuita desde arxiv (<https://arxiv.org/abs/1810.05691>) y cuya fue publicación se encuentra publicado como Proceedings (https://link.springer.com/chapter/10.1007/978-3-030-32047-8_16) en la conferencia internacional número, SISAP (Similarity Search and Applications) 2019.



(a) Run time in log-log space

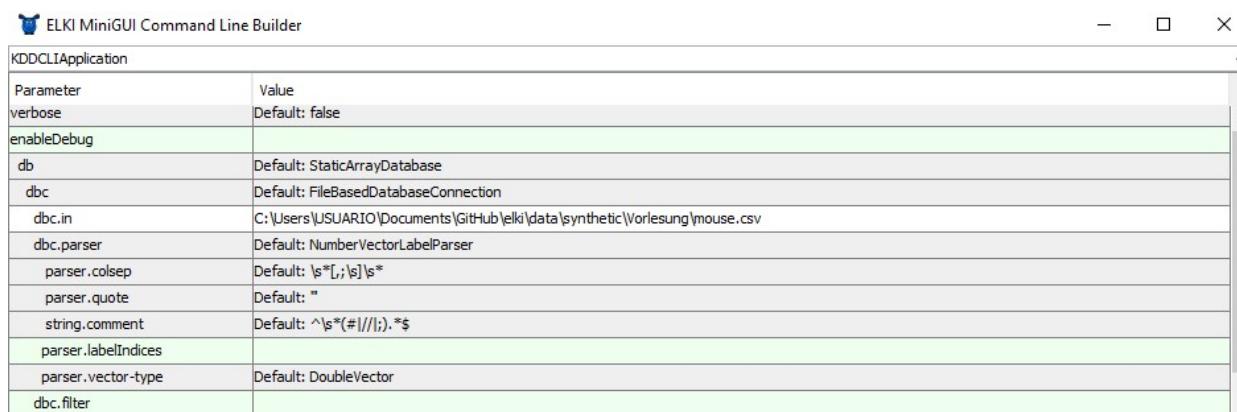


(b) Speedup in log-log space

Una escala logarítmica es una escala de medida que utiliza el logaritmo de una cantidad física en lugar de la propia cantidad. Un ejemplo sencillo de escala logarítmica muestra divisiones igualmente espaciadas en el eje vertical de un gráfico marcadas con 1, 10, 100, 1000, ..., en vez de 0, 1, 2, 3, ...

El algoritmo FastPAM2 se encuentra implementado en el paquete `cluster` (<https://cran.r-project.org/web/packages/cluster/index.html>) del R, vea la página 46 del manual de ayuda en PDF. Sin embargo los algoritmos propuestos en el artículo están implementados en el ELKI (a large open-source library for data analysis (<https://arxiv.org/abs/1902.03616>)), veamos los algoritmo que se encuentran implementados (<https://elki-project.github.io/algorithms/>) en dicho aplicativo y notaremos que ahí si esta implementado el FastCLARA por ejemplo y no lo está en el paquete en R.

Veamos a continuación algunas capturas de los resultados gráficos que muestra el ELKI, se puede descargar en formato java ejecutable, seguimos el ejemplo de la misma página (<https://elki-project.github.io/tutorial/>).

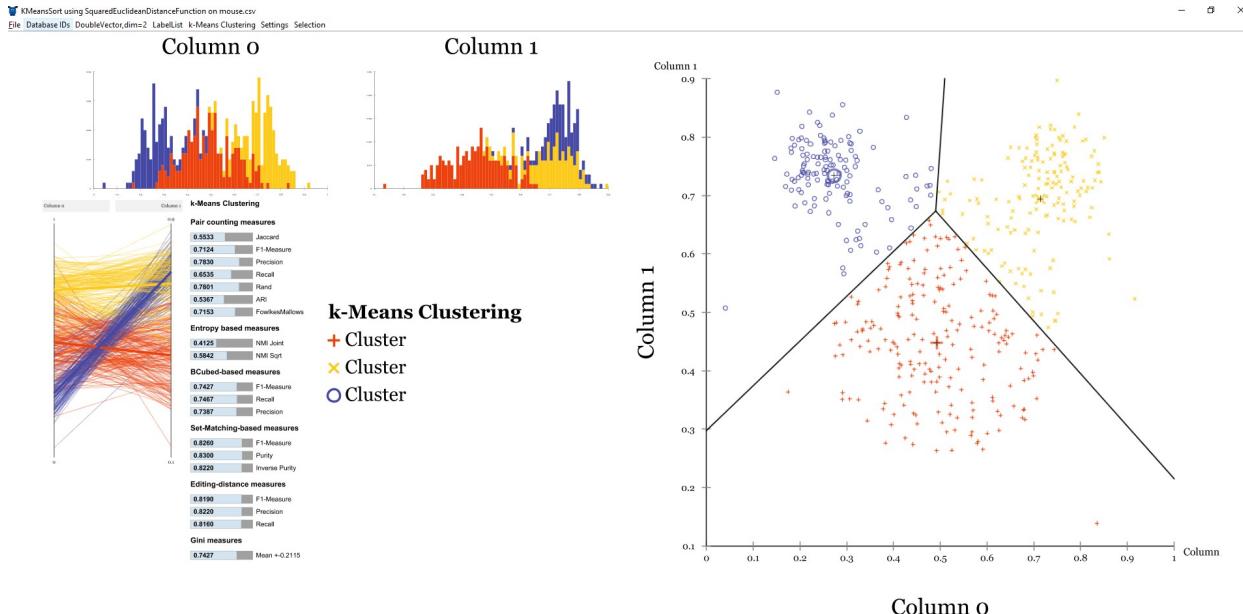


db.index	
time	true
algorithm	clustering.kmeans.KMeansSort
kmeans.k	3
kmeans.initialization	FirstKInitialMeans
algorithm.distancefunction	Default: minkowski.SquaredEuclideanDistanceFunction
kmeans.maxiter	Default: 0
evaluator	Default: AutomaticEvaluation
resulthandler	Default: AutomaticVisualization

Load Save Remove Run Task

```
dbc.in "C:\Users\USUARIO\Documents\GitHub\elki\data\synthetic\Vorlesung\mouse.csv" -time -algorithm clustering.kmeans.KMeansSort -kmeans.k 3 -kmeans.initialization FirstKInitialMeans
```

```
de.lmu.ifis.dbs.elki.datasource.FileBasedDatabaseConnection.load: 16 ms
class de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.initialization.FirstKInitialMeans.time: 0 ms
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.1.reassignments: 500
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.1.variance-sum: 16.28386667935692
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.2.reassignments: 51
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.2.variance-sum: 9.711397483997999
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.3.reassignments: 36
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.3.variance-sum: 8.550499038341506
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.4.reassignments: 12
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.4.variance-sum: 8.191069775404758
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.5.reassignments: 8
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.5.variance-sum: 8.13730217690176
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.6.reassignments: 2
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.6.variance-sum: 8.120727835968975
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.7.reassignments: 2
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.7.variance-sum: 8.118013966486162
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.8.reassignments: 2
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.8.variance-sum: 8.115208106060335
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.iterations: 9
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.distance-computations: 8740
de.lmu.ifis.dbs.elki.algorithm.clustering.kmeans.KMeansSort.runtime: 16 ms
```



7 Fuzzy clustering

7.1 Idea intuitiva

Los métodos de clustering descritos hasta ahora (K-means, hierarchical, K-medoids, CLARA...) asignan cada observación únicamente a un cluster, de ahí que también se conozcan como hard clustering. Los métodos de fuzzy clustering o soft clustering se caracterizan porque, cada observación, puede pertenecer potencialmente a

varios clusters, en concreto, cada observación tiene asignado un grado de pertenencia a cada uno de los cluster.

Fuzzy c-means (FCM) es uno de los algoritmos más empleado para generar fuzzy clustering. Se asemeja en gran medida al algoritmo de k-means pero con dos diferencias:

- El cálculo de los centroides de los clusters. La definición de centroide empleada por c-means es: la media de todas las observaciones del set de datos ponderada por la probabilidad de pertenecer al cluster.
- Devuelve para cada observación la probabilidad de pertenecer a cada cluster.

7.2 Ejemplo

El set de datos USArests contiene información sobre el número de delitos (asaltos, asesinatos y secuestros) junto con el porcentaje de población urbana para cada uno de los 50 estados de USA. Se pretende estudiar si existe una agrupación subyacente de los estados empleando fuzzy clustering.

Hide

```
data("USArrests")
```

Como la magnitud de los valores difiere notablemente entre variables, se procede a escalarlas antes de aplicar el clustering.

Hide

```
datos <- scale(USArrests)
```

La función `fanny()` (fuzzy analysis) del paquete `cluster` permite aplicar el algoritmo c-means clustering.

Hide

```
#library(cluster)
fuzzy_cluster <- fanny(x = datos, diss = FALSE, k = 3, metric = "euclidean", stand = FALSE)
```

El objeto devuelto `fanny()` incluye entre sus elementos: una matriz con el grado de pertenencia de cada observación a cada cluster (las columnas son los clusters y las filas las observaciones).

Hide

```
head(fuzzy_cluster$membership)
```

```
##          [,1]      [,2]      [,3]
## Alabama  0.4676004 0.3144516 0.2179480
## Alaska   0.4278809 0.3178707 0.2542484
## Arizona  0.5092197 0.2945668 0.1962135
## Arkansas 0.2934077 0.3787718 0.3278205
## California 0.4668527 0.3084149 0.2247324
## Colorado  0.4542018 0.3236683 0.2221299
```

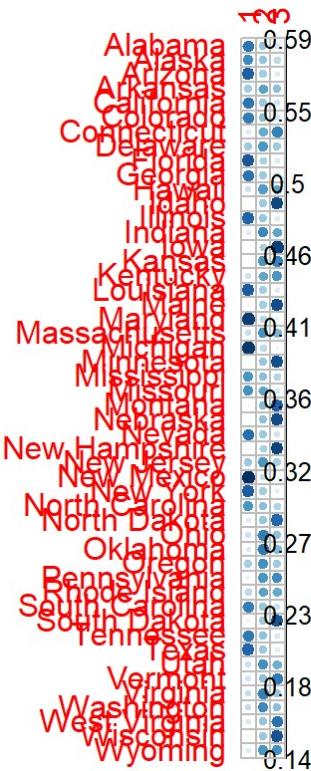
Hide

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

[Hide](#)

```
corrplot(fuzzy_cluster$membership, is.corr = FALSE)
```



El coeficiente de partición Dunn normalizado y sin normalizar. Valores normalizados próximos a 0 indican que la estructura tiene un alto nivel fuzzy y valores próximos a 1 lo contrario.

[Hide](#)

```
fuzzy_cluster$coeff
```

```
## dunn_coeff normalized
## 0.37371071 0.06056606
```

El cluster al que se ha asignado mayoritariamente cada observación.

