Minería de Datos para el Análisis de Big Data

Por: Carlos Carreño

ccarrenovi@Gmail.com

Abril,2021

Modulo 7 Modelos de Clustering

- Introducción
- Medidas de Distancia
- Escalar Variables
- Método K-means
- Método K-mediods
- Método CLARA
- Clustering Jerarquico

Introducción

- El termino Clustering se refiere a un conjunto de métodos no supervisados del machine learning cuya finalidad es encontrar patrones o grupos (clústeres) dentro de un grupo de observaciones.
- Se pueden identificar tres grupos de métodos:
  - ***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*).
  - ***Hierarchical Clustering***: Este tipo de algoritmos no requieren que el usuario especifique de antemano el número de *clusters*. (*agglomerative clustering*, *divisive clusterig*).
  - Métodos que combinan o modifican los anteriores (*hierarchical K-means*, *fuzzy clustering*, *model based clustering* y *density based clustering*).

Medidas de distancia

- El término **distancia** se emplea dentro del contexto del *clustering* como ***cuantificación de la similitud o diferencia*** entre observaciones.
- La característica que hace del *clustering* un método adaptable a escenarios muy diversos es que puede emplear cualquier tipo de distancia.

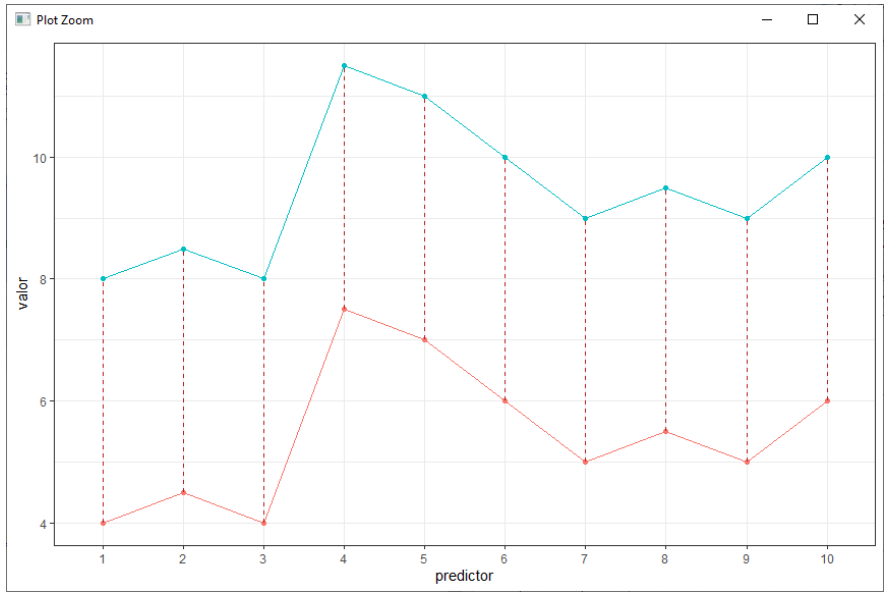
Distancia Euclidea

- La distancia euclídea entre dos puntos *p* y *q* se define como la longitud del segmento que une ambos puntos.
- En un espacio euclídeo n-dimensional donde cada punto está definido por un vector de n coordenadas: p=(p1,p2,p3,...,pn) y q=(q1,q2,q3,...,qn).

