Análisis cluster

April 2024

Introducción

Objetivo

- `Objetivo`: cómo agrupar observaciones estableciendo grupos (o clusters) con las más similares.
- `Aprendizaje supervisado`: en la muestra `se indica a qué grupo pertenece cada observación`.
  - → Regresión Logística
  - → Análisis Discriminante
- `Aprendizaje no supervisado` (o automático): en este caso `no disponemos de una muestra inicial donde se indique a qué grupo pertenece cada observación`. De hecho, en algunas ocasiones podemos decidir cuántos grupos queremos establecer.
  - → Análisis Cluster

El contexto

- Dispondremos de `una muestra` (o población) de n individuos (objetos) en los que hemos medido k variables numéricas $(X_1, \ldots, X_k)$.
- Sin embargo, en este caso, `no dispondremos de una variable` Y que nos diga a qué grupo (población) pertenece cada observación.
- Incluso, en algunos casos, `no sabremos ni siquiera el número de grupos`.
- De hecho, lo que haremos será determinar los valores de Y que nos asigne los grupos que minimicen una `función costo` adecuada.
- Para ello tendremos que utilizar una función `distancia` que nos mida cómo de similares son dos observaciones (individuos).
- La `elección de esta distancia` es muy importante y la solución final dependerá de la distancia elegida.

Distancias entre individuos

La distancia Euclídea

- La distancia más popular utilizada es la `distancia Euclídea`, definida como

$$d_E(\mathbf{x}, \mathbf{c}) = \sqrt{(\mathbf{x} - \mathbf{c})'(\mathbf{x} - \mathbf{c})} = \sqrt{\sum_{j=1}^{k} (x_j - c_j)^2}$$

  para todo $\mathbf{x}, \mathbf{c} \in \mathbb{R}^k$ (vectores columna).

- En nuestro contexto, habitualmente $\mathbf{x} = (x_1, \ldots, x_k)'$ representará un `individuo` y $\mathbf{c} = (c_1, \ldots, c_k)'$ el `centroide` de un grupo.

- En R se puede computar como

```
dE <- function(x, y) sqrt(sum((x - y)*(x - y)))
```

- Por ejemplo, para $\boldsymbol{x} = (0, 0)'$ e $\boldsymbol{y} = (1, 1)'$:

```
x <- c(0, 0)
y <- c(1, 1)
dE(x, y)
```

```
[1] 1.414214
```

La distancia de Mahalanobis

- Otra opción es la `distancia de Mahalanobis` que usa la métrica de los datos, definida como

$$d_M(\mathbf{x}, \mathbf{c}) = \sqrt{(\mathbf{x} - \mathbf{c})' V^{-1} (\mathbf{x} - \mathbf{c})},$$

  donde $V = Cov(X_1, \ldots, X_k)$.
- El principal problema es que `si hay grupos`, esta matriz puede ser `distinta en cada grupo`.
- Incluso, aunque supongamos que todos los grupos tienen la misma matriz de covarianzas, estos tendrán medias distintas y, como desconocemos los grupos, no podemos estimar V (como hacíamos en el Análisis Discriminante).
- Una solución es suponer inicialmete que todos los individuos están en un mismo grupo (población) y calcular (estimar) la media y la covarianza en ella.

- En R se puede calcular usando la función `mahalanobis(x, y, V)`, que proporciona el cuadrado de esta distancia, para

$$V = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1 \end{pmatrix},$$

$x = (0, 0)'$ e $y = (1, 1)'$,

```r
V <- matrix(c(1, 1/2,
              1/2, 1), nrow = 2, ncol = 2, byrow = TRUE)
x <- c(0, 0)
y <- c(1, 1)
mahalanobis(x, y, V)
```

```
[1] 1.333333
```

- O bien, como

```r
dM <- function(x, y, V) sqrt(sum(t(x - y) %*% solve(V) %*% (x - y)))
dM(x, y, V)
```

```
[1] 1.154701
```

```r
## Si hacemos el cuadrado
dM(x, y, V)^2
```

```
[1] 1.333333
```

- Obviamente, si $V = I$ (matriz identidad), se obtiene la `distancia Euclídea` que, por lo tanto, representará a `vv.aa. independientes con varianza uno`.
- En otros casos, la distancia de Mahalanobis tendrá en cuenta las varianzas de las variables y sus covarianzas (correlaciones o dependencia).
- Las circunferencias (`elipsoides`) obtenidas con $d_V(\mathbf{x}, \boldsymbol{\mu}, V) = cte.$ coincidirán con los `conjuntos de nivel de la distribución normal multivariante` $\mathcal{N}_k(\boldsymbol{\mu}, V)$ cuya función de densidad es

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k |V|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})' V^{-1} (\mathbf{x}-\boldsymbol{\mu})\right).$$

  → De esta forma, bajo este modelo y si conocemos V, el individuo con medidas $\mathbf{x}$ se asignará al grupo en donde sea más verosímil, es decir, donde $f_i(x)$ sea máxima, siendo f_i la densidad $\mathcal{N}_k(\boldsymbol{\mu}_i, V)$ (tal y como hacíamos en AD).
- Ahora el problema es que no sabemos cómo estimar $\boldsymbol{\mu}_i$ y V y tampoco sabemos si hay una matriz de covarianzas V común.

Otras distancias interesantes

- La `distancia absoluta` (Manhattan, de ciudad o geometría del taxista)

$$d_A(\mathbf{x}, \mathbf{c}) = \sum_{j=1}^{k} |x_j - c_j|$$

(que usa las cuadrículas como caminos).