[Hide](#)

```
head(fuzzy_cluster$clustering)
```

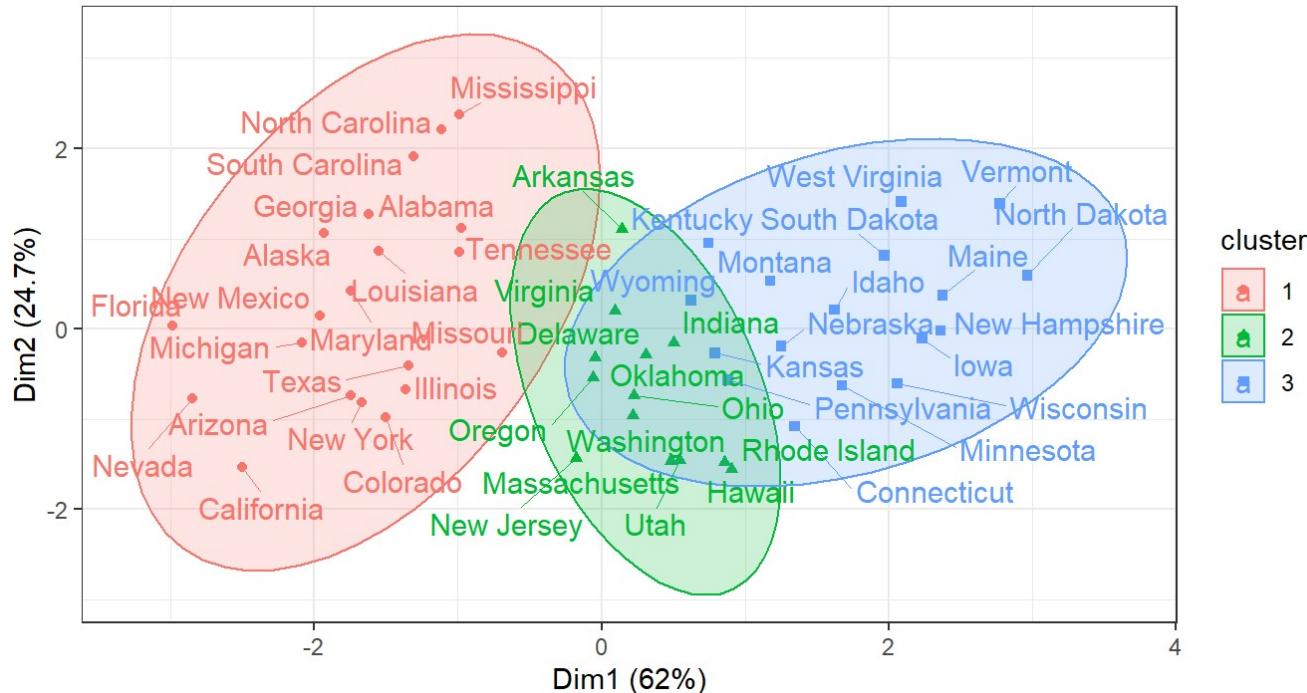
```
##      Alabama      Alaska      Arizona      Arkansas      California      Colorado
##          1          1          1          2          1          1
```

Para obtener una representación gráfica del clustering se puede emplear la función `fviz_cluster()`.

Hide

```
#library(factoextra)
fviz_cluster(object = fuzzy_cluster, repel = TRUE, ellipse.type = "norm", pallete = "jco") + theme_bw() + labs(title = "Fuzzy Cluster plot")
```

Fuzzy Cluster plot



8 Model based clustering

8.1 Idea intuitiva

El clustering basado en modelos considera que las observaciones proceden de una distribución que es a su vez una combinación de dos o más componentes (clusters), cada uno con una distribución propia. En principio, cada cluster puede estar descrito por cualquier función de densidad, pero normalmente se asume que siguen una distribución multivariante normal. Para estimar los parámetros que definen la función de distribución de cada cluster (media y matriz de covarianza si se asume que son de tipo normal) se recurre al algoritmo de Expectation-Maximization (EM). Este resuelve distintos modelos en los que el volumen, forma y orientación de las distribuciones pueden considerarse iguales para todos los clusters o distintas para cada uno. Por ejemplo, un posible modelo es: volumen constante, forma variable, orientación variable.

El paquete `mclust` emplea maximum likelihood para ajustar todos estos modelos con distinto número k de clusters y selecciona el mejor en base al Bayesian Information Criterion (BIC).

8.2 Ejemplo

El set de datos diabetes del paquete `mclust` contiene 3 parámetros sanguíneos medidos en 145 pacientes con 3 tipos distintos de diabetes. Se pretende emplear model-based-clustering para encontrar las agrupaciones.

[Hide](#)

```
library(mclust)
data("diabetes")
head(diabetes)
```

```
##      class glucose insulin sspg
## 1 Normal     80     356   124
## 2 Normal     97     289   117
## 3 Normal    105     319   143
## 4 Normal     90     356   199
## 5 Normal     90     323   240
## 6 Normal     86     381   157
```

[Hide](#)

```
# Estandarización de variables
datos4 <- scale(diabetes[, -1])

# Model-based-clustering
model_clustering2 <- Mclust(data = datos4, G = 1:10)
summary(model_clustering2)
```

```
## -----
## Gaussian finite mixture model fitted by EM algorithm
## -----
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model
## with 3 components:
##
## log-likelihood  n df      BIC      ICL
##           -169.0908 145 29 -482.5069 -501.4662
##
## Clustering table:
##   1 2 3
## 81 36 28
```

El algoritmo de ajuste selecciona como mejor modelo el formado por 3 clusters, cada uno con forma elipsoidal y con volume, shape y orientation propias.

El clustering basado en modelos es de tipo fuzzy, es decir, para cada observación se calcula un grado de pertenencia a cada cluster y se asigna finalmente al que mayor valor tiene.

[Hide](#)

```
# Grado de asignación a cada cluster
head(model_clustering2$z)
```

```
##          [,1]      [,2]      [,3]
## 1 0.9906745 0.008991332 3.341728e-04
## 2 0.9822128 0.017783229 3.974744e-06
## 3 0.9777871 0.022157665 5.527579e-05
## 4 0.9774763 0.022312280 2.113743e-04
## 5 0.9208978 0.079034264 6.789759e-05
## 6 0.9863472 0.012977950 6.748263e-04
```

```
# Clasificación final
head(model_clustering2$classification)
```

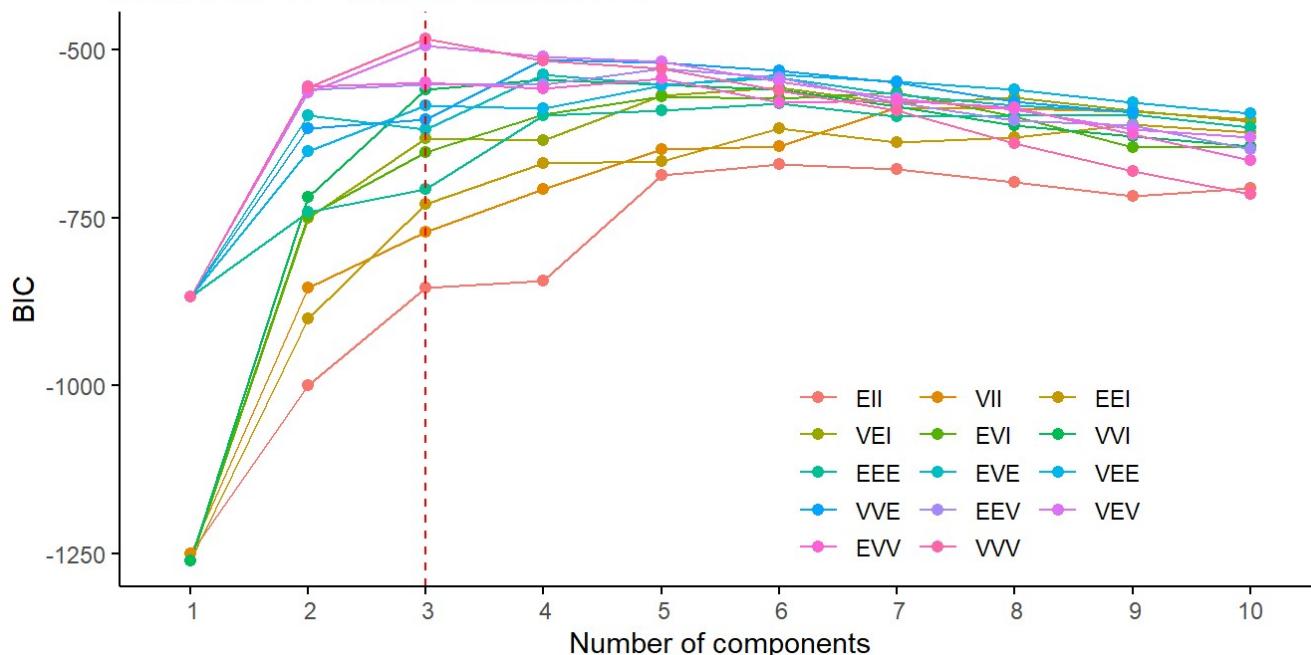
```
## 1 2 3 4 5 6
## 1 1 1 1 1 1
```

La visualización del clustering puede hacerse mediante la función `plot.Mclust()` o mediante `fviz_mclust()`.

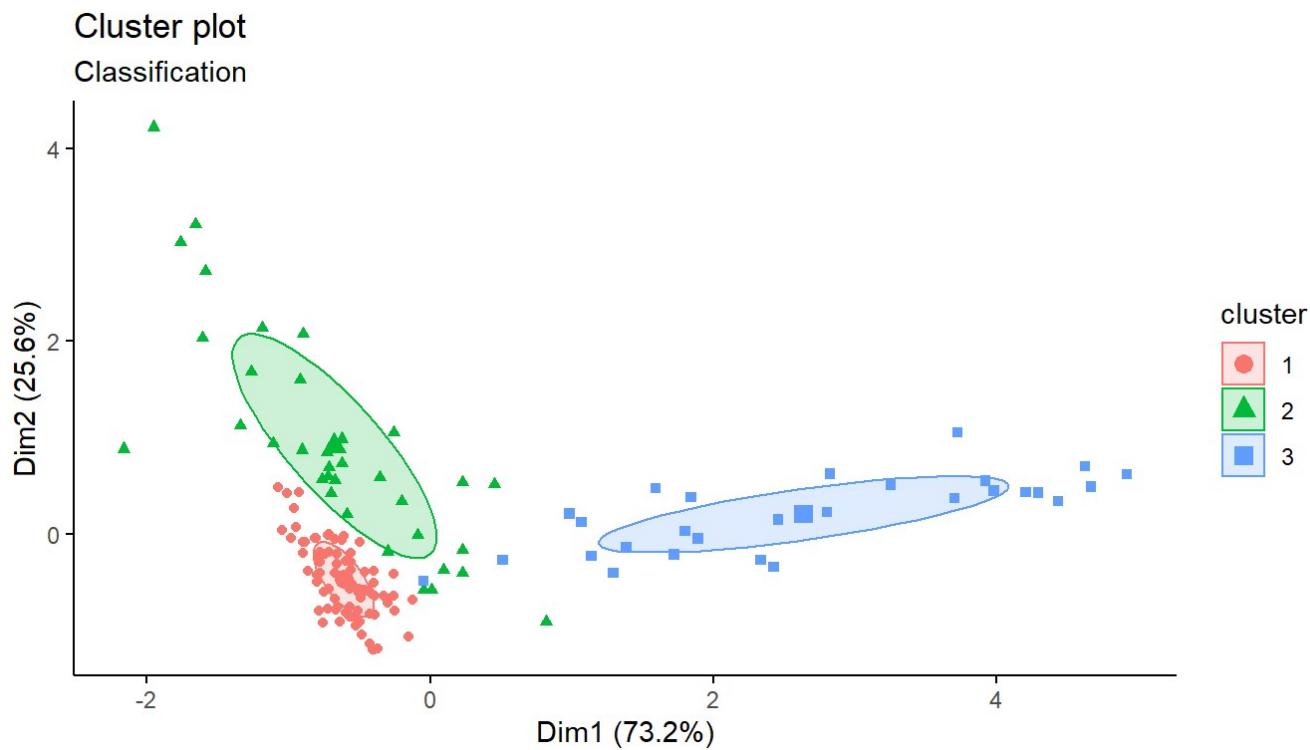
```
#library(factoextra)
# Curvas del valor BIC en función del número de clusters para cada modelo.
# Atención al orden en el que se muestra la variable horizontal, por defecto es
# alfabético.
fviz_mclust(object = model_clustering2, what = "BIC", pallete = "jco") +
  scale_x_discrete(limits = c(1:10))
```