$$d_{euc}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} =$$

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Ejemplo: Distancia Euclidea




```r
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)
datos

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 ***euclídea*** 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.

Distancia de Manhattan

- La distancia de Manhattan, también conocida como *taxicab metric*, *rectilinear distance* o *L1 distance*, 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.

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

Correlación

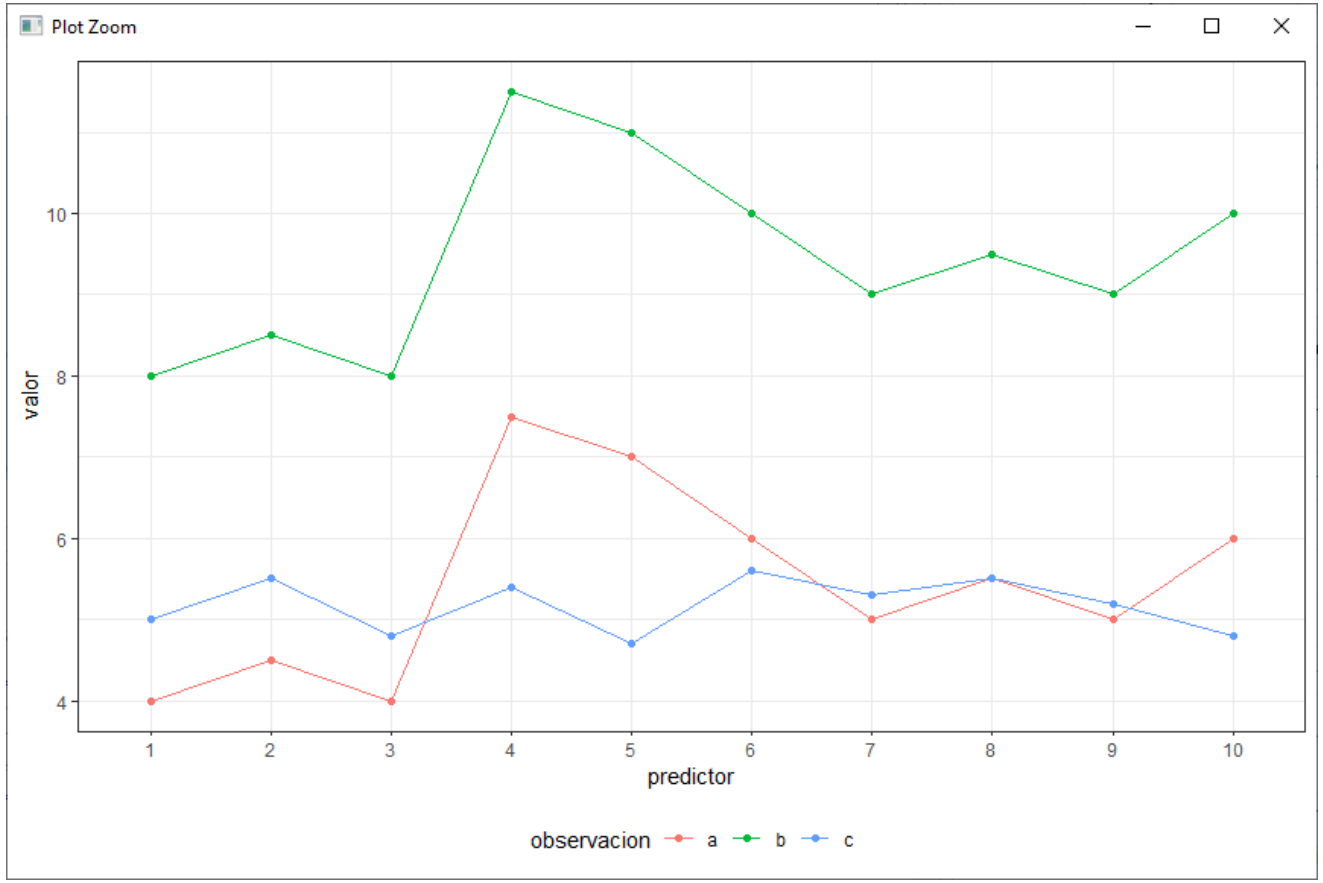
- La correlación es una medida de distancia muy útil cuando la definición de similitud se hace en términos de patrón o forma y no de desplazamiento o magnitud.

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

- Donde la correlación puede ser de distintos tipos (*Pearson*, *Spearman*, *Kendall*…)

Ejemplo: Correlación

- Que medidas son mas similares?



Correlación **Jackknife**

- 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.
- La correlacion *Jackknife*, 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 $\widehat{r_i}$ es el coeficiente de correlación entre las variables A y B, habiendo excluido la observación i.

- Error estándar y el intervalo de confianza

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

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

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

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

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 y el que se obtiene si se emplean todas las observaciones. A esta diferencia se le conoce como Bias.

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