- La `distancia` L_s

$$d_s(\mathbf{x}, \mathbf{c}) = \left(\sum_{j=1}^{k} (x_j - c_j)^s \right)^{1/s}$$

para $s > 0$.

- La `distancia de Pearson`

$$d_P(\mathbf{x}, \mathbf{c}) = \sqrt{\sum_{j=1}^{k} \left(\frac{x_j - c_j}{\sigma_j} \right)^2}$$

donde σ_i es la desviación típica de X_i, $i = 1, \ldots, k$.

→ Este último caso es equivalente a estandarizar los datos usando $Z_i = X_i/\sigma_i$ o $Z_i = (X_i - \mu_i)/\sigma_i$ lo que nos asegura que las variables tendrán magnitudes similares aunque se usen unidades diferentes en ellas (esto no ocurre en la distancia Euclídea).

→ El principal problema es que desconocemos σ_i y μ_i que tendrán que ser estimados usando todos los datos (sin grupos).

→ Obviamente, es equivalente a usar la distancia Euclidea con los datos estandarizados.

→ La distancia no dependerá de las unidades usadas en cada variable (es invariante por cambio de escala).

Distancias de individuos a grupos y distancias entre grupos

Distancias de individuos a grupos

- Además de definir las distancias entre individuos, también tendremos que definir `distancias de individuos a grupos` o `distancias entre grupos`,
  - → lo que nos llevará a definir diversas funciones `coste` que determinarán diferentes soluciones finales.
  - → Estas vendrán determinadas por el problema que queremos resolver.
- Por ejemplo, si queremos calcular la `distancia de un individuo` $\mathbf{x}$ `a un grupo` $\{\mathbf{z}_i : i \in G\}$ formado por $m = |G|$ individuos podemos definir las distancias siguientes:

$$d_1(\mathbf{x}, G) := d(\mathbf{x}, \mathbf{C}), \quad \mathbf{C} = \frac{1}{|G|} \sum_{i \in G} \mathbf{z}_i$$

$$d_2(\mathbf{x}, G) := \min_{i \in G} d(\mathbf{x}, \mathbf{z}_i),$$

$$d_3(\mathbf{x}, G) := \max_{i \in G} d(\mathbf{x}, \mathbf{z}_i),$$

$$d_4(\mathbf{x}, G) := \sum_{i \in G} d(\mathbf{x}, \mathbf{z}_i),$$

$$d_5(\mathbf{x}, G) := \sum_{i \in G} d^2(\mathbf{x}, \mathbf{z}_i),$$

donde d es una distancia entre individuos.

- Otra opción interesante es calcular (o estimar) una función de densidad para los individuos de un mismo grupo y calcular las distancias como

$$d(\mathbf{x}, G_j) = 1 - \frac{f_j(\mathbf{x})}{f_1(\mathbf{x}) + \cdots + f_m(\mathbf{x})}.$$

Distancias entre grupos

- Análogamente, para las `distancias entre grupos` se pueden usar:

$$D_1(G_1, G_2) = d(C_1, C_2), \quad C_j = \frac{1}{|G_j|} \sum_{i \in G_j} \mathbf{z}_i, \quad j = 1, 2$$

$$D_2(G_1, G_2) = \min_{i \in G_1, j \in G_2} d(\mathbf{z}_i, \mathbf{z}_j),$$

$$D_3(G_1, G_2) = \max_{i \in G_1, j \in G_2} d(\mathbf{z}_i, \mathbf{z}_j),$$

$$D_4(G_1, G_2) = \frac{1}{|G_1||G_2|} \sum_{i \in G_1, j \in G_2} d(\mathbf{z}_i, \mathbf{z}_j),$$

$$D_5(G_1, G_2) = \frac{1}{|G_1||G_2|} \sum_{i \in G_1, j \in G_2} d^2(\mathbf{z}_i, \mathbf{z}_j).$$

- En d_1 o en D_1 podemos utilizar otros `centroides` C_1 y C_2 distintos de la media de cada grupo.
- Estas `distancias entre grupos` nos permitirán representar sus distancias y, posteriormente establecer a partir de qué nivel uniremos los grupos formando los gráficos denominados `dendogramas`.
- Finalmente debemos definir una `función costo` que trataremos de minimizar para obtener la solución óptima de ese problema.

Función costo

- Supongamos que asignamos los n individuos a un grupo mediante una variable Y que nos indicará con $y_i = j$ que el individuo i se asigna al grupo j.
- Podemos definir la función costo

$$J(y) = \sum_j \sum_{i:y_i=j} d(\mathbf{x}_i, G_j),$$

donde $\sum_{j:y_i=j} 1 = 1$ para todo i (cada elemento se asigna a un único grupo).

- También se pueden usar distancias al cuadrado

$$J^*(y) = \sum_j \sum_{i:y_i=j} d^2(\mathbf{x}_i, G_j).$$

- En estos métodos, tenemos que `fijar un número máximo de grupos` ya que si no, la solución óptima será tener n grupos (uno para cada elemento).
- `Otra opción` podría ser `maximizar la suma total de las distancias entre grupos` para la clasificación y:

$$D(y) = \sum_{i<j} D(G_i, G_j).$$

- Todas estas opciones nos llevarán a problemas diferentes que tendrán que resolverse (cuando sea posible) usando sus técnicas específicas (la mayoría de Investigación Operativa).

Métodos cluster

Clasificación

- Estos métodos se pueden dividir en dos grandes grupos:
  - → los `métodos jerárquicos`:
    - ↬ parten de la idea de juntar las unidades (individuos o grupos) más similares (cercanas).
  - → los `métodos no jerárquicos`:
    - ↬ establecen un determinado número de grupos y se irá asignando cada individuo al grupo más cercano.
- Solamente veremos un método de cada tipo.