Model selection

Best model: VVV | Optimal clusters: n = 3

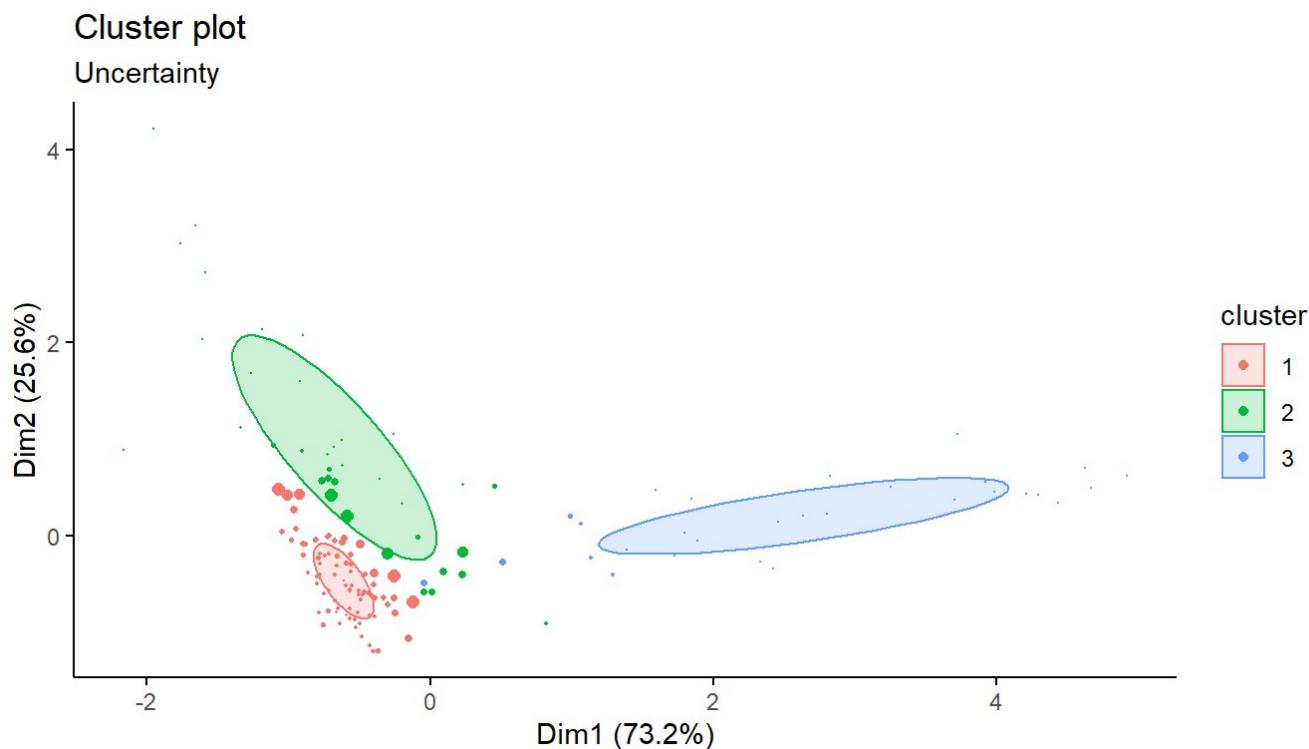


```
# Clusters  
fviz_mclust(model_clustering2, what = "classification", geom = "point", pallete = "jco")
```



Hide

```
# Certeza de las clasificaciones. Cuanto mayor el tamaño del punto menor la  
# seguridad de la asignación  
fviz_mclust(model_clustering2, what = "uncertainty", pallete = "jco")
```



8.3 K-bumps

El algoritmo EM es sensible a la elección de los valores iniciales de los parámetros a estimar. La selección inapropiada de estos valores iniciales puede conducir a una estimación insatisfactoria de la mezcla. Varios métodos que intentan resolver este problema se informan en la literatura (ver Biernacki et al., 2003 [Biernacki C, Celeux G, Govaert G (2003) Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. Comput Stat Data Anal 41(3):561–575], para una descripción general de las estrategias de inicialización simples). La inicialización aleatoria es probablemente la más utilizada, aunque las estimaciones finales podrían ser diferentes cada vez que se ejecuta el algoritmo. Debido a que estamos fijando k por adelantado, una estrategia alternativa y bastante común consiste en usar un método de particionamiento más sofisticado que intente particionar directamente los datos observados x en un conjunto de k clústeres, satisfaciendo así el principio de datos completos subyacente al algoritmo EM (ver Maulik et al. 2011, Sección 1.2.3, para una encuesta reciente sobre estos métodos). Al usar algún método de particionamiento, los pesos iniciales y los parámetros de los componentes para el algoritmo EM se pueden estimar por separado en cada grupo.

La mayoría de los método de particionamiento existentes, de los cuales el k -means es el más utilizado, asociar aproximadamente un punto representativo a cada grupo (por ejemplo, el centroide para k -means, el medoide para los k -medoides de Kaufman y Rousseeuw 1990, y la modoa para k -modes de Huang 1998); a continuación, se realiza un algoritmo iterativo, comenzando desde una inicialización aleatoria de estos puntos representativos, para asignar cada punto de datos al clúster cuyo punto representativo es el más cercano. Desafortunadamente, cuando la partición proporcionada por un PCM se utiliza para inicializar el algoritmo EM, las estimaciones finales pueden cambiar en este caso cada vez que se ejecuta el algoritmo, porque el algoritmo EM hereda la inicialización aleatoria del algoritmo de agrupación de particiones.

9 Agrupamiento espacial basado en densidad

(DBSCAN)

9.1 Idea intuitiva

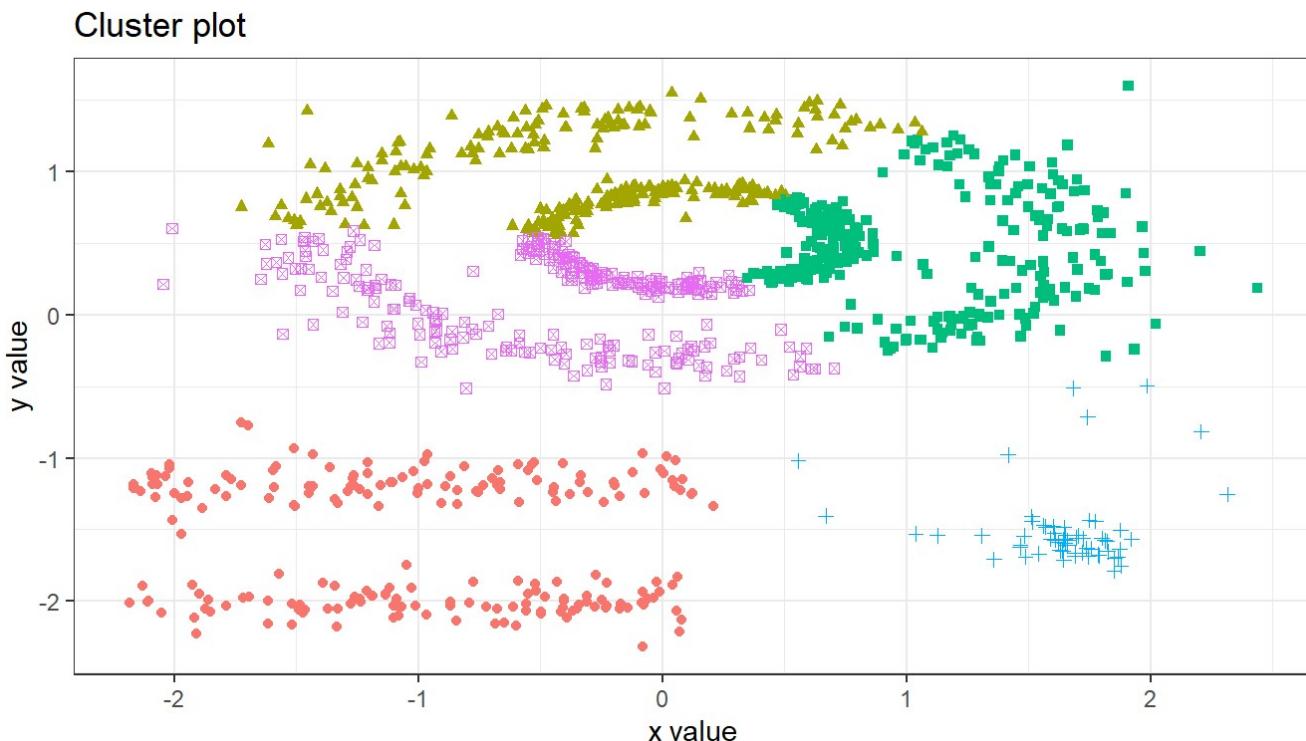
Density-based spatial clustering of applications with noise (DBSCAN) fue presentado en 1996 por Ester et al. como una forma de identificar clusters siguiendo el modo intuitivo en el que lo hace el cerebro humano, identificando regiones con alta densidad de observaciones separadas por regiones de baja densidad.

Véase la siguiente representación bidimensional de los datos multishape del paquete factoextra.

El cerebro humano identifica fácilmente 5 agrupaciones y algunas observaciones aisladas (ruido). Véanse ahora los clusters que se obtienen si se aplica, por ejemplo, K-means clustering.

Hide

```
library(factoextra)
data("multishapes")
datos <- multishapes[, 1:2]
set.seed(321)
km_clusters <- kmeans(x = datos, centers = 5, nstart = 50)
fviz_cluster(object = km_clusters, data = datos, geom = "point", ellipse = FALSE,
             show.clust.cent = FALSE, pallete = "jco") +
  theme_bw() +
  theme(legend.position = "none")
```



Los clusters generados distan mucho de representar las verdaderas agrupaciones. Esto es así porque los métodos de partitioining clustering como k-means, hierarchical, k-medoids, c-means... son buenos encontrando agrupaciones con forma esférica o convexa que no contengan un exceso de outliers o ruido, pero fallan al tratar de identificar formas arbitrarias. De ahí que el único cluster que se corresponde con un grupo real sea el

amarillo.

DBSCAN evita este problema siguiendo la idea de que, para que una observación forme parte de un cluster, tiene que haber un mínimo de observaciones vecinas dentro de un radio de proximidad y de que los clusters están separados por regiones vacías o con pocas observaciones.

El algoritmo DBSCAN necesita dos parámetros:

- Epsilon (ϵ): radio que define la región vecina a una observación, también llamada ϵ -neighborhood.
- Minimum points (minPts): número mínimo de observaciones dentro de la región epsilon.