- Si se calcula la diferencia entre cada correlación $\widehat{r_i}$ estimada en el proceso de Jackknife y el valor de correlación (r_i) obtenido si se emplean todas las observaciones, se puede identificar que 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.

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

Simple Matching Coefficient

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

$$SMC = \frac{\text{número coincidencias}}{\text{número total de atributos}} = \frac{M_{00} + M11}{M_{00} + M_{01} + M_{10} + M_{11}}$$

- 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.

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.
- *Jaccard* solo cuenta como coincidencias cuando el atributo está presente en ambos sets

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

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

…

- 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.
- Ejemplo:
  - 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.

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 0^0) su coseno toma el valor de 1, si son perpendiculares (forman un ángulo de 90^0) su coseno es 0 y si tienen orientaciones opuestas (ángulo de 180º) 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 ($X - \bar{X}$), la medida recibe el nombre de coseno centrado y es equivalente a la correlación de Pearson.

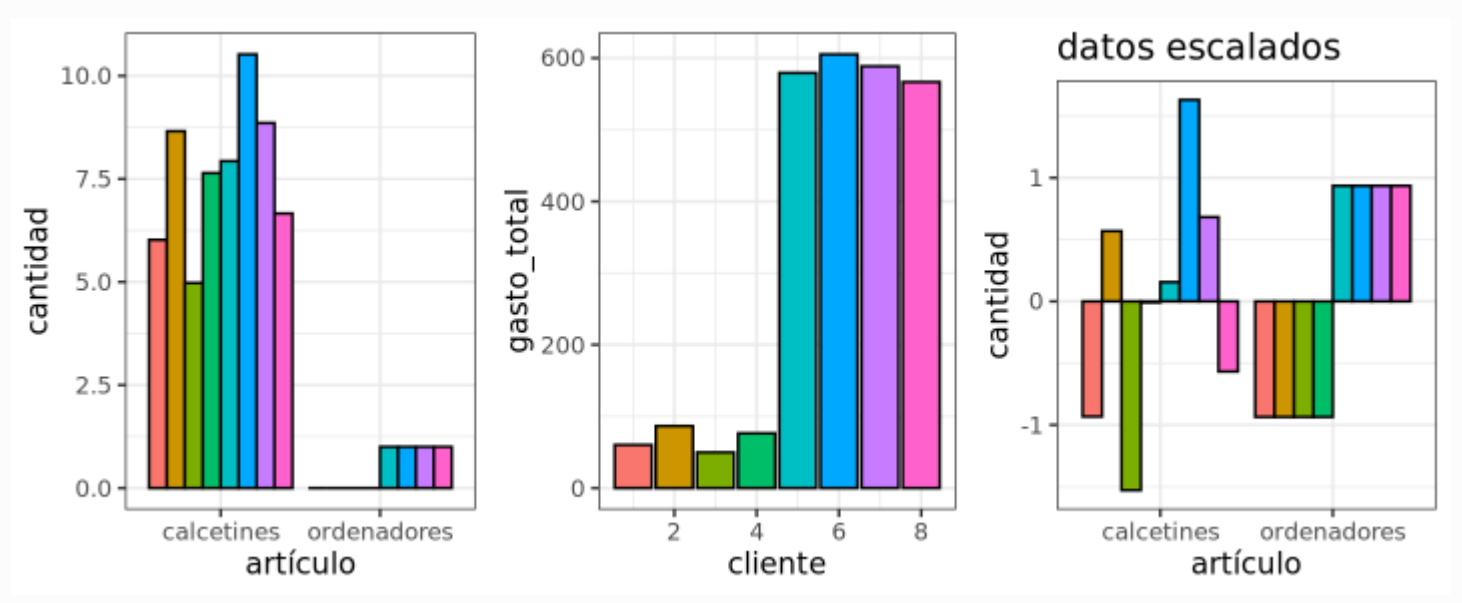
Escala de las Variables

- Al igual que en otros métodos estadísticos, ***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***.

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

Ejemplo:

- 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.



...

- 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 centrando 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.

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

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.