Método no jerárquico de las K-medias

- El método de las K-medias (`K-means`) es sin duda el método no jerárquico `más popular`.
- Habitualmente usa la distancia Euclídea con los datos sin estandarizar (cuando tienen escalas similares) o estandarizados (distancia de Pearson, cuando tienen escalas diferentes) pero se puede aplicar a otras distancias.
- En este caso tenemos que `fijar un número de grupos predeterminado` K con $1 < K \leq n/2$.
- Posteriormente podremos aumentar o disminuir K según la solución obtenida.

- K es el número de grupos.
- k es el número de variables.
- n es el número de observaciones.
- Estos números pueden ser diferentes.

Algoritmo del método de las K-medias

- Paso 0: Determinar K centroides $C_1^0, \ldots, C_K^0 \in \mathbb{R}^k$ al azar.
- Paso 1: En la iteración m, formar el grupo G_j^m con las observaciones que están más cercanas al centroide C_j^{m-1} para $j = 1, \ldots, K$.
- Paso 2: Calcular el centroide C_j^m del grupo G_j^m definido como el punto que minimiza

$$\sum_{i \in G_j^m} d(\mathbf{x}_i, C_j^m),$$

  o considerando las distancias al cuadrado

$$\sum_{i \in G_j^m} d^2(\mathbf{x}_i, C_j^m),$$

  para $j = 1, \ldots, K$.
- Paso 3: Repetir pasos 1 y 2 hasta que no se produzcan cambios en los grupos del paso 1 o hasta que se haya iterado un número determinado de veces.

- Si usamos la distancia Euclídea y el error cuadrático, los `centroides del paso 2` serán las `medias aritméticas de los datos de cada grupo` ya que si queremos minimizar

$$\min_P \sum_{j \in G} d^2(\mathbf{O}_j, P)$$

para un grupo G, tenemos que

$$\begin{aligned}
\sum_{j \in G} d^2(\mathbf{O}_j, P) &= \sum_{j \in G} (\mathbf{O}_j - P)'(\mathbf{O}_j - P) \\
&= \sum_{j \in G} (\mathbf{O}_j - \overline{\mathbf{O}}_G + \overline{\mathbf{O}}_G - P)'(\mathbf{O}_j - \overline{\mathbf{O}}_G + \overline{\mathbf{O}}_G - P) \\
&= |G|(\overline{\mathbf{O}}_G - P)'(\overline{\mathbf{O}}_G - P) + \sum_{j \in G} (\mathbf{O}_j - \overline{\mathbf{O}}_G)'(\mathbf{O}_j - \overline{\mathbf{O}}_G) \\
&\quad + 2 \sum_{j \in G} (\overline{\mathbf{O}}_G - P)'(\mathbf{O}_j - \overline{\mathbf{O}}_G)
\end{aligned}$$

- (Cont.)

$$\sum_{j\in G} d^2(\mathbf{O}_j, P) = |G|(\overline{\mathbf{O}}_G - P)'(\overline{\mathbf{O}}_G - P) + \sum_{j\in G}(\mathbf{O}_j - \overline{\mathbf{O}}_G)'(\mathbf{O}_j - \overline{\mathbf{O}}_G)$$
$$+2(\overline{\mathbf{O}}_G - P)'\sum_{j\in G}(\mathbf{O}_j - \overline{\mathbf{O}}_G)$$
$$= |G|(\overline{\mathbf{O}}_G - P)'(\overline{\mathbf{O}}_G - P) + \sum_{j\in G}(\mathbf{O}_j - \overline{\mathbf{O}}_G)'(\mathbf{O}_j - \overline{\mathbf{O}}_G),$$

donde $\overline{\mathbf{O}}_G = \frac{1}{|G|}\sum_{j\in G}\mathbf{O}_j$ es la media del grupo G.

- En la expresión final, el segundo sumando es constante (no depende de P) y el mínimo del primer sumando se alcanza con $P = \overline{\mathbf{O}}_G$ (ya que es la distancia entre esos dos puntos).
- De esta forma el paso 2 es inmediato y en el paso 1 simplemente calculamos las distancias a estos nuevos K centroides (medias) asignando cada individuo al grupo del centroide más cercano (distancia d_1).

- El nombre `k-means` proviene de esta propiedad.
- Además, si usamos como función de coste las dadas previamente y hay cambios en los grupos, esta función es estrictamente decrecientes en el paso 1 ya que hay al menos un `objeto` cuya distancia al grupo (centroide) ha disminuido.
- Al recalcular los centroides en el paso 2 las sumas de estas distancias en los grupos disminuirán aún más o se quedarán iguales a las del paso 1.
- Como las `opciones del paso 1` son `finitas` ,
  - → este algoritmo conducirá hasta una solución óptima local en un número finito de pasos, que puede ser muy grande.
  - → Para `evitar este problema` podemos aplicar el algoritmo varias veces con centroides iniciales diferentes y comparar las soluciones óptimas finales de cada algoritmo.
- Veremos que con unos pocos pasos podemos obtener soluciones muy buenas.