Empleando estos dos parámetros, cada observación del set de datos se puede clasificar en una de las siguientes tres categorías:

- Core point: observación que tiene en su ϵ -neighborhood un número de observaciones vecinas igual o mayor a minPts.
- Border point: observación no satisface el mínimo de observaciones vecinas para ser core point pero que pertenece al ϵ -neighborhood de otra observación que sí es core point.
- Noise u outlier: observación que no es core point ni border point.

Por último, empleando las tres categorías anteriores se pueden definir tres niveles de conectividad entre observaciones:

- Directamente alcanzable (direct density reachable): una observación A es directamente alcanzable desde otra observación B si A forma parte del ϵ -neighborhood de B y B es un core point. Por definición, las observaciones solo pueden ser directamente alcanzables desde un core point.
- Alcanzable (density reachable): una observación A es alcanzable desde otra observación B si existe una secuencia de core points que van desde B a A .
- Densamente conectadas (density connected): dos observaciones A y B están densamente conectadas si existe una observación core point C tal que A y B son alcanzables desde C .

La siguiente imagen muestra las conexiones existentes entre un conjunto de observaciones si se emplea $min.Pts = 4$. La observación A y el resto de observaciones marcadas en rojo son core points, ya que todas ellas contienen al menos 4 observaciones vecinas (incluyéndose a ellas mismas) en su ϵ -neighborhood. Como todas son alcanzables entre ellas, forman un cluster. Las observaciones B y C no son core points pero son alcanzables desde A a través de otros core points, por lo tanto, pertenecen al mismo cluster que A . La observación N no es ni un core point ni es directamente alcanzable, por lo que se considera como ruido.

Algoritmo

1. Para cada observación x_i calcular la distancia entre ella y el resto de observaciones. Si en su ϵ -neighborhood hay un número de observaciones $\geq min.Pts$ marcar la observación como core point, de lo contrario marcarla como visitada.
2. Para cada observación x_i marcada como core point, si todavía no ha sido asignada a ningún cluster, crear uno nuevo y asignarla a él. Encontrar recursivamente todas las observaciones densamente conectadas a ella y asignarlas al mismo cluster.
3. Iterar el mismo proceso para todas las observaciones que no hayan sido visitadas.
4. Aquellas observaciones que tras haber sido visitadas no pertenecen a ningún cluster se marcan como outliers.

Como resultado, todo cluster cumple dos propiedades: todos los puntos que forman parte de un mismo cluster están densamente conectados entre ellos y, si una observación A es densamente alcanzable desde cualquier otra observación de un cluster, entonces A también pertenece al cluster.

Selección de parámetros

Como ocurre en muchas otras técnicas estadísticas, en DBSCAN no existe una forma única y exacta de encontrar el valor adecuado de epsilon (ϵ) y $minPts$. A modo orientativo se pueden seguir las siguientes premisas:

- $minPts$: cuanto mayor sea el tamaño del set de datos, mayor debe ser el valor mínimo de observaciones vecinas. En el libro Practical Guide to Cluster Analysis in R recomiendan no bajar nunca de 3. Si los datos contienen niveles altos de ruido, aumentar $minPts$ favorecerá la creación de clusters significativos menos influenciados por outliers.
- epsilon: una buena forma de escoger el valor de ϵ es estudiar las distancias promedio entre las $k = minPts$ observaciones más próximas. Al representar estas distancias en función de ϵ , el punto de inflexión de la curva suele ser un valor óptimo. Si el valor de ϵ escogido es muy pequeño, una proporción alta de las observaciones no se asignarán a ningún cluster, por el contrario, si el valor es demasiado grande, la mayoría de observaciones se agruparán en un único cluster.

Ventajas de DBSCAN

- No requiere que el usuario especifique el número de clusters.
- Es independiente de la forma que tengan los clusters, no tienen por qué ser circulares.
- Puede identificar outliers, por lo que los clusters generados no se influenciados por ellos.

Desventajas de DBSCAN

- No es un método totalmente determinístico: los border points que son alcanzables desde más de un cluster pueden asignarse a uno u otro dependiendo del orden en el que se procesen los datos.
- No genera buenos resultados cuando la densidad de los grupos es muy distinta, ya que no es posible encontrar los parámetros ϵ y $minPts$ que sirvan para todos a la vez.

9.2 Ejemplo

El set de datos multishape del paquete factoextra contiene observaciones que pertenecen a 5 grupos distintos junto con cierto ruido (outliers). Como se espera que la distribución espacial de los grupos no sea esférica, se aplica el método de clustering DBSCAN.

En R existen dos paquetes con funciones que permiten aplicar el algoritmo DBSCAN: fpc y dbscan. El segundo contiene una modificación del algoritmo original que lo hace más rápido. La función kNNdistplot del paquete dbscan calcula y representa las k-distancias para ayudar a identificar el valor óptimo de epsilon.

Hide

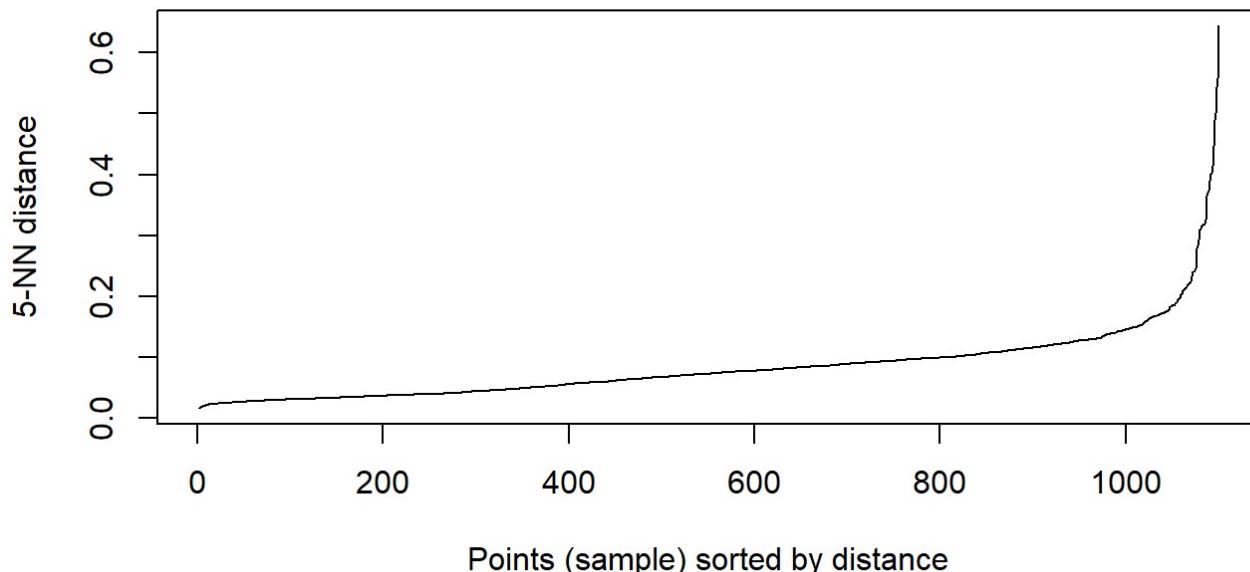
```
library(fpc)
library(dbscan)
```

```
## 
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:fpc':  
##  
##     dbscan
```

Hide

```
library(factoextra)  
data("multishapes")  
datos <- multishapes[, 1:2]  
  
# Selección del valor óptimo de epsilon. Como valor de minPts se emplea 5.  
dbscan::kNNdistplot(datos, k = 5)
```



La curva tiene el punto de inflexión en torno a 0.15, por lo que se escoge este valor como epsilon para DBSCAN.

Hide

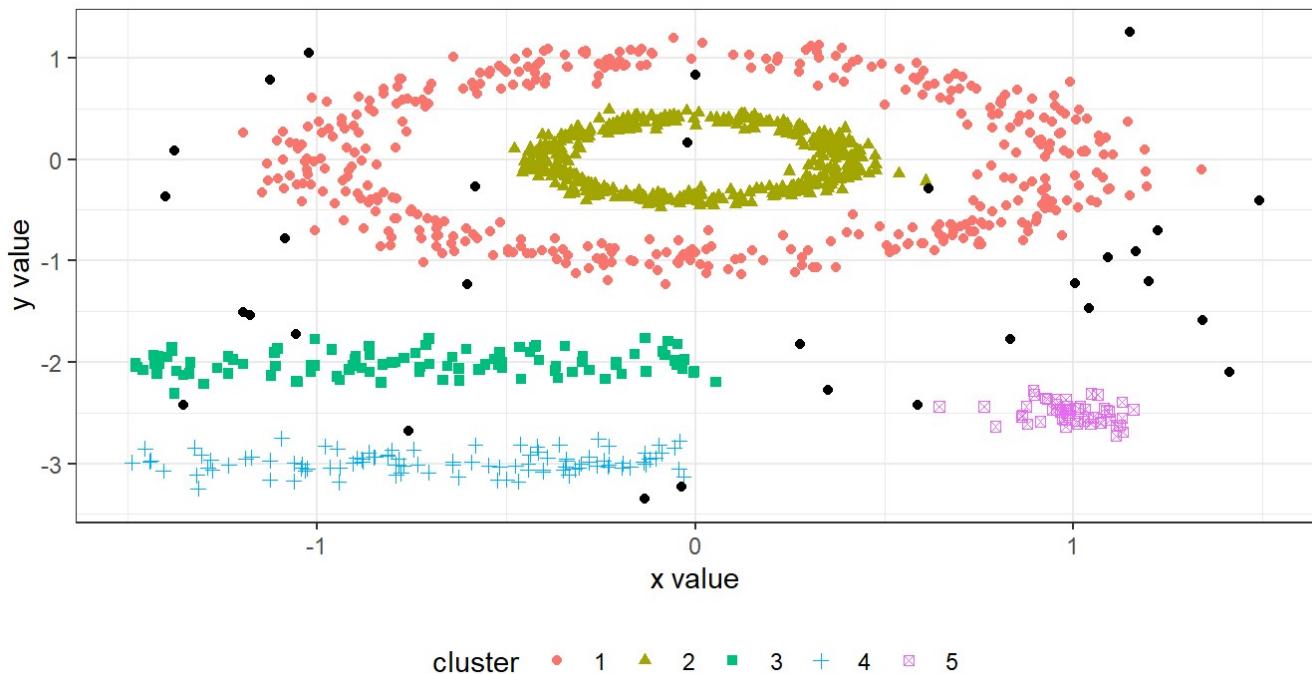
```
set.seed(321)  
  
# DBSCAN con epsilon = 0.15 y minPts = 5  
dbscan_cluster <- fpc::dbscan(data = datos, eps = 0.15, MinPts = 5)  
  
# Resultados de la asignación  
head(dbscan_cluster$cluster)
```