```r
#Ejemplo: Resumen de Medidas
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)

# 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)

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

install.packages("factoextra")
library(factoextra)
#La función fviz_dist del paquete factoextra resulta muy útil
#para generar representaciones gráficas (heatmap) de matrices de distancia.
fviz_dist(dist.obj = mat_dist, lab_size = 5) +
  theme(legend.position = "none")
```

Metodo K-means clustering

- El **método *K-means clustering*** (MacQueen, 1967) agrupa las observaciones en **K clusters distintos**, donde el número ***K* lo determina el analista** antes de ejecutar del algoritmo.
- *K-means clustering* encuentra los *K* mejores *clusters*, entendiendo **como *mejor cluster* aquel cuya varianza interna (*intra-cluster variation*) sea lo más pequeña** posible. Se trata por lo tanto de un problema de optimización, en el que se reparten las observaciones en *K clusters* de forma que la suma de las varianzas internas de todos ellos sea lo menor posible.

...

Considérense C_1,..., C_K como los sets formados por los índices de las observaciones de cada uno de los *clusters*. Por ejemplo, el set C_1 contiene los índices de las observaciones agrupadas en el *cluster* 1. La nomenclatura empleada para indicar que la observación i pertenece al *cluster* k es: $i \in C_k$. Todos los sets satisfacen dos propiedades:

- $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. Significa que toda observación pertenece al menos a uno de los *K clusters*.
- $C_k \cap C_{k'} = \emptyset$ para todo $k \neq k'$. Implica que los *clusters* no solapan, ninguna observación pertenece a más de un *cluster* a la vez.

Dos de las medidas más comúnmente empleadas definen la varianza interna de un *cluster* ($W(C_k)$) como:

- La suma de las distancias euclídeas al cuadrado entre cada observación (x_i) y el centroide (μ) de su *cluster*. Esto equivale a la suma de cuadrados internos del *cluster*.

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

- La suma de las distancias euclídeas al cuadrado entre todos los pares de observaciones que forman el *cluster*, dividida entre el número de observaciones del *cluster*.

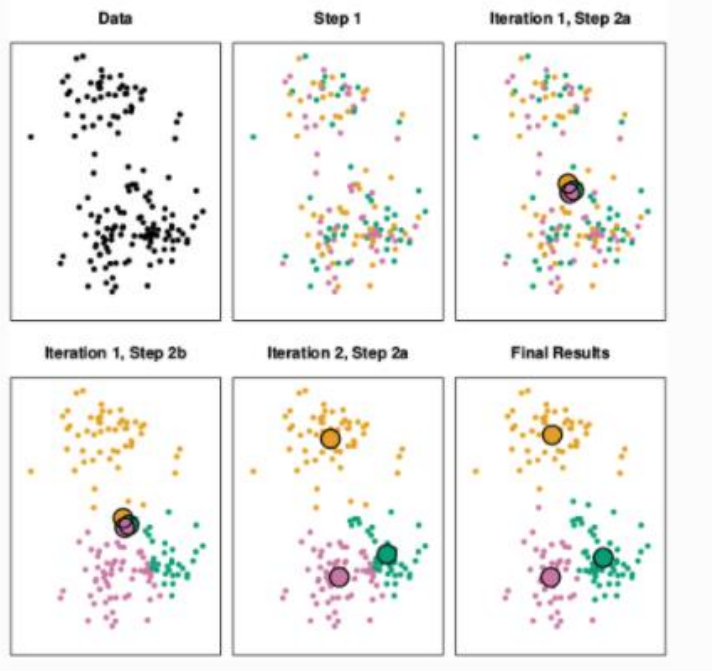
$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

Minimizar la suma total de varianza interna $\sum_{k=1}^{k} W(C_k)$ de forma exacta (encontrar el mínimo global) es un proceso muy complejo debido a la inmensa cantidad de formas en las que *n* observaciones se pueden dividir en *K* grupos. Sin embargo, es posible obtener una solución que, aun no siendo la mejor de entre todas las posibles, es muy buena (óptimo local). El algoritmo empleado para ello es:

---

1. Asignar aleatoriamente un número entre 1 y K a cada observación. Esto sirve como asignación inicial aleatoria de las observaciones a los *clusters*.
2. Iterar los siguientes pasos hasta que la asignación de las observaciones a los *clusters* no cambie o se alcance un número máximo de iteraciones establecido por el usuario.
    - 2.1 Para cada uno de los *clusters* calcular su centroide. Entendiendo por centroide la posición definida por la media de cada una de las dimensiones (variables) de las observaciones que forman el *cluster*. Aunque no es siempre equivalente, puede entenderse como el centro de gravedad.
    - 2.2 Asignar cada observación al *cluster* cuyo centroide está más próximo.

- Este algoritmo garantiza que, en cada paso, se reduzca la intra-varianza total de los *clusters* hasta alcanzar un *óptimo local*. La siguiente imagen muestra cómo van cambiando las asignaciones de las observaciones a medida que se ejecuta cada paso del algoritmo.