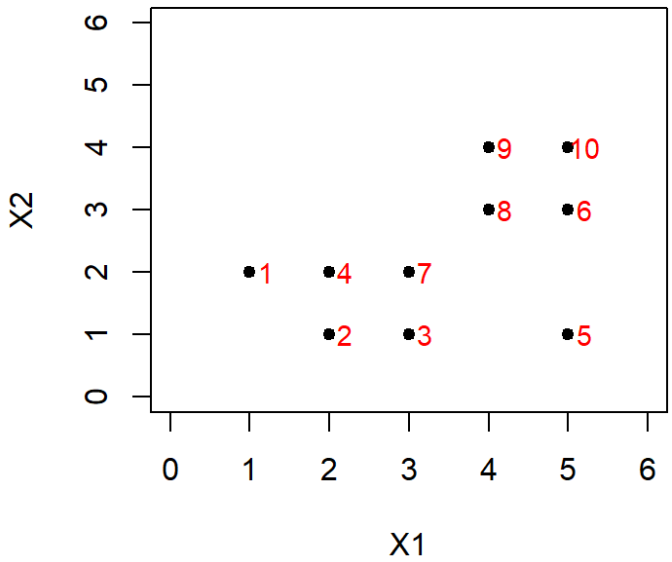
Un ejemplo sencillo

- Mostramos con un ejemplo sencillo el funcionamiento del algoritmo.
- Para ello usaremos los datos analizados previamente con regresión logística pero ahora supondremos que no conocemos los grupos de esa muestra.
- Supongamos que tenemos dos variables predictoras X_1 y X_2 (k = 2) y los datos siguientes:

Individuo i	X_1	X_2	Y
1	1	2	
2	2	1	
3	3	1	
4	2	2	
5	5	1	
6	5	3	
7	3	2	
8	4	3	
9	4	4	
10	5	4	

- Individuos sin agrupamiento inicial.

▶ Code



- Observamos que `tienen unidades similares` (por lo que podremos usar la distancia Euclídea) y que parecen formar dos grupos diferentes.
- El `objetivo` es determinar la variable Y que nos asigne cada individuo a un grupo.
- En tabla anterior, hemos dejado en blanco la columna de la variable Y para señalar que en este caso no tenemos una muestra de entrenamiento y, por lo tanto, no podremos saber cuál es la solución óptima (que mejor clasifique a los individuos).

  → `Análisis no supervisado` (o automático).

- Para aplicar el algoritmo con $K = 2$ medias (grupos) elegimos dos centroides al azar (dentro de la zona donde están los individuos).
- Para ello usamos la instrucción `runif(2,0,6)` (fijando previamente la semilla con `set.seed`).
- Primer centroide en el paso 0:

```
set.seed(123124)
C1_0 = runif(2, 0, 6)
C1_0
```

```
[1] 3.101323 1.401716
```

- Segundo centroide en el paso 0:

```
set.seed(123121)
C2_0 = runif(2, 0, 6)
C2_0
```

```
[1] 2.2950230 0.3726236
```

- Otra opción sería usar dos de esos puntos al azar.

- Con los centroides obtenidos, $C_1^0 = (3.101323, 1.401716)$ y $C_2^0 = (2.2950230, 0.3726236)$, obtenemos las distancias y agrupaciones siguientes:

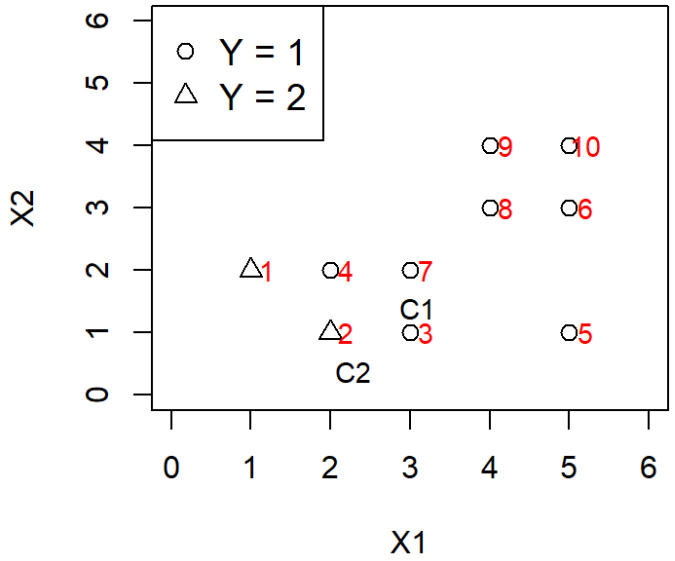
▶ Code

Individuo i	X_1	X_2	$d(\mathbf{X}, C_1^0)$	$d(\mathbf{X}, C_2^0)$	Y
1	1	2	2.1848343	2.0797689	2
2	2	1	1.1723001	0.6932819	2
3	3	1	0.4142971	0.9437127	1
4	2	2	1.2533377	1.6539022	1
5	5	1	1.9407090	2.7767790	1
6	5	3	2.4818314	3.7709425	1
7	3	2	0.6068031	1.7735125	1
8	4	3	1.8336119	3.1321005	1
9	4	4	2.7493091	4.0080926	1
10	5	4	3.2180825	4.5249044	1

- Cada individuo se asigna al grupo más cercano (midiendo su distancia a cada centroide).

- Individuos agrupados en el primer paso del algoritmo `K-means` con los centroides iniciales (negro).

▶ Code



- En el siguiente paso, calculamos lo nuevos centroides, que son las medias de los individuos de cada grupo.