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

Hide

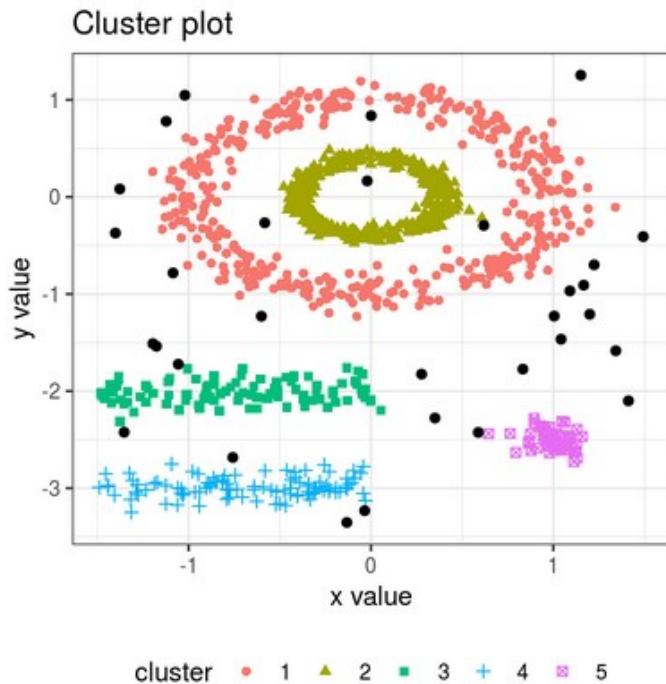
```
# Visualización de los clusters
fviz_cluster(object = dbscan_cluster, data = datos, stand = FALSE,
              geom = "point", ellipse = FALSE, show.clust.cent = FALSE,
              pallete = "jco") +
theme_bw() +
theme(legend.position = "bottom")
```

Cluster plot



Hide

```
knitr::include_graphics("images/cluster.png")
```



10 Número óptimo de clusters

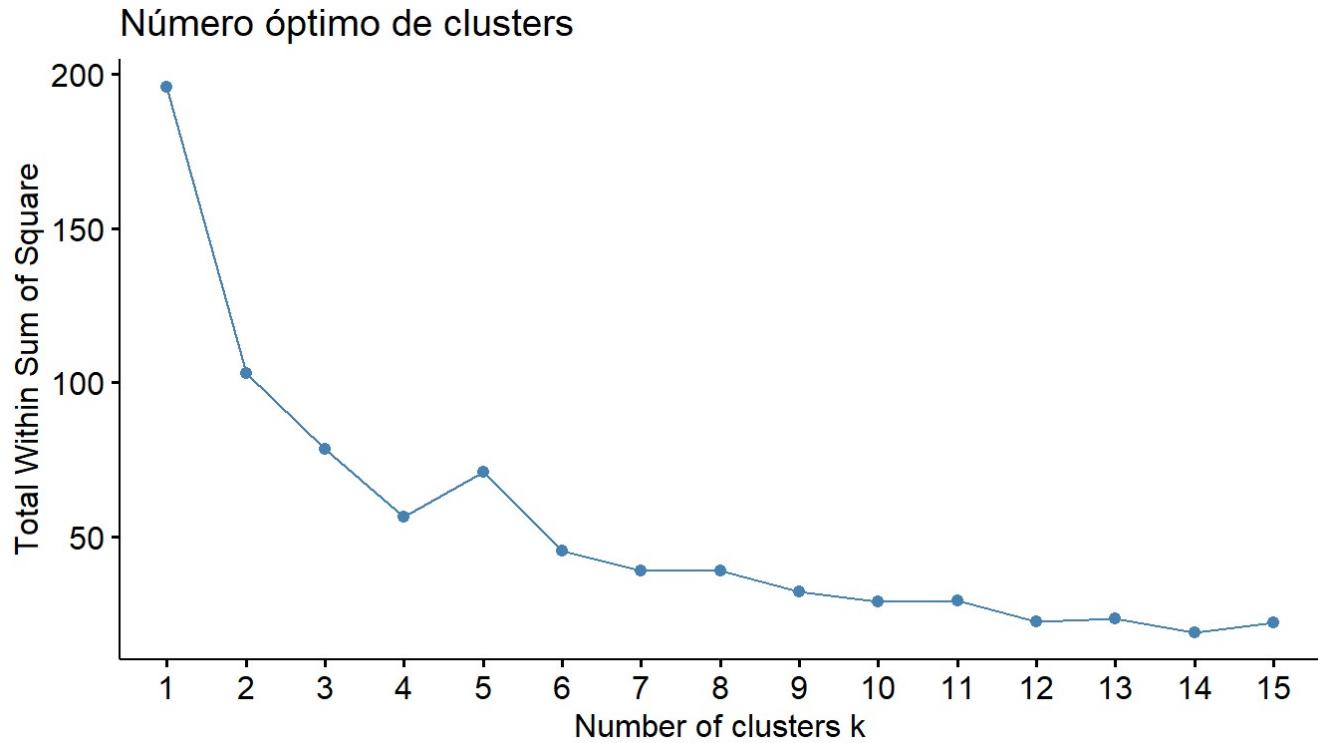
Determinar el número óptimo de clusters es uno de los pasos más complicados a la hora de aplicar métodos de clustering, sobre todo cuando se trata de partitioning clustering, donde el número se tiene que especificar antes de poder ver los resultados. No existe una forma única de averiguar el número adecuado de clusters. Es un proceso bastante subjetivo que depende en gran medida del tipo de clustering empleado y de si se dispone de información previa sobre los datos con los que se está trabajando, por ejemplo, estudios anteriores pueden sugerir o acotar las posibilidades. A pesar de ello, se han desarrollado varias estrategias que ayudan en el proceso.

10.1 Elbow method

El método Elbow sigue una estrategia comúnmente empleada para encontrar el valor óptimo de un hiperparámetro. La idea general es probar un rango de valores del hiperparámetro en cuestión, representar gráficamente los resultados obtenidos con cada uno e identificar aquel punto de la curva a partir del cual la mejora deja de ser sustancial (principio de verosimilitud). En los casos de partitioning clustering, como por ejemplo K-means, las observaciones se agrupan de una forma tal que se minimiza la varianza total intra-cluster. El método Elbow calcula la varianza total intra-cluster en función del número de clusters y escoge como óptimo aquel valor a partir del cual añadir más clusters apenas consigue mejoría. La función fviz_nbclust() del paquete factoextra automatiza todo el proceso, empleando como medida de varianza intra-cluster la suma de residuos cuadrados internos (wss).

Hide

```
library(factoextra)
datos <- scale(USArrests)
fviz_nbclust(x = datos, FUNcluster = kmeans, method = "wss", k.max = 15) +
  labs(title = "Número óptimo de clusters")
```



La curva indica que a partir de 4 clusters la mejora es mínima. Este mismo análisis también puede realizarse sin recurrir a la función `fviz_nbclust()`.

Hide

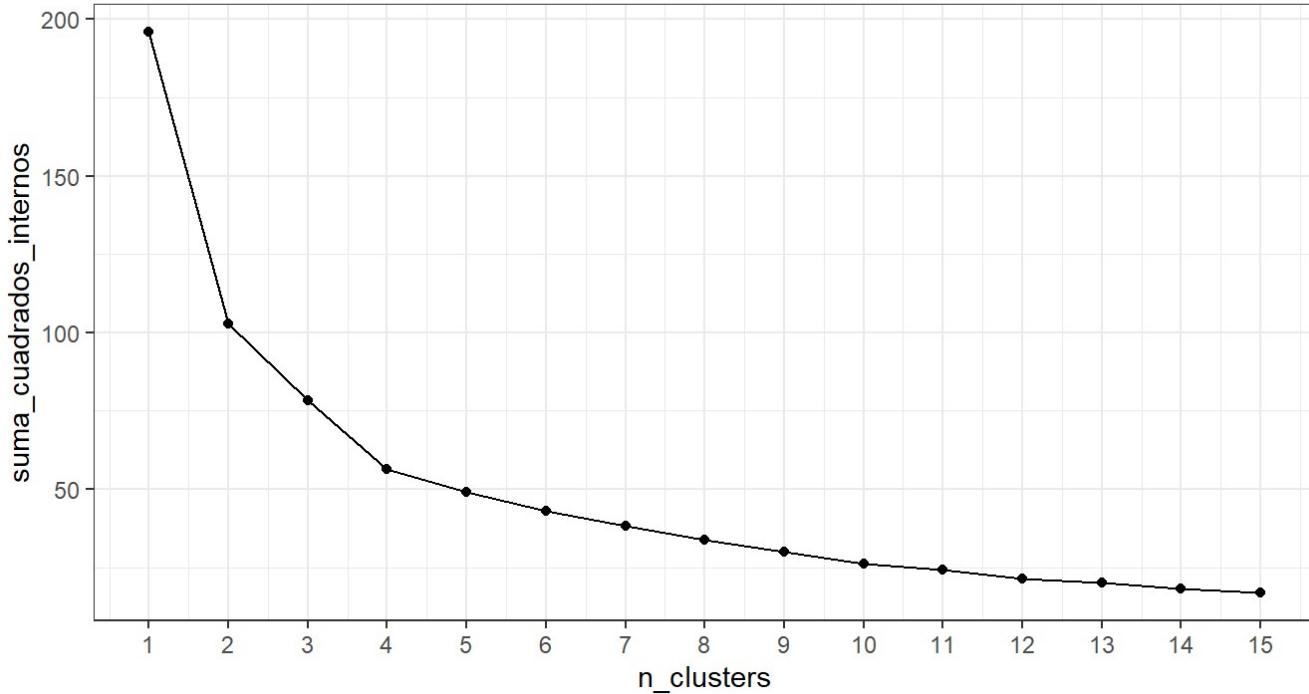
```
calcular_totwithinss <- function(n_clusters, datos, iter.max=1000, nstart=50) {  
  # Esta función aplica el algoritmo kmeans y devuelve la suma total de  
  # cuadrados internos.  
  cluster_kmeans <- kmeans(centers = n_clusters, x = datos, iter.max = iter.max,  
                           nstart = nstart)  
  return(cluster_kmeans$tot.withinss)  
}  
  
# Se aplica esta función con para diferentes valores de k  
total_withinss <- map_dbl(.x = 1:15,  
                          .f = calcular_totwithinss,  
                          datos = datos)  
total_withinss
```

```
## [1] 196.00000 102.86240  78.32327  56.40317  48.94420  42.83303  38.25764  
## [8] 33.85843  29.86789  26.18348  24.05222  21.47090  20.15762  18.04643  
## [15] 16.81152
```

Hide

```
data.frame(n_clusters = 1:15, suma_cuadrados_internos = total_withinss) %>%
  ggplot(aes(x = n_clusters, y = suma_cuadrados_internos)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:15) +
  labs(title = "Evolución de la suma total de cuadrados intra-cluster") +
  theme_bw()
```

Evolución de la suma total de cuadrados intra-cluster



10.2 Average silhouette method

El método de average silhouette es muy similar al de Elbow, con la diferencia de que, en lugar minimizar el total inter-cluster sum of squares (wss), se maximiza la media de los silhouette coefficient o índices silueta (s_i). Este coeficiente cuantifica cómo de buena es la asignación que se ha hecho de una observación comparando su similitud con el resto de observaciones de su cluster frente a las de los otros clusters. Su valor puede estar entre -1 y 1, siendo valores altos un indicativo de que la observación se ha asignado al cluster correcto.