...

- Otra forma de implementar el algoritmo de K-means clustering es la siguiente:
  1. Especificar el número K de clusters que se quieren crear.
  2. Seleccionar de forma aleatoria k observaciones del set de datos como centroides iniciales.
  3. Asignar cada una de las observaciones al centroide más cercano.
  4. Para cada uno de los K clusters recalcular su centroide.
- Repetir los pasos 3 y 4 hasta que las asignaciones no cambien o se alcance el número máximo de iteraciones establecido.
- Debido a que el algoritmo de K-means no evalúa todas las posibles distribuciones de las observaciones sino solo parte de ellas, los resultados obtenidos dependen de la asignación aleatoria inicial (paso 1). Por esta razón, es importante ejecutar el algoritmo varias veces (25-50), cada una con una asignación aleatoria inicial distinta, y seleccionar aquella que haya conseguido un menor valor de varianza total.

Ventajas y Desventajas

- *K-means* es uno de los métodos de *clustering* más utilizados. Destaca por la sencillez y velocidad de su algoritmo, sin embargo, presenta una serie de limitaciones que se deben tener en cuenta.
  - Requiere que se indique de antemano el número de *clusters* que se van a crear. Esto puede ser complicado si no se dispone de información adicional sobre los datos con los que se trabaja. Se han desarrollado varias estrategias para ayudar a identificar potenciales valores óptimos de *K* (ver más adelante), aunque todas ellas son orientativas.
  - Las agrupaciones resultantes pueden variar dependiendo de la asignación aleatoria inicial de los centroides. Para minimizar este problema se recomienda repetir el proceso de *clustering* entre 25-50 veces y seleccionar como resultado definitivo el que tenga menor suma total de varianza interna. Aun así, solo se puede garantizar la reproducibilidad de los resultados si se emplean semillas.
  - Presenta problemas de robustez frente a *outliers*. La única solución es excluirlos o recurrir a otros métodos de *clustering* más robustos como *K-medoids (PAM)*.

Método **K-medoids clustering**

- *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* o ruido. A modo de idea intuitiva puede considerarse como la analogía entre media y mediana.

Algoritmo K-medoids

- El algoritmo más empleado para aplicar *K-medoids* se conoce como *PAM* (*Partitioning Around Medoids*) y sigue los siguientes pasos:
  1. Seleccionar *K* observaciones aleatorias como *medoids* iniciales. También es posible identificarlas de forma específica.
  2. Calcular la matriz de distancia entre todas las observaciones si esta no se ha calculado anteriormente.
  3. Asignar cada observación a su *medoid* más cercano.
  4. 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*.
  5. 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.

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, puden aplicarse en *K-medoids*.
- Para sets de datos grandes necesita muchos recursos computacionales.

Método CLARA

- 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*) es un método que combina la idea de *K-medoids* con el *resampling* 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.

Algoritmo del método 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:

   2.1 Aplicar el algoritmo *PAM* e identificar cuáles son los *k medoids*.

   2.2 Utilizando los *medoids* del paso anterior agrupar todas las observaciones del set de datos.

   2.3 Calcular 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 2.3.

Demo

- Realzar las practicas de :
- K-means
- K-medoids
- CLARA

Clustering Jerárquico

