Estadística Aplicada III

Métodos de agrupación: Conglomerados

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Semana 14

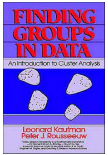
ITAM

Conglomerados (Clusters)

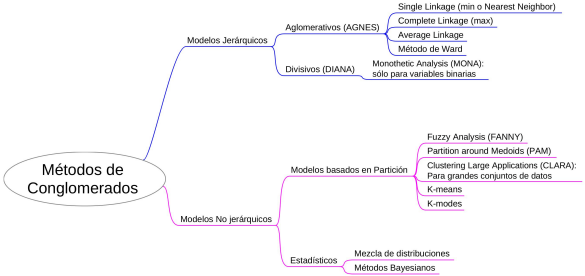
Introducción I

- El **análisis de conglomerados** (o *clustering*) tiene por objeto *descubrir formas de agrupar* elementos de una muestra en grupos homogéneos, en función de *las similitudes* entre ellos.
- Se puede aplicar a las observaciones de la muestra (*items*) de **X** o a las variables (las columnas de **X**).
- Se consideran un conjunto de *métodos de aprendizaje estadístico no supervisado*, pues no se conoce de antemano el número de grupos que se deben formar. Para determinar el número de grupos se examinan *algunas* de las posibles formas de agrupar las observaciones.

Hay muchos algoritmos de conglomerados que buscan agrupaciones maximales sin tener que buscar en todas las combinaciones posibles.
Algunos de los algoritmos se clasifican como se indica en la siguiente gráfica, y están programados en el paquete `cluster`, basados en el libro de Kaufman y Rousseeuw (1990).



Introducción II



- Los modelos son los siguientes:
  - **Métodos basados en partición**: (`pam`, `clara`, `fanny`): producen conglomerados para una K fija. Necesitan un cluster incial, tienen muchos posibles criterios para optimizar, algunos basados en modelos probabilísticos. Pueden tener grupo(s) distinto(s) 'outliers'.

Introducción III

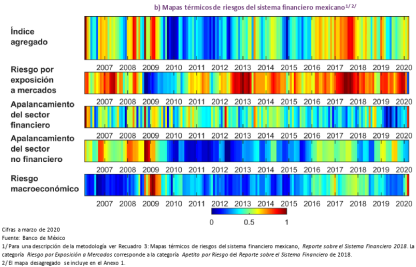
- **Métodos jerárquicos aglomerativos**: (`hclust`, `agnes`, `mclust`). Producen una sucesión de k grupos para cada $k = n, \dots, 1$ grupos que se amalgaman sucesivamente. Las principales diferencias son en el cálculo de las (di)similitudes de grupo a grupo a partir de las (di)similitudes de item a item. Son computacionalmente fáciles.
- **Métodos jerárquicos divisivos**: (`diana`, `mona`). Producen un conjunto de conglomerados, para cada $k = 1, \dots, K \ll n$. Computacionalmente, es casi imposible encontrar divisiones óptimas.

Advertencia:

- Los diferentes métodos de conglomerados pueden dar soluciones muy diferentes y esto puede llevar a veces a una sobreinterpretación.
- Los métodos de clustering **NO** son los mejores métodos para descubrir agrupaciones 'interesantes' de los datos. En combinación con métodos de visualización son más efectivos.
- Usualmente, el investigador conoce *a priori* si el agrupamiento producido es razonable o no.

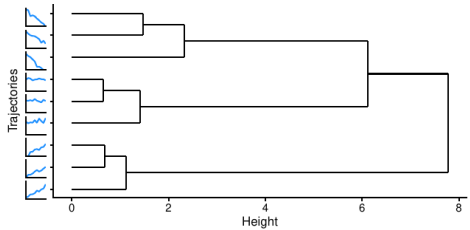
Aplicaciones I

- Minería de textos: Agrupación de textos por temas o contenidos.
- Psicología: formar conglomerados sobre los síntomas y características demográficas de pacientes deprimidos, para identificar subtipos de depresión, para encontrar tratamientos específicos.
- Mercadotecnia: técnicas de segmentación de consumidores para enfocar estrategias específicas. Ver Wedel y Kamakura, *Market Segmentation*, Kluwer, 2000.
- Medicina: se aplica para catalogar expresión de genes obtenida de datos de microarreglos.
- Estabilidad Financiera: se pueden obtener clasificación de factores de riesgo para identificar tiempos de alto riesgo vs zonas de menor riesgo.