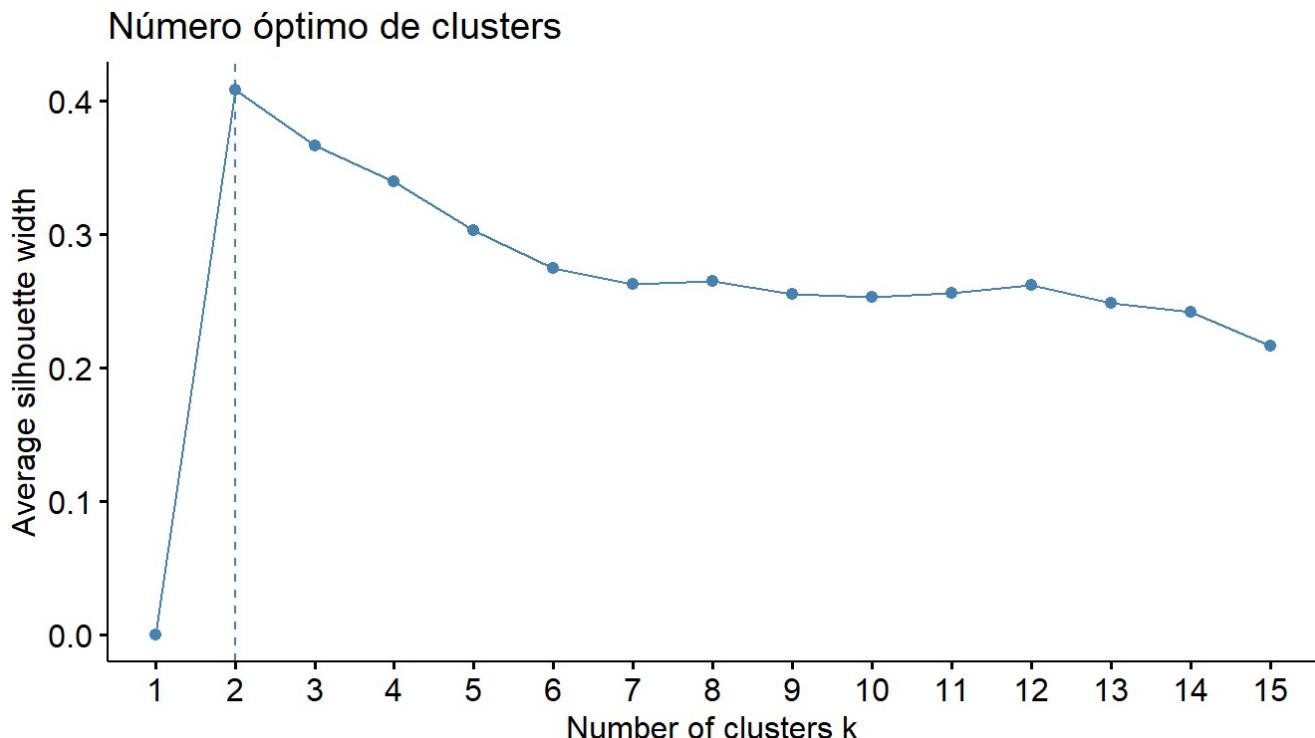
Para cada observación i , el silhouette coefficient (s_i) se obtiene del siguiente modo:

- Calcular el promedio de las distancias (llámese a_i) entre la observación i y el resto de observaciones que pertenecen al mismo cluster. Cuanto menor sea a_i , mejor ha sido la asignación de i a su cluster.
- Calcular la distancia promedio entre la observación i y el resto de clusters. Entendiendo por distancia promedio entre i y un determinado cluster C como la media de las distancias entre i y las observaciones del cluster C .
- Identificar como b_i a la menor de las distancias promedio entre i y el resto de clusters, es decir, la distancia al cluster más próximo (neighbouring cluster).
- Calcular el valor de silhouette como:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

[Hide](#)

```
library(factoextra)
datos <- scale(USArrests)
fviz_nbclust(x = datos, FUNcluster = kmeans, method = "silhouette", k.max = 15) +
  labs(title = "Número óptimo de clusters")
```



El método de average silhouette considera como número óptimo de clusters aquel que maximiza la media del silhouette coeficient de todas las observaciones, en este caso 2.

Este mismo análisis también puede realizarse sin recurrir a la función fviz_nbclust().

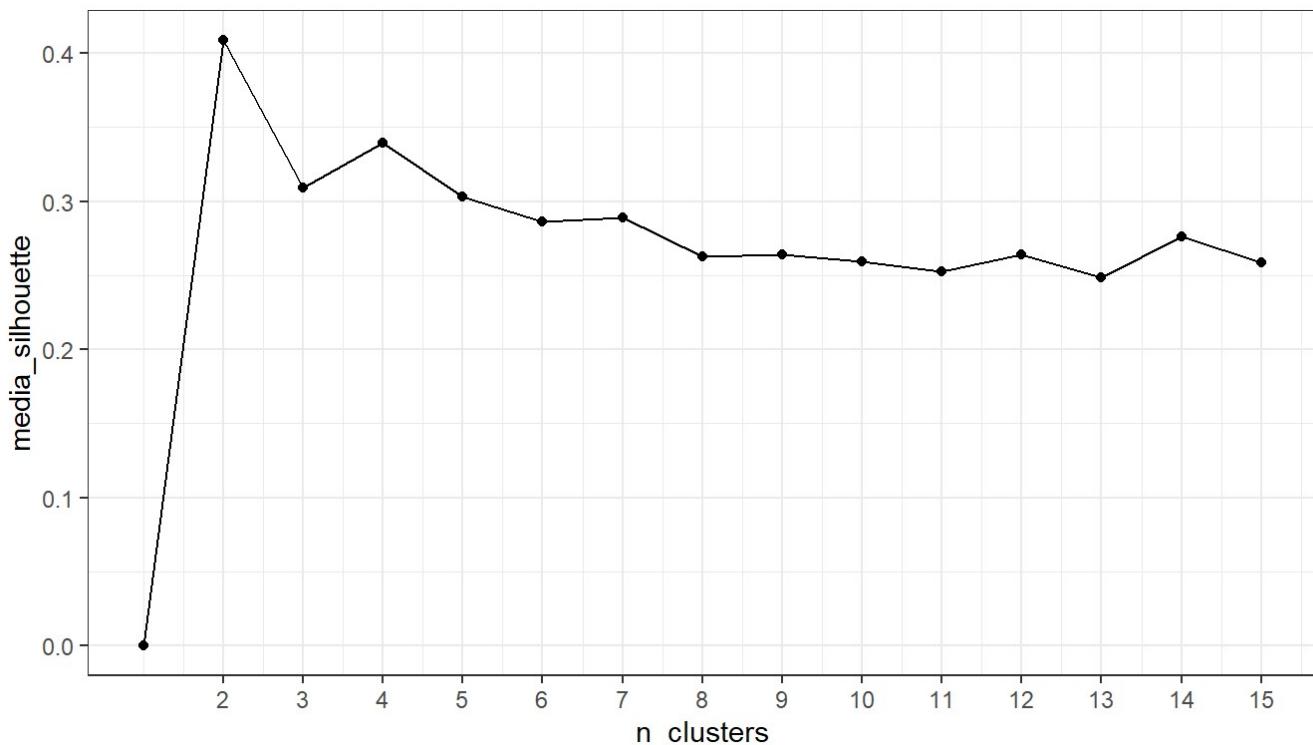
[Hide](#)

```
# ÍNDICES SILUETA KMEANS
# =====

silhouette_kmeans <- function(n_clusters, datos, iter.max=1000, nstart=50){
  # Esta función aplica el algoritmo kmeans y devuelve la media del índice silueta
  if (n_clusters == 1) {
    # Para n_clusters = 1, el indice silueta es 0
    media_silhouette <- 0
  }else {
    cluster_kmeans <- kmeans(centers = n_clusters, x = datos, iter.max = iter.max,
                             nstart = nstart)
    valores_silhouette <- cluster::silhouette(cluster_kmeans$cluster,
                                                get_dist(x = datos, method = "euclidean"))
    media_silhouette <- summary(valores_silhouette)[[4]]
  }
  return(media_silhouette)
}

datos <- scale(USArrests)
valores_medios_silhouette <- map_dbl(.x = 1:15,
                                         .f = silhouette_kmeans,
                                         datos = datos)

data.frame(n_clusters = 1:15, media_silhouette = valores_medios_silhouette) %>%
  ggplot(aes(x = n_clusters, y = media_silhouette)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 2:15) +
  theme_bw()
```



El el caso de hierarchical clustering (`hclust()` , `diana()` ...) es necesario cortar el árbol para cada uno de los valores k antes de calcular los índices silueta.

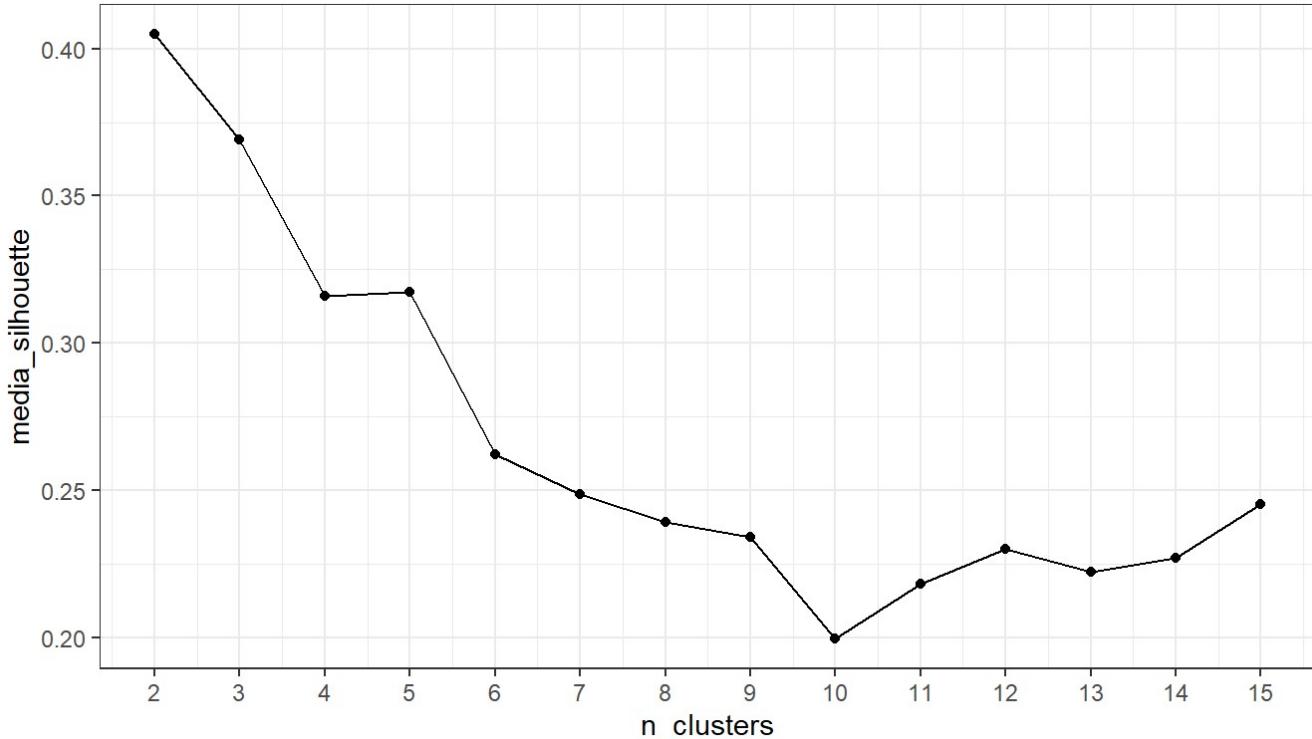
Hide

```
# ÍNDICES SILUETA HIERARCHICAL CLUSTERING
# =====

custom_silhouette <- function(n_clusters, dendograma, distancia, datos) {
  # Esta función calcula el indice silueta medio de un dendograma
  # para un determinado número de clusters.
  set.seed(123)
  valores_silhouette <- cluster::silhouette(stats::cutree(dendograma,
    k = n_clusters),
    get_dist(x = datos, method = distancia))
  media_silhouette <- summary(valores_silhouette)[[4]]
  return(media_silhouette)
}

datos <- scale(USArrests)
hc_euclidea_completo <- hclust(d = dist(x = datos, method = "euclidean"),
  method = "complete")
valores_medios_silhouette <- map_dbl(.x = 2:15,
  .f = custom_silhouette,
  dendograma = hc_euclidea_completo,
  distancia = "euclidean",
  datos = datos)

data.frame(n_clusters = 2:15, media_silhouette = valores_medios_silhouette) %>%
  ggplot(aes(x = n_clusters, y = media_silhouette)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 2:15) +
  theme_bw()
```