- ***Hierarchical clustering*** es una alternativa a los métodos de *partitioning clustering* que no requiere que se pre-especifique el número de *clusters*. Los métodos que engloba el *hierarchical clustering* se subdividen en dos tipos dependiendo de la estrategia seguida para crear los grupos:
  - ***Agglomerative clustering (bottom-up)*****:** el agrupamiento se inicia en la base del árbol, donde cada observación forma un *cluster* individual. Los *clusters* se van combinado a medida que la estructura crece hasta converger en una única "rama" central.
  - ***Divisive clustering (top-down)*****:** es la estrategia opuesta al *agglomerative clustering*, se inicia con todas las observaciones contenidas en un mismo *cluster* y se suceden divisiones hasta que cada observación forma un *cluster* individual.

Agglomerative

- La estructura resultante de un ***agglomerative hierarchical clustering*** se obtiene mediante un algoritmo sencillo.

1. El proceso se inicia considerando cada una de las observaciones como un cluster individual, formando así la base del dendrograma (hojas).
2. Se inicia un proceso iterativo hasta que todas las observaciones pertenecen a un único cluster:

   2.1 Se calcula la distancia entre cada posible par de los n clusters. El investigador debe determinar el tipo de medida emplea para cuantificar la similitud entre observaciones o grupos (distancia y linkage).

   2.2 Los dos clusters más similares se fusionan, de forma que quedan n-1 clusters.
3. Determinar dónde cortar la estructura de árbol generada (dendrograma).

- Sucesión de iteraciones para formar los clusters

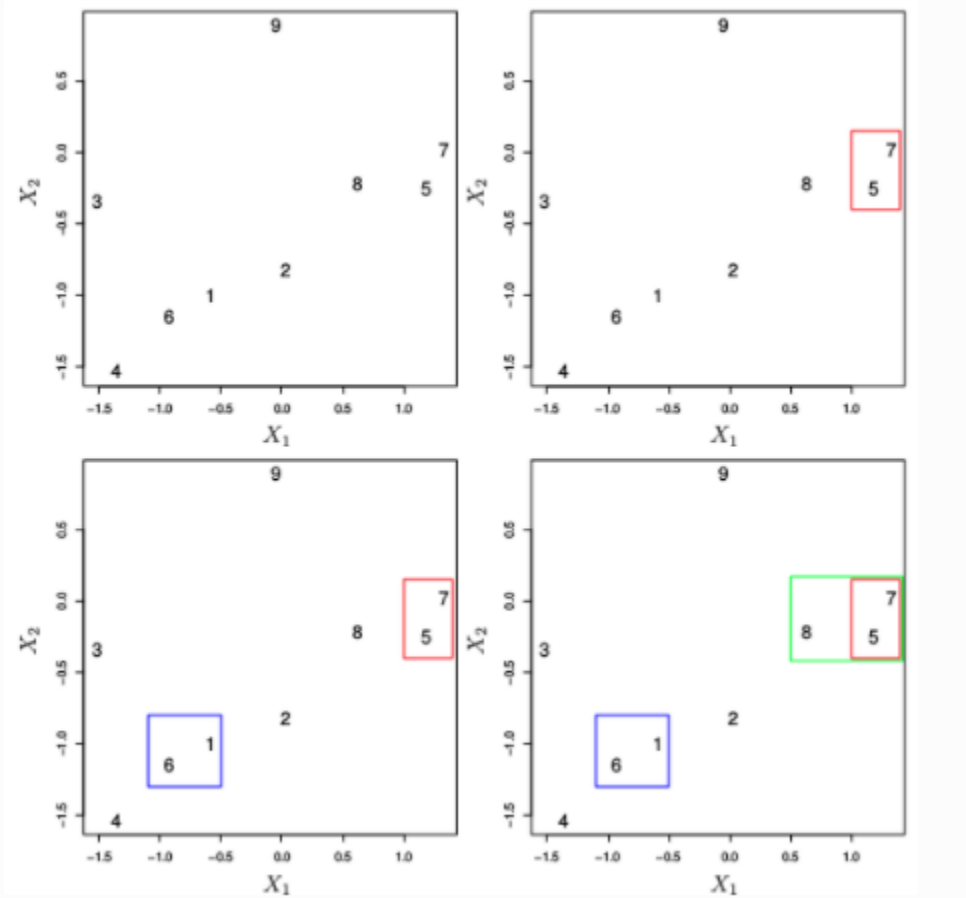


Imagen obtenida del libro ISLR

...

- Para crear el árbol de grupos, se tiene que extender el concepto de distancia entre pares de observaciones para que sea aplicable a pares de grupos, cada uno formado por varias observaciones. A este proceso se le conoce como *linkage*.

Tipos de Linkage

- **Complete or Maximum**: Se calcula la distancia entre todos los posibles pares formados por una observación del *cluster A* y una del *cluster B*. La mayor de todas ellas se selecciona como la distancia entre los dos *clusters*. Se trata de la medida más conservadora (*maximal intercluster dissimilarity*).
- **Single or Minimum**: Se calcula la distancia entre todos los posibles pares formados por una observación del *cluster A* y una del *cluster B*. La menor de todas ellas se selecciona como la distancia entre los dos *clusters*. Se trata de la medida menos conservadora (*minimal intercluster dissimilarity*).
- **Average**: Se calcula la distancia entre todos los posibles pares formados por una observación del *cluster A* y una del *cluster B*. El valor promedio de todas ellas se selecciona como la distancia entre los dos *clusters* (*mean intercluster dissimilarity*).
- **Centroid**: Se calcula el centroide de cada uno de los *clusters* y se selecciona la distancia entre ellos como la distancia entre los dos *clusters*.
- **Ward**: Se trata de un método general. La selección del par de *clusters* que se combinan en cada paso del *agglomerative hierarchical clustering* se basa en el valor óptimo de una función objetivo, pudiendo ser esta última cualquier función definida por el analista. El conocido método *Ward's minimum variance* es un caso particular en el que el objetivo es minimizar la suma total de varianza *intra-cluster*

Divisive

- El algoritmo más conocido de *divisive hierarchical clustering* es *DIANA* (*DIvisive ANAlysis Clustering*). Este algoritmo se inicia con un único *cluster* que contiene todas las observaciones, a continuación, se van sucediendo divisiones hasta que cada observación forma un *cluster* independiente.