▶ Code

```
[1] 3.875 2.500
```

▶ Code

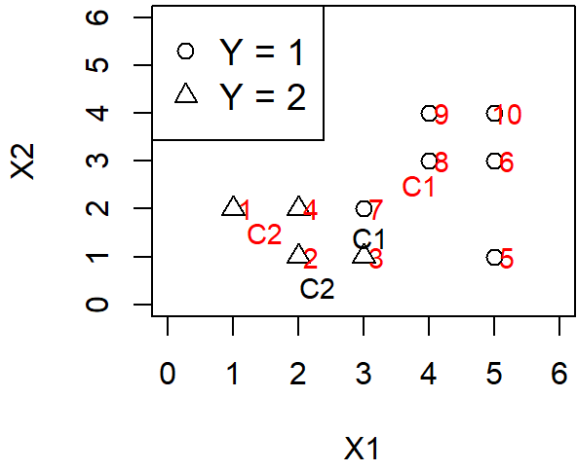
```
[1] 1.5 1.5
```

- Estos centroides son:

$$C_1^1 = (3.875, 2.5)$$
$$C_2^1 = (1.5, 1.5).$$

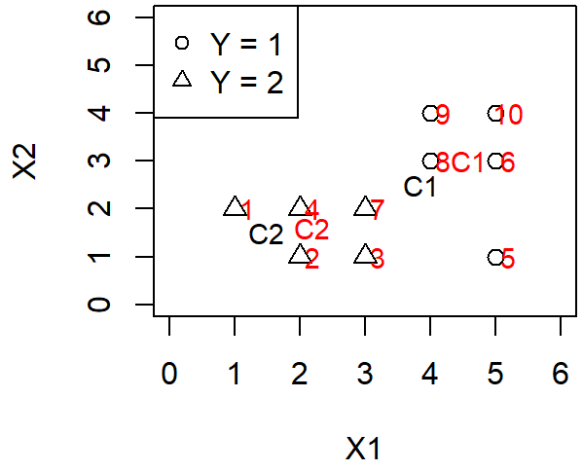
- Individuos agrupados en el primer y segundo paso del algoritmo K-means con los centroides iniciales (negro) y los nuevos (rojo).

▶ Code



- Repetimos los cálculos con los nuevos centroides. El reparto del paso uno conduce a los mismos grupos que en la iteración anterior y el algoritmo se detiene, obteniendo los centroides finales: $C_1^2 = (4.6, 3.0)$ y $C_2^2 = (2.2, 1.6)$.

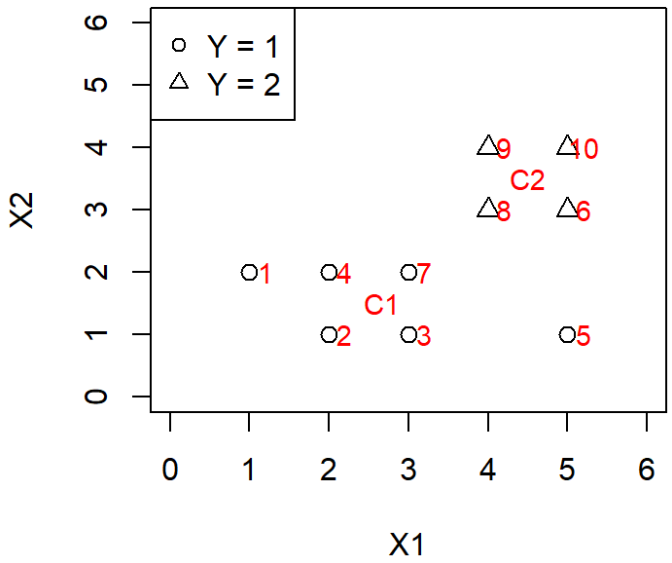
▶ Code



- Cuando los grupos no varían en dos iteraciones seguidas los centroides coinciden y el algoritmo se detiene.
- `Problema`: este algoritmo puede depender de los valores iniciales.
- Presentamos la agrupación de los individuos con otros centroides iniciales.

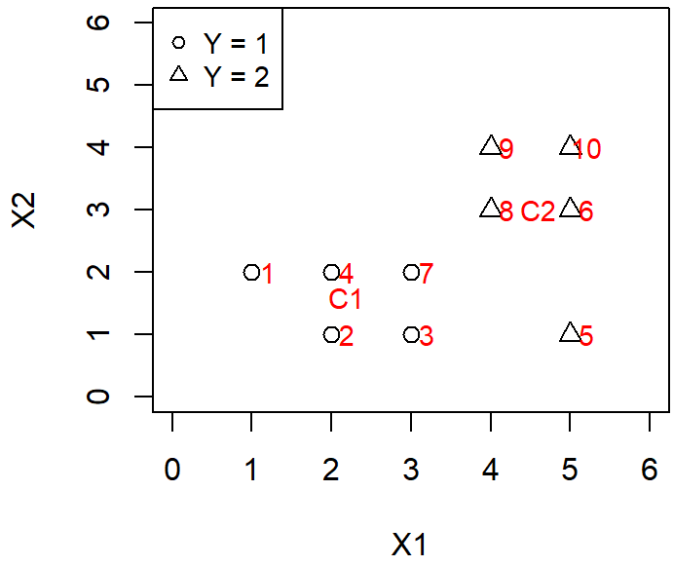
- Centroides iniciales: $C_1^0 = (2.482099, 2.270985)$ y $C_2^0 = (3.851084, 5.344700)$.
- Centroides finales: $C_1 = (2.66667, 1.5)$ y $C_2 = (4.5, 3.5)$.

▶ Code



- Centroides iniciales: $C_1^0 = (2, 3)$ y $C_2^0 = (5, 3)$.
- Centroides finales $C_1 = (2.2, 1.6)$ y $C_2 = (4.6, 3.0)$.

▶ Code



¿Cómo comparar distintas soluciones?

- Para comparar las soluciones podemos utilizar diversas medidas.
- Por ejemplo podíamos usar la función de costo J (con distancias Euclídeas).