Aplicaciones II

- Series de tiempo, análisis de datos funcionales: agrupación de trayectorias



Similitud/disimilitud

Características generales de la similitud I

- El concepto de similitud es bastante general y puede incluso ser subjetiva. Se puede definir para varios tipos de datos: cuantitativos, binarios, nominales, ordinales o mixtos.
- Una función de similitud se define sobre pares de observaciones, y debe simétrica, no negativa y creciente conforme los objetos son más similares.
- Se considera que una medida de similitud es inversamente proporcional a una medida de distancia. La distancia puede ser considerada como una medida de disimilitud.

Características generales de la similitud II

Ejemplos

- Presencia o ausencia de características: los objetos serán más similares si comparten más características. Por ejemplo:

Insecto	Vuela	tiene antenas	es venenoso	vive en comunidad
Abeja	1	0	1	1
Araña	0	0	1	0
Hormiga	0	1	0	1

- Para variables, medidas de asociación, como correlación son útiles.
- 12 marcas de yogurth evaluadas por 10 jueces en nueve variables. Los yogurths son presentados en pares a los panelistas a los que se les pide evaluar que tan similares son las dos muestras en una linea de escala descriptiva de 15cm.
- La similitud entre códigos Morse puede medirse como el porcentaje de veces que las personas confunden las sucesiones de símbolos después de escucharlos en una sucesión rápida.

Formalizando Similitud I

Definición

- Una *matriz de similitud* **C** es simétrica ($\mathbf{C}' = \mathbf{C}$) y tal que

$$0 \leq c_{ij} \leq c_{ii} \quad \forall i, j$$

- Una *matriz de disimilitud* **D** también es simétrica ($d_{ij} = d_{ji}$) y cumple:

$$d_{ii} = 0, \quad d_{ij} \geq 0 \quad i \neq j.$$

Si además se cumple la desigualdad del triángulo ($d_{ij} \leq d_{ik} + d_{kj}$), la disimilaridad es una distancia real.

- Con frecuencia se intercambian los coeficientes de similitud a disimilaridad y viceversa. Transformaciones incluyen:
  - (similaridad a disimilaridad) $d_{ij} = c - c_{ij}$ para alguna constante c.
  - (disimilaridad a similaridad) $c_{ij} = \frac{1}{1+d_{ij}}$

Formalizando Similitud II

- (similaridad a disimilaridad) La transformación estándar: $d_{ij} = (c_{ii} - 2c_{ij} + c_{jj})^{1/2}$
- No todas las disimilaridades son distancias reales.

Ejemplos I

A continuación consideraremos varios ejemplos de medidas, tomando en cuenta el tipo de variable (discreta, continua, binaria) y las escalas de medición (nominal, ordinal, de intervalo, de razón). Algunas de estas ya las hemos definido antes:

- Continuas:
  - **Distancia Euclideana:** La distancia usual para variables numéricas:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})'(\mathbf{x} - \mathbf{y})}$$

  - **Distancia de Mahalanobis:** Los datos se ponderan por su variabilidad:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})'\mathbf{S}^{-1}(\mathbf{x} - \mathbf{y})},$$

    aunque no siempre se conocen los grupos de antemano y por lo tanto no se puede estimar **S** (como en conglomerados).
  - **Norma supremo:**

$$d(\mathbf{x}, \mathbf{y}) = max|x_i - y_i|$$

Ejemplos II

- **Distancia de Minkowski:**

$$d_m(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^{p} |x_i - y_i|^m \right]^{1/m}$$

  Cuando $m = 1$ es la 'distancia Manhattan'.

- Para variables no negativas:
  - **métrica de Canberra:**

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} \frac{|x_i - y_i|}{x_i + y_i}$$

  - **coeficiente de Czekanowski:**

$$d(\mathbf{x}, \mathbf{y}) = 1 - \frac{2 \sum_{i=1}^{p} \text{mín}(x_i, y_i)}{\sum_{i=1}^{p} (x_i + y_i)}$$

- Binarias:

Ejemplos III

- **Presencia o ausencia de características:**

$$\sum_{i=1}^{p}(x_{ij} - x_{kj})^2,$$

donde:

$$(x_{ij} - x_{kj})^2 = \begin{cases} 0 & x_{ij} = x_{kj} = 1 \text{ o } x_{ij} = x_{kj} = 0 \\ 1 & x_{ij} \neq x_{kj} \end{cases}$$

aquí estamos comparando la $j-$ésima variable de los items i y k.

- distancias definidas en términos de correlación
  - **distancia correlación de Pearson**: Para dos variables x y y:

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

Esta distancia mide el grado de relación lineal entre dos variables del conjunto. También se puede usar la correlación de Spearman y la de Kendall como medidas de distancia, midiendo la distancia en términos de dependencia, como medidas no paramétricas.