- En cada iteración, se selecciona el *cluster* con mayor diametro, entendiendo por diámetro de un *cluster* la mayor de las diferencias entre dos de sus observaciones.
- Una vez seleccionado el *cluster*, se identifica la observación más dispar, que es aquella con mayor distancia promedio respecto al resto de observaciones que forman el *cluster*, esta observación inicia el nuevo *cluster*.
- Se reasignan las observaciones en función de si están más próximas al nuevo *cluster* o al resto de la partición, dividiendo así el *cluster* seleccionado en dos nuevos *clusters.*

...

1. Todas las *n* observaciones forman un único cluster.
2. Repetir hasta que hayan *n* clusters:

   2.1 Calcular para cada cluster la mayor de las distancias entre pares de observaciones (diámetro del cluster).

   2.2 Seleccionar el cluster con mayor diámetro.

      2.3 Calcular la distancia media de cada observación respecto a las demas.

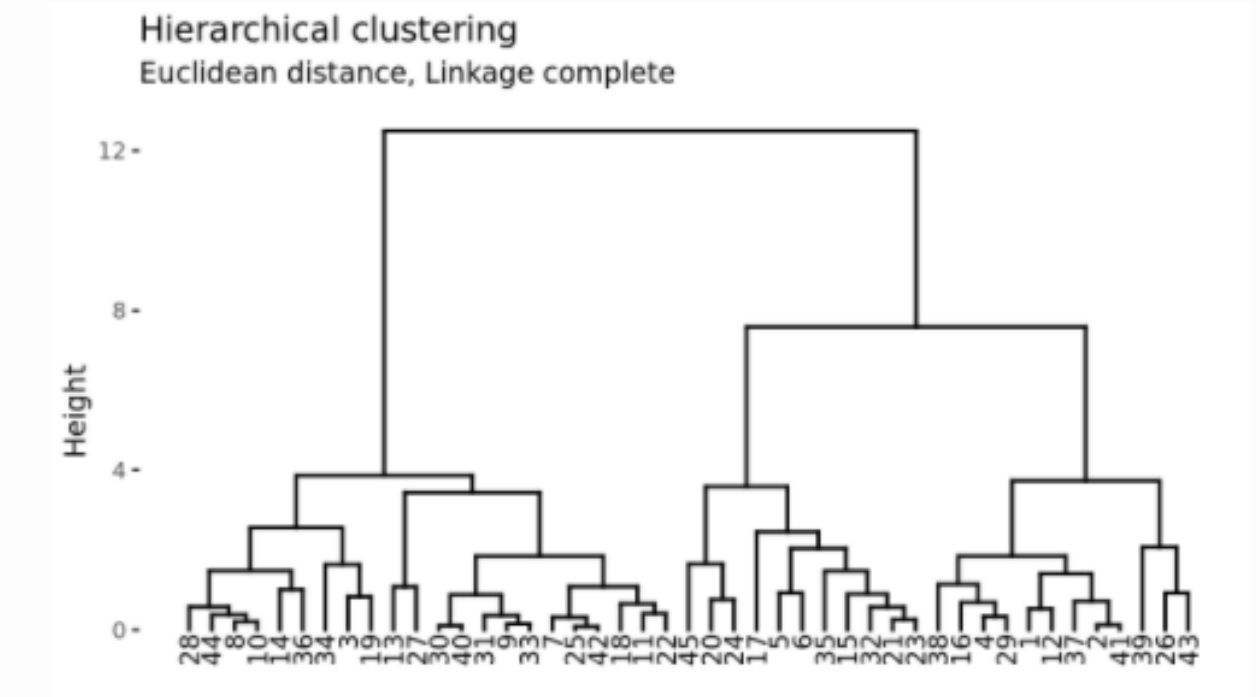
      2.4 La observación más distante inicia un nuevo cluster

      2.5 Se reasignan las observaciones restantes al nuevo cluster o al viejo dependiendo de cual está más próximo.

Nota: A diferencia del *clustering* aglomerativo, en el que hay que elegir un tipo de distancia y un método de *linkage*, en el *clustering* divisivo solo hay que elegir la distancia, no hay *linkage*.

Dendrograma

- Al aplicar *hierarchical clustering*, empleando como medida de similitud la distancia euclídea y *linkage complete*, se obtiene el siguiente dendrograma.



...

- En la base del dendrograma, cada observación forma una terminación individual conocida como hoja o *leaf* del árbol.
- A medida que se asciende por la estructura, pares de hojas se fusionan formando las primeras ramas. Estas uniones (nodos) se corresponden con los pares de observaciones más similares. También ocurre que ramas se fusionan con otras ramas o con hojas.
- Cuanto más temprana (más próxima a la base del dendrograma) ocurre una fusión, mayor es la similitud. Esto significa que, para cualquier par de observaciones, se puede identificar el punto del árbol en el que las ramas que contienen dichas observaciones se fusionan.
- La altura a la que esto ocurre (eje vertical) indica cómo de similares/diferentes son las dos observaciones. Los dendrogramas, por lo tanto, se deben interpretar únicamente en base al eje vertical y no por las posiciones que ocupan las observaciones en el eje horizontal, esto último es simplemente por estética y puede variar de un programa a otro.

Verificar el árbol resultante

- Una vez creado el dendrograma, hay que evaluar hasta qué punto su estructura refleja las distancias originales entre observaciones.
- Una forma de hacerlo es empleando el coeficiente de correlación entre las distancias cophenetic del dendrograma (altura de los nodos) y la matriz de distancias original.
- Cuanto más cercano es el valor a 1, mejor refleja el dendrograma la verdadera similitud entre las observaciones. Valores superiores a 0.75 suelen considerarse como buenos. Esta medida puede emplearse como criterio de ayuda para escoger entre los distintos métodos de linkage.
- En R, la función **cophenetic**() calcula las distancias cophenetic de un hierarchical clustering.

Ejemplo:

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

# Matriz de distancias euclídeas
mat_dist <- dist(x = datos, method = "euclidean")
# Dendrogramas con linkage complete y average
hc_euclidea_complete <- hclust(d = mat_dist, method = "complete")
hc_euclidea_average  <- hclust(d = mat_dist, method = "average")
cor(x = mat_dist, cophenetic(hc_euclidea_complete))
```