▶ Code

- Para la solución $C_1 = (2.666667, 1.5)$ y $C_2 = (4.5, 3.5))$ $(sol1)$:

▶ Code

```
[1] 9.823299
```

$J(y_{sol1}) = 9.823299$.

- Para la solución $C_1 = (2.2, 1.6)$ y $C_2 = (4.6, 3.0)$ $(sol2)$:

▶ Code

```
[1] 9.521839
```

$J(y_{sol2}) = 9.521839$.

- Y para J^* (con distancias Euclídeas al cuadrado),

▶ Code

- Para la solución $C_1 = (2.666667, 1.5)$ y $C_2 = (4.500000, 3.5))$ $(sol1)$:

▶ Code

```
[1] 12.83333
```

$J^*(y_{sol1}) = 12.83333.$

- Para la solución $C_1 = (2.2, 1.6)$ y $C_2 = (4.6, 3.0)$ $(sol2)$:

▶ Code

```
[1] 11.2
```

$J^*(y_{sol2}) = 11.2.$

- En ambos casos la solución segunda parece dar mejores resultados:

$$J(y_{sol1}) = 9.823299 > J(y_{sol2}) = 9.521839,$$

$$J^*(y_{sol1}) = 12.83333 > J^*(y_{sol2}) = 11.2.$$

K-means se puede ejecutar de forma automática en R

- El algoritmo K-means se puede ejecutar de forma automática en R con el comando Kmeans.
- Por defecto, usa el algoritmo de Hartigan and Wong (1979).
- Para ejecutarlo en este ejemplo:

```
X1 <- c(1, 2, 3, 2, 5, 5, 3, 4, 4, 5)
X2 <- c(2, 1, 1, 2, 1, 3, 2, 3, 4, 4)
d <- data.frame(X1, X2)
CA <- kmeans(d, 2)
CA$centers
```

```
   X1  X2
1 4.6 3.0
2 2.2 1.6
```

- Los grupos coinciden con los obtenidos en la segunda solución anterior (óptima).
- También se pueden guardar

```
CA1 <- CA$centers[1,]
CA2 <- CA$centers[2,]
```

- Los grupos se obtienen con

```
CA$cluster
```

```
[1] 2 2 2 2 1 1 2 1 1 1
```

- La solución coincide con la representada en la gráfica.
- Las sumas de las distancias al cuadrado en los grupos se obtienen con

```
CA$withinss
```

```
[1] 7.2 4.0
```

obteniendo $\mathbf{7.2 + 4.0 = 11.2}$ (como antes).

- El comando

```
CA$totss
```

```
[1] 30.5
```

proporciona la suma de las distancias al cuadrado sin grupos (o con un único grupo) obteniéndose 30.5.

- Si se calcula directamente,

▶ Code

```
[1] 30.5
```

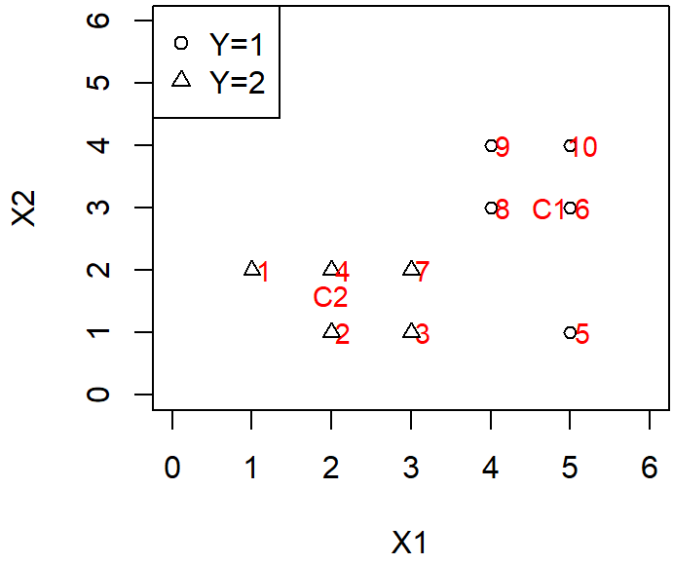
- Por lo que al agruparlos se ha producido una disminución del

$$1 - \frac{11.2}{30.5} = 0.6327869$$

por uno, es decir, con dos grupos la `variabilidad` se reduce un 63.28%.

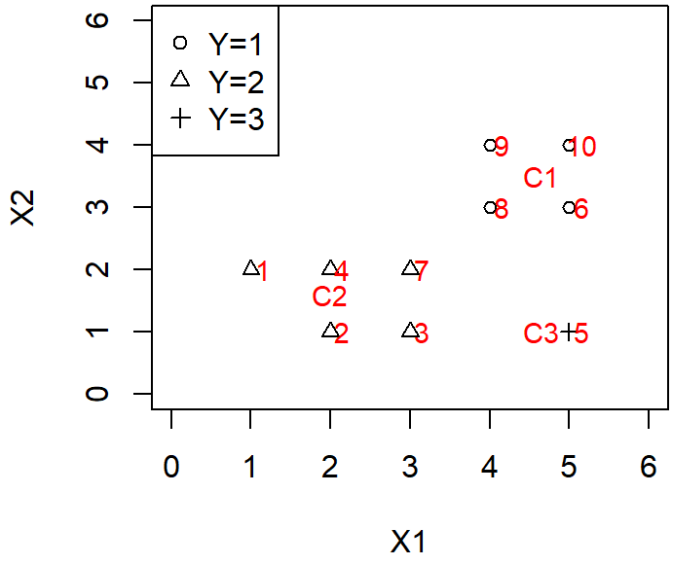
- Individuos agrupados con `Kmeans` en 2 grupos:

▶ Code



- Individuos agrupados con `Kmeans` en 3 grupos (hay un grupo que tiene un único dato, por lo tanto, será su centroide):

▶ Code



- Al ejecutar este comando pueden aparecer otras soluciones porque las `soluciones dependen de los valores iniciales`.
- La función `kmeans` permite ejecutar el algoritmo con diferentes valores de partida (argumento `nstart`) y proporcionar la mejor de esas soluciones.

```
CA <- kmeans(d, 3, nstart = 10)
```

- Con tres grupos la variabilidad se reduce en un 80.3 %.

```
CA = kmeans(d, 3, nstart = 10)
1- (sum(CA$withinss)/CA$totss)
```

```
[1] 0.8032787
```

Método jerárquico

Índice de similaridad

- En este caso no fijamos de antemano un número de grupos.
- Lo que haremos es, dada una distancia, definir un `índice de similaridad entre dos observaciones` con

$$I(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 1 - \frac{d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})}{\max\limits_{r,s} d(\mathbf{x}^{(r)}, \mathbf{x}^{(s)})} \in [0, 1].$$

- Se puede dar una definición análoga para grupos.
- La idea es establecer clasificaciones calculando los índices de similitud (o distancias) que se van obteniendo.
- Finalmente, dependiendo del índice de similitud elegido, obtendremos un número determinado de grupos (uniendo los que tienen similitud menor que ese índice).

Algoritmos

- Consideraremos dos algoritmos.
- En el primero `partiremos de` n `grupos formados por un individuo cada uno`.
  - → En el primer paso uniremos las dos observaciones más cercanas (distancia mínima) que serán las que tengan un índice de similitud mayor.
  - → Recalculamos las distancias para estos grupos y unimos los dos grupos más cercanos, continuamos así hasta conseguir un único grupo.
- En el segundo, procederemos de forma inversa, `partiremos de un único grupo` formado por todas las observaciones.
  - → En un primer paso separaremos en dos de forma que las distancias entre estos dos grupos sea máxima (o las distancias a esos dos grupos de sus individuos sea mínima).
  - → En el siguiente paso formaremos un tercer grupo tomando individuos de los grupos 1 y 2 con un criterio similar.
  - → Procederemos así hasta conseguir n grupos.
- Claramente, este método es más lento que el anterior.

Un ejemplo sencillo

- Para ver un ejemplo analizaremos los datos del ejemplo anterior usando el primer método.
- En primer lugar calculamos las distancias Euclídeas entre todos los individuos:

▶ Code

D	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}
O_1	0	1.41	2.24	**1**	4.12	4.12	2	3.16	3.61	**4.47**
O_2	1.41	0	1	1	3	3.61	1.41	2.83	3.61	4.24
O_3	2.24	1	0	1.41	2	2.83	1	2.24	3.16	3.61
O_4	1	1	1.41	0	3.16	3.16	1	2.24	2.83	3.61
O_5	4.12	3	2	3.16	0	2	2.24	2.24	3.16	3
O_6	4.12	3.61	2.83	3.16	2	0	2.24	1	1.41	1
O_7	2	1.41	1	1	2.24	2.24	0	1.41	2.24	2.83
O_8	3.16	2.83	2.24	2.24	2.24	1	1.41	0	1	1.41
O_9	3.61	3.61	3.16	2.83	3.16	1.41	2.24	1	0	1
O_{10}	**4.47**	4.24	3.61	3.61	3	1	2.83	1.41	1	0

- Observamos que el máximo se alcanza en $D_{1,10} = 4.47$ y el mínimo eliminando los ceros es 1 y se alcanza en varios puntos (esto se debe a que los puntos son discretos).
- El primero que detecta el programa es $D_{1,4} = 1$ por lo que será el primer grupo $G_1 = \{1, 4\}$.
- El índice de similaridad será

$$I(\mathbf{x}^{(1)}, \mathbf{x}^{(4)}) = 1 - \frac{d(\mathbf{x}^{(1)}, \mathbf{x}^{(4)})}{\max\limits_{r,s} d(\mathbf{x}^{(r)}, \mathbf{x}^{(s)})} = 1 - \frac{1}{4.472136} = 0.7763932.$$

- El siguiente paso dependerá de la distancia entre grupos elegida.

- Si queremos mantener esas distancias y detectar esos empates debemos elegir la distancia del `vecino más próximo` (D_2).
- Todas las demás nos darán valores mayores. Con esta distancia (tras varias iteraciones) uniríamos todos los puntos que están a distancia 1 de alguno del grupo obteniendo:

$$G_1 = \{1, 2, 3, 4, 7\},\ G_2 = \{5\}, G_3 = \{6, 8, 9, 10\}.$$

- La matriz de distancias D_2 para estos tres grupos será

D_2	G_1	G_2	G_3
G_1	0	2	$\mathbf{1.41}$
G_2	2	0	2
G_3	$\mathbf{1.41}$	2	0

- El mínimo se alcanza en $d(G_1, G_3) = d(\mathbf{x}^{(7)}, \mathbf{x}^{(8)}) = 1.41$.