10.3 Gap statistic method

El estadístico gap fue publicado por R.Tibshirani, G.Walther y T. Hastie, autores también del magnífico libro *Introduction to Statistical Learning*. Este estadístico compara, para diferentes valores de k , la varianza total intra-cluster observada frente al valor esperado acorde a una distribución uniforme de referencia. La estimación del número óptimo de clusters es el valor k con el que se consigue maximizar el estadístico gap, es decir, encuentra el valor de k con el que se consigue una estructura de clusters lo más alejada posible de una distribución uniforme aleatoria. Este método puede aplicarse a cualquier tipo de clustering.

El algoritmo del gap statistic method es el siguiente:

- Hacer clustering de los datos para un rango de valores de k ($k = 1, \dots, K = n$) y calcular para cada uno el valor de la varianza total intra-cluster (W_k).
- Simular B sets de datos de referencia, todos ellos con una distribución aleatoria uniforme. Aplicar clustering a cada uno de los sets con el mismo rango de valores k empleado en los datos originales, calculando en cada caso la varianza total intra-cluster (W_{kb}). Se recomienda emplear valores de $B = 500$.
- Calcular el estadístico gap para cada valor de k como la desviación de la varianza observada W_k respecto del valor esperado acorde a la distribución de referencia (W_{kb}). Calcular también su desviación estándar.

$$\text{gap}(k) = \frac{1}{B} \sum_{b=1}^B \log(W_{kb}) - \log(W_k)$$

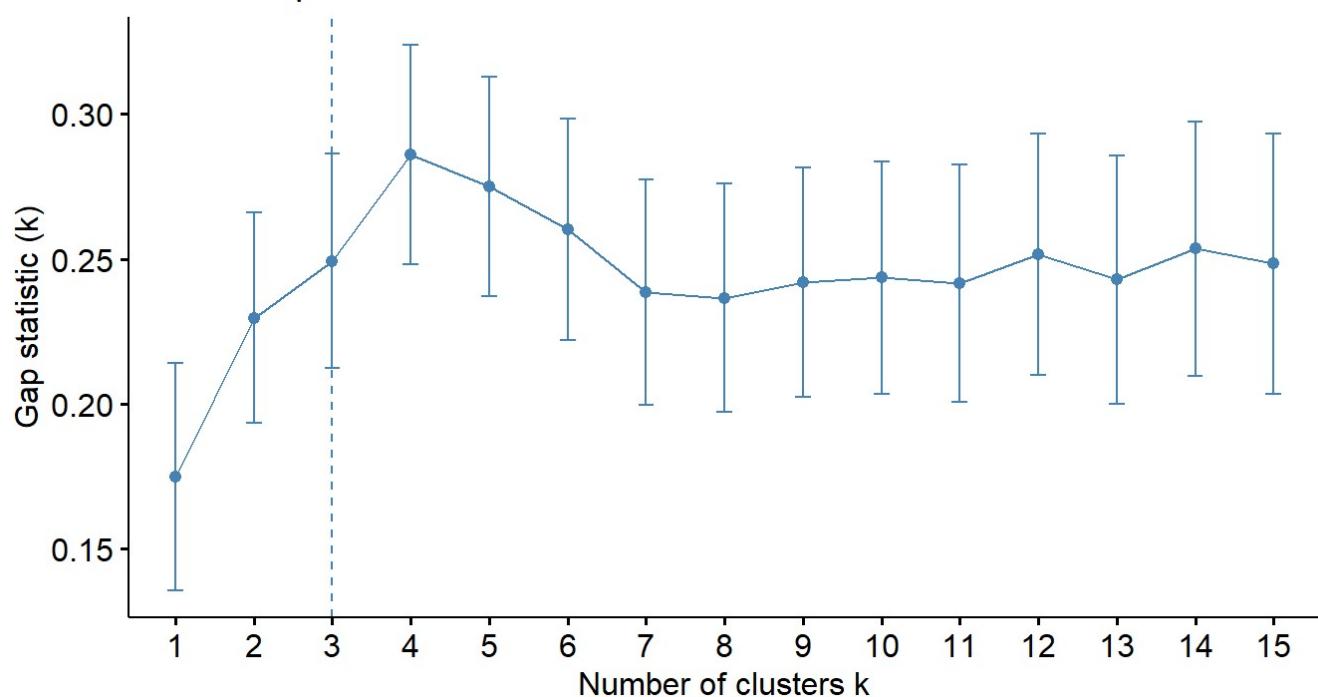
- Identificar el número de clusters óptimo como el menor de los valores k para el que el estadístico gap se aleja menos de una desviación estándar del valor gap del siguiente k : $\text{gap}(k) \geq \text{gap}(k + 1) - s_{k+1}$.

Se puede obtener el el estadístico gap con la función `fviz_nbclust()` o con la función `clusGap()` del paquete cluster.

Hide

```
library(factoextra)
datos <- scale(USArrests)
set.seed(896)
fviz_nbclust(x = datos, FUNcluster = kmeans, method = "gap_stat", nboot = 500,
             k.max = 15, verbose = FALSE, nstart = 50) +
  labs(title = "Número óptimo de clusters")
```

Número óptimo de clusters

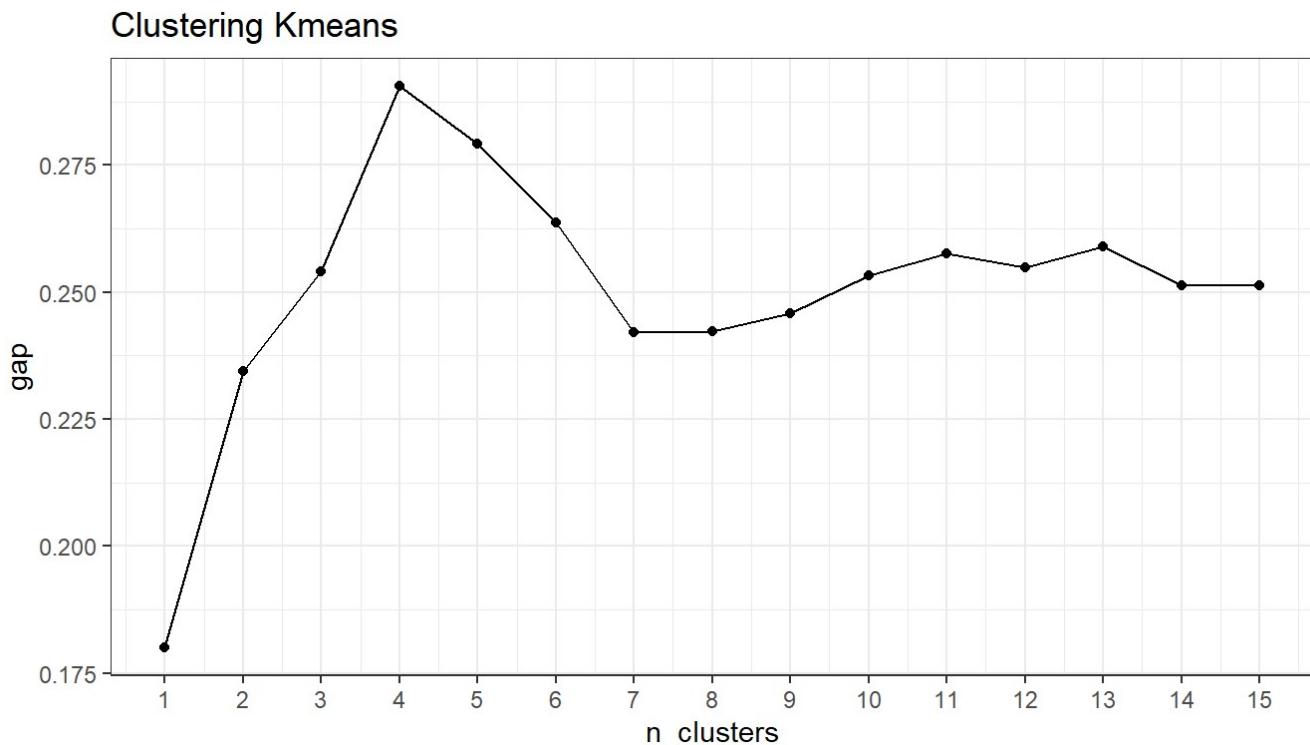
[Hide](#)

```
set.seed(896)
kmeans_gap <- clusGap(x = datos,
                        FUNcluster = kmeans,
                        K.max = 15,
                        B = 500,
                        verbose = FALSE,
                        nstart = 50)
kmeans_gap
```

```
## Clustering Gap statistic ["clusGap"] from call:  
## clusGap(x = datos, FUNcluster = kmeans, K.max = 15, B = 500, verbose = FALSE, n  
start = 50)  
## B=500 simulated reference sets, k = 1..15; spaceH0="scaledPCA"  
## --> Number of clusters (method 'firstSEmax', SE.factor=1): 3  
## logW E.logW gap SE.sim  
## [1,] 3.458369 3.638351 0.1799816 0.03934034  
## [2,] 3.135112 3.369586 0.2344744 0.03538104  
## [3,] 2.977727 3.231814 0.2540873 0.03622365  
## [4,] 2.826221 3.116833 0.2906128 0.03679382  
## [5,] 2.738868 3.018052 0.2791840 0.03739620  
## [6,] 2.666967 2.930670 0.2637025 0.03829521  
## [7,] 2.609895 2.851991 0.2420959 0.03882508  
## [8,] 2.537521 2.779858 0.2423366 0.03954516  
## [9,] 2.466005 2.711888 0.2458827 0.04055586  
## [10,] 2.394884 2.648090 0.2532064 0.04177478  
## [11,] 2.329117 2.586737 0.2576205 0.04265796  
## [12,] 2.272276 2.527104 0.2548282 0.04371998  
## [13,] 2.210102 2.469100 0.2589976 0.04501197  
## [14,] 2.160878 2.412150 0.2512719 0.04660752  
## [15,] 2.104753 2.356131 0.2513786 0.04722798
```