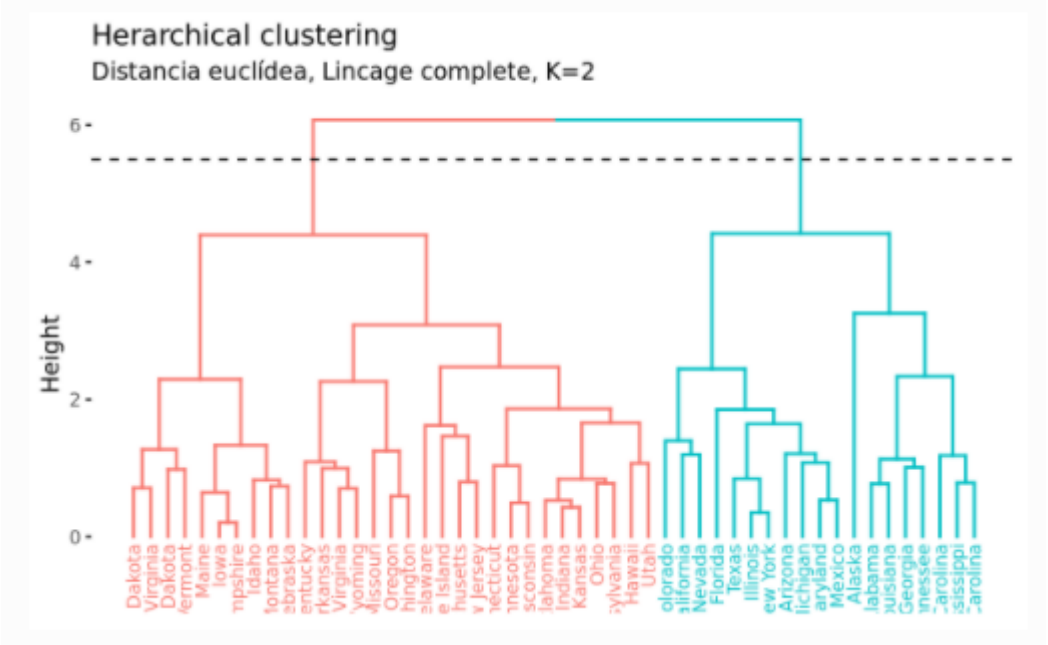
```
## [1] 0.6979437
```

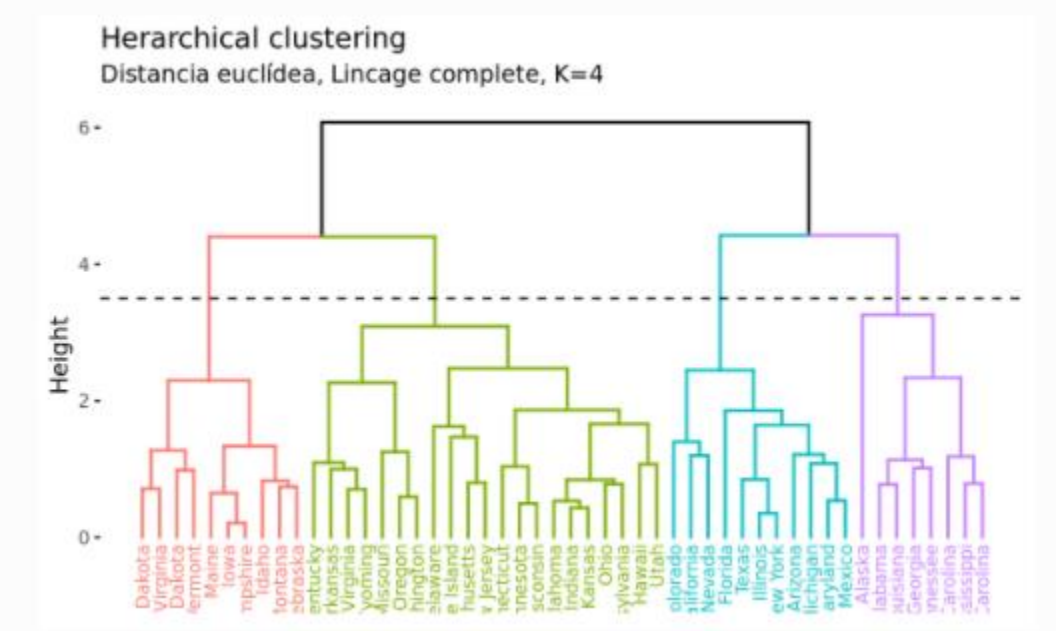
```
cor(x = mat_dist, cophenetic(hc_euclidea_average))
```

```
## [1] 0.7180382
```

Cortar el árbol para generar los clusters

- Además de representar en un dendrograma la similitud entre observaciones, se tiene que poder identificar el número de *clusters* creados y qué observaciones forman parte de cada uno. Si se realiza un corte horizontal a una determinada altura del dendrograma, el número de ramas que sobrepasan (en sentido ascendente) dicho corte se corresponde con el número de *clusters*.





Nota: Si se realiza el corte a la altura de 5, se obtienen dos *clusters*, mientras que si se hace a la de 3.5 se obtienen 4.

Hierarchical K-means clustering

- K-means es uno de los métodos de clustering más utilizados y cuyos resultados son satisfactorios en muchos escenarios, sin embargo, como se ha explicado en apartados anteriores, sufre las limitaciones de necesitar que se especifique el número de clusters de antemano y de que sus resultados puedan variar en función de la iniciación aleatoria. Una forma de contrarrestar estos dos problemas es combinando el K-means con el hierarchical clustering. Los pasos a seguir son los siguientes:

1. Aplicar hierarchical clustering a los datos y cortar el árbol en k clusters. El número óptimo puede elegirse de forma visual o con cualquiera de los métodos explicados en la sección Número óptimo de clusters.
2. Calcular el centro (por ejemplo, la media) de cada cluster.
3. Aplicar k-means clustering empleando como centroides iniciales los centros calculados en el paso

- El algoritmo de K-means tratará de mejorar la agrupación hecha por el hierarchical clustering en el paso 1, de ahí que las agrupaciones finales puedan variar respecto a las iniciales.

Fuzzy clustering

- 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 a al *cluster*.
  - Devuelve para cada observación la probabilidad de pertenecer a cada *cluster*.

Preguntas

- Alguna pregunta?