- Por lo que en el segundo paso uniríamos los grupos G_1 y G_3 que tendrán un índice de similaridad

$$I(\mathbf{x}^{(7)}, \mathbf{x}^{(8)}) = 1 - \frac{d(\mathbf{x}^{(7)}, \mathbf{x}^{(8)})}{\max\limits_{r,s} d(\mathbf{x}^{(r)}, \mathbf{x}^{(s)})} = 1 - \frac{1.41}{4.472136} = 0.6847144.$$

- En el último paso uniríamos el grupo G_2 con $G_1 \cup G_3$ a distancia 2 y similaridad

$$I(\mathbf{x}^{(3)}, \mathbf{x}^{(5)}) = 1 - \frac{d(\mathbf{x}^{(3)}, \mathbf{x}^{(5)})}{\max\limits_{r,s} d(\mathbf{x}^{(r)}, \mathbf{x}^{(s)})} = 1 - \frac{2}{4.472136} = 0.5527864.$$

- El dendograma debe mostrar estas uniones usando las distancias o los índices de similaridad.
- Con otras distancias y/o usando el segundo método (que parte de un único grupo que se separa en dos) podemos obtener resultados diferentes.
- También observamos que los resultados no tienen por qué coincidir con los obtenidos con el algoritmo K-medias.
- La elección de un método u otro dependerá de los datos que tengamos y del problema que se quiera resolver (`costo`).
  - → Por ejemplo, si lo que queremos es agrupar a los usuarios para ser atendidos por centros deberemos usar distancias basadas en centroides que representarán dónde se situarán (aproximadamente) esos centros.
  - → Por contra, si lo que queremos es simplemente clasificar empresas o países según sus características, estos centroides no serán tan importantes.

¿Cómo realizar este agrupamiento de forma automática en R?

- Podemos usar la función `hclust`.
- En primer lugar calcularemos las distancias con

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

representadas en forma de vector.

- Para visualizarlas en forma de matriz usaremos `as.matrix(D)[1:10, 1:10]`.
- Ahora, para hacer un CA utilizamos la instrucción

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

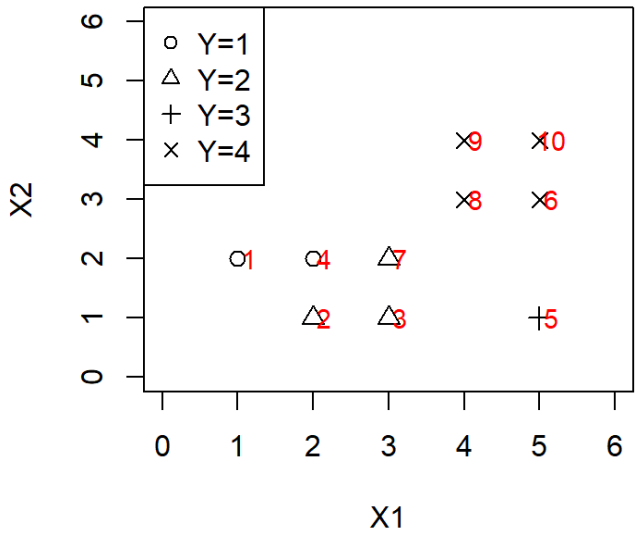
- Hemos usado el método de agrupación `complete` que usa la distancia del vecino más lejano (es el que usa R por defecto).
- Otras opciones son:
  - → `single`: vecino más cercano,
  - → `average`: media de todas las distancias entre todas las parejas de puntos de los dos grupos y
  - → `centroid`: distancia entre los centroides.

- Para $K = 4$ grupos

```
grupos <- cutree(CA2, k = 4)
```

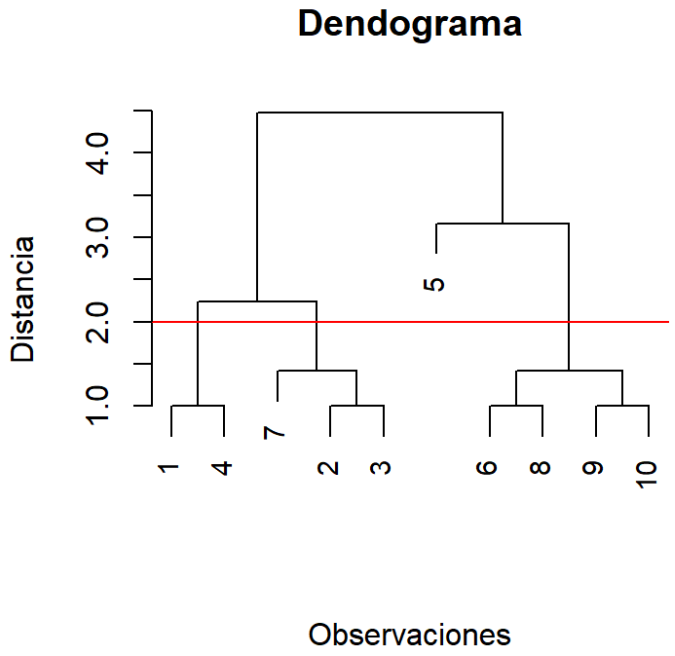
obtendremos los grupos:

▶ Code



- Representación del `dendograma`.
- La línea roja en el dendograma representa la distancia que nos da 4 grupos.

▶ Code



- En el dendograma primero se unen los puntos que están a menor distancia (en este caso era distancia 1) y, posteriormente se calculan las distancias entre grupos usando la distancia al vecino más lejano (D_3).
- Por ejemplo, la distancia de la observación 7 al grupo $\{2, 3\}$ es

```
1  dE(d[7, ], d[2, ])
```

```
[1] 1.414214
```

es decir, 1.414214.

- Lo mismo ocurre con la distancia entre los grupos $\{6, 8\}$ y $\{9, 10\}$. La distancia mayor es la de la observación 5 al grupo $\{6, 8, 9, 10\}$ obtenida con

```
1  dE(d[5, ], d[9, ])
```

```
[1] 3.162278
```

y que vale 3.162278.