[Hide](#)

```
kmeans_gap$Tab %>%  
as.data.frame() %>%  
rowid_to_column(var = "n_clusters") %>%  
ggplot(aes(x = n_clusters, y = gap)) +  
geom_line() +  
geom_point() +  
labs(title = "Clustering Kmeans") +  
scale_x_continuous(breaks = 1:20) +  
theme_bw()
```



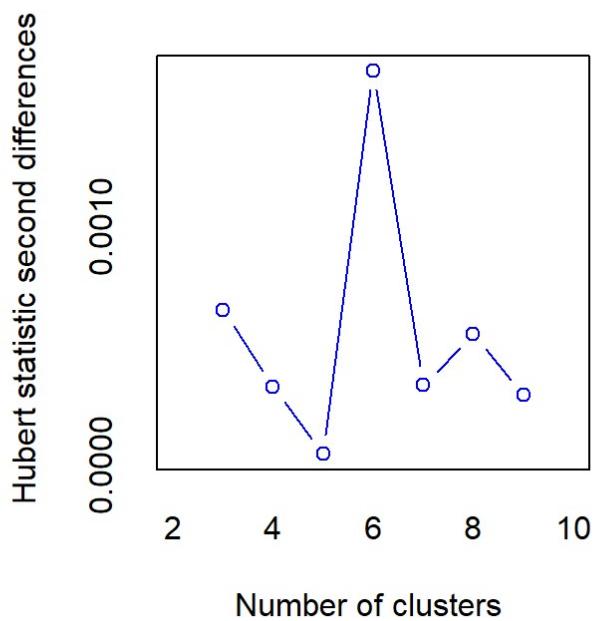
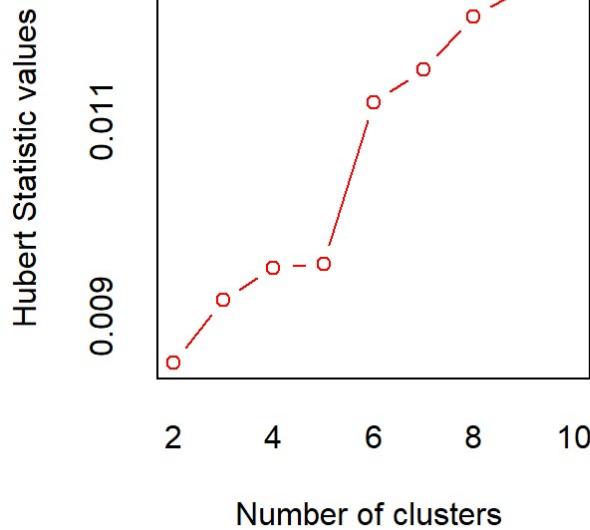
Los métodos Elbow, Silhouette y gap no tienen por qué coincidir exactamente en su estimación del número óptimo de clusters, pero tienden a acotar el rango de posibles valores. Por esta razón es recomendable calcular los tres y en función de los resultados decidir.

Además de estos tres métodos, existen en la bibliografía muchos otros desarrollados también para identificar el número óptimo de clusters. La función `NbClust()` del paquete NbClust incorpora 30 índices distintos, dando la posibilidad de calcularlos todos en un único paso. Esto último es muy útil, ya que permite identificar el valor en el que coinciden más índices, aportando seguridad de que se está haciendo una buena elección.

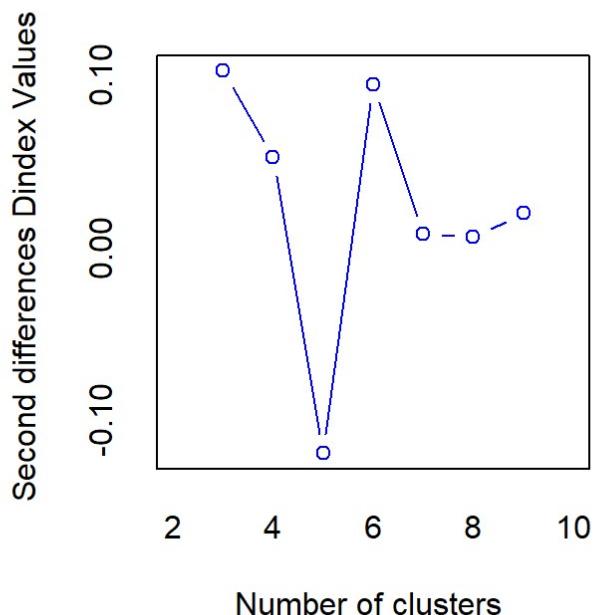
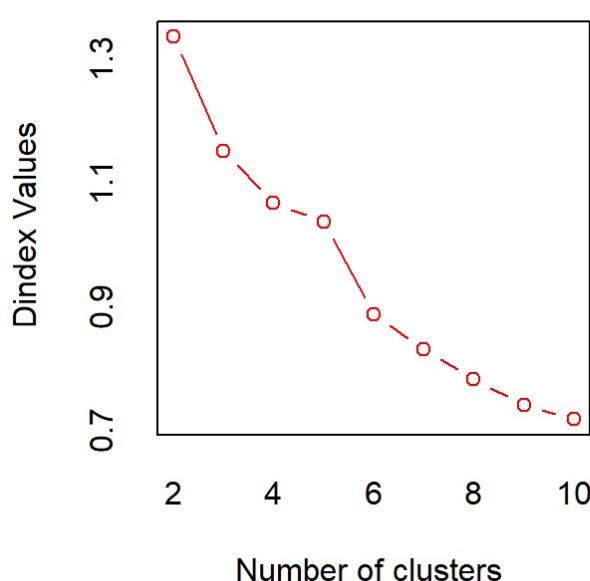
Hide

```
library(factoextra)
library(NbClust)

datos <- scale(USArrests)
numero_clusters <- NbClust(data = datos, distance = "euclidean", min.nc = 2,
                             max.nc = 10, method = "kmeans", index = "alllong")
```



```
## *** : 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.
##
```

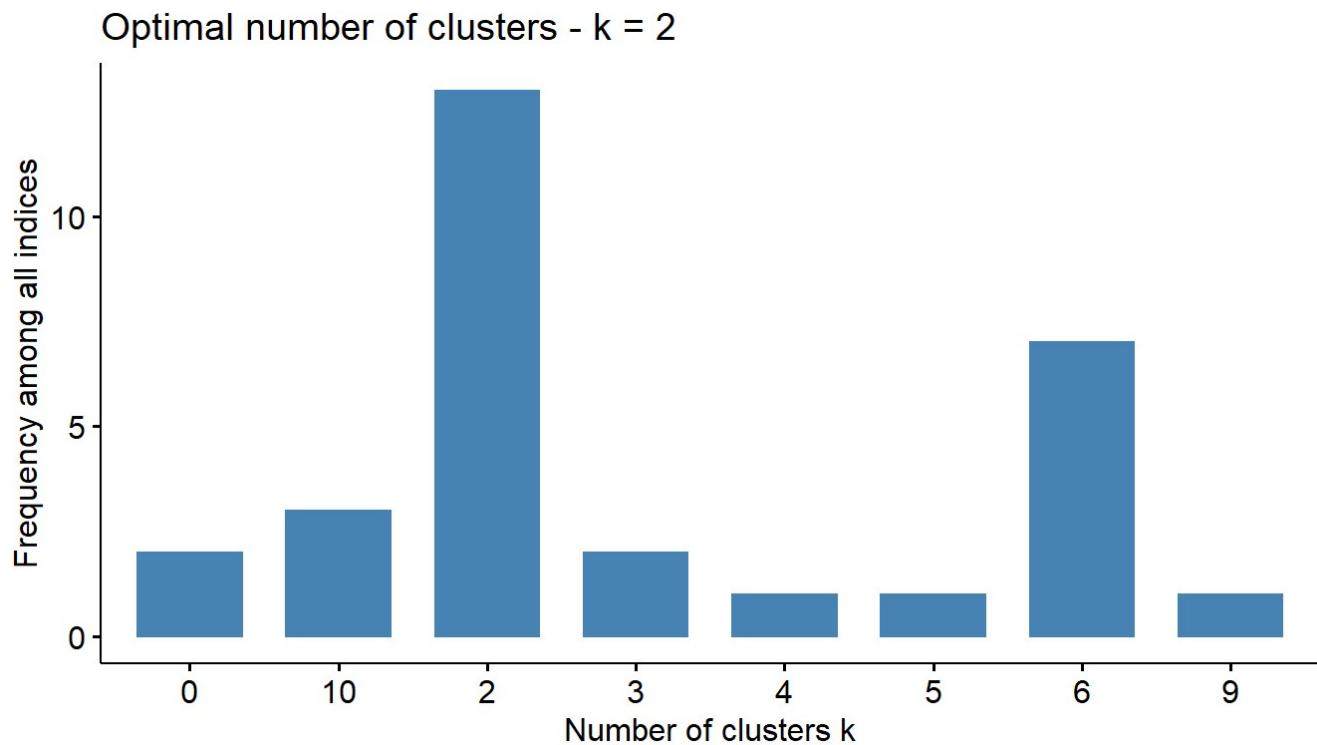


```
## *** : The D index is a graphical method of determining the number of clusters.  
## In the plot of D index, we seek a significant knee (the significant  
peak in Dindex  
## second differences plot) that corresponds to a significant increase  
of the value of  
## the measure.  
##  
## *****  
## * Among all indices:  
## * 13 proposed 2 as the best number of clusters  
## * 2 proposed 3 as the best number of clusters  
## * 1 proposed 4 as the best number of clusters  
## * 1 proposed 5 as the best number of clusters  
## * 7 proposed 6 as the best number of clusters  
## * 1 proposed 9 as the best number of clusters  
## * 3 proposed 10 as the best number of clusters  
##  
## ***** Conclusion *****  
##  
## * According to the majority rule, the best number of clusters is 2  
##  
## *****
```

[Hide](#)

```
fviz_nbclust(numero_clusters)
```

```
## Among all indices:  
## =====  
## * 2 proposed 0 as the best number of clusters  
## * 13 proposed 2 as the best number of clusters  
## * 2 proposed 3 as the best number of clusters  
## * 1 proposed 4 as the best number of clusters  
## * 1 proposed 5 as the best number of clusters  
## * 7 proposed 6 as the best number of clusters  
## * 1 proposed 9 as the best number of clusters  
## * 3 proposed 10 as the best number of clusters  
##  
## Conclusion  
## =====  
## * According to the majority rule, the best number of clusters is 2 .
```



11 Bibliografía

- https://rpubs.com/Joaquin_AR/ (https://rpubs.com/Joaquin_AR/)
- Introduction to Statistical Learning, Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
-Points of Significance: Clustering, Nature Methods, Martin Krzywinski & Naomi Altman
- Practical Guide to Cluster Analysis in R, Alboukadel kassambara
- Cluster Analysis for Gene Expression Data: A Survey. Dixin Jiang, Chun Tang, Aidong Zhang,
Department of Computer Science and Engineering
- Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? by
Fionn Murtagh y Pierre Legendre
- clValid, an R package for cluster validation. Guy Brock, Vasyil Pihur, Susmita Datta, and Somnath Datta
Department of Bioinformatics and Biostatistics, University of Louisville
- https://en.wikipedia.org/wiki/Jaccard_index (https://en.wikipedia.org/wiki/Jaccard_index)
- How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. C. Fraley
and A. E. Raftery
- <https://en.wikipedia.org/wiki/DBSCAN> (<https://en.wikipedia.org/wiki/DBSCAN>)