Ejemplos IV

$$d_{spear}(x,y) = 1 - \frac{\sum_{i=1}^{n}(R(x_i) - \bar{R}(x))(R(y_i) - \bar{R}(y))}{\sqrt{\sum_{i=1}^{n}(R(x_i) - \bar{R}(x))^2 \sum_{i=1}^{n}(R(y_i) - \bar{R}(y))^2}}$$

donde $R(u)$ se refiere al rango de u. En el caso de la de Kendall:

$$d_{kendall}(x,y) = 1 - \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

donde n_c es el número de pares concordantes y n_d el número de pares discordantes.

- **similaridad coseno de Eisen basada en correlación**: Elimina la media de los datos

$$d_{\cos}(x,y) = 1 - \frac{|\sum_{i=1}^{n} x_i y_i|}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} = 1 - \frac{|\mathbf{x} \cdot \mathbf{y}|}{||\mathbf{x}|| ||\mathbf{y}||}$$

Fórmula de Gower I

- Siempre es posible construir similaridades a partir de distancias, con las transformaciones mencionadas antes.
- Sin embargo, disimilaridades que son distancias reales no siempre pueden construirse a partir de similitudes. Esto sólo se puede hacer si la matriz **C** es semidefinida positiva (Gower, 1971).
- Si $\mathbf{C} \geq \mathbf{0}$ y con la similitud máxima escalada de tal forma que $c_{ii} = 1$,

$$d_{ik} = \sqrt{2(1 - c_{ik})}$$

define una distancia. Esta es la fórmula de Gower.

Funciones en R para distancias I

- En R hay algunas funciones para calcular matrices de distancias a partir de datos:
  - la función `dist` puede calcular a partir de una matriz numérica o `data.frame` las distancias: `euclidean`, `max`, `manhattan`, `canberra`, `binary` o `minkoswki`:

```
x <- matrix(rnorm(100), nrow = 5)
dist(x, diag = T) # euclideana por default, no incluye la diagonal por default
          1        2        3        4        5
1 0.000000
2 6.788648 0.000000
3 6.964754 5.407102 0.000000
4 5.804607 6.833922 6.908577 0.000000
5 6.335322 6.869649 8.232067 8.113333 0.000000
dist(x, "canberra", diag = T)
          1        2        3        4        5
1  0.00000
2 15.33435  0.00000
3 15.15228 10.73517  0.00000
4 15.30676 11.61279 10.56333  0.00000
5 15.10393 14.89818 16.27519 16.40162  0.00000
dist(x, "binary", diag = T) # revisar definición de binary
  1 2 3 4 5
1 0
2 0 0
3 0 0 0
4 0 0 0 0
5 0 0 0 0 0
```

Funciones en R para distancias II

- La función `daisy` calcula matrices de disimilaridades en donde las variables pueden ser de tipos mezclados. En este caso, aplica una generalización de la transformación de Gower que se mencionó antes[17]:

```
library(cluster)
data(flower) # características de 18 flores, variables diferentes tipos
str(flower)
```

```
'data.frame': 18 obs. of  8 variables:
 $ V1: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 2 2 ...
 $ V2: Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 1 2 2 ...
 $ V3: Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 2 1 1 ...
 $ V4: Factor w/ 5 levels "1","2","3","4",..: 4 2 3 4 5 4 4 2 3 5 ...
 $ V5: Ord.factor w/ 3 levels "1"<"2"<"3": 3 1 3 2 2 3 3 2 1 2 ...
 $ V6: Ord.factor w/ 18 levels "1"<"2"<"3"<"4"<..: 15 3 1 16 2 12 13 7 4 14 ...
 $ V7: num  25 150 150 125 20 50 40 100 25 100 ...
 $ V8: num  15 50 50 50 15 40 20 15 15 60 ...
```

Se calcula la matriz de distancia considerando redondeo para simplificar el espacio:

Funciones en R para distancias III

```
round(daisy(flower, metric = "gower"), 2)
```

```
Dissimilarities :
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
2  0.89
3  0.53 0.51
4  0.35 0.55 0.57
5  0.41 0.62 0.37 0.64
6  0.23 0.66 0.30 0.42 0.34
7  0.29 0.60 0.49 0.34 0.42 0.19
8  0.42 0.46 0.60 0.30 0.47 0.57 0.41
9  0.58 0.43 0.45 0.81 0.33 0.51 0.59 0.64
10 0.61 0.45 0.47 0.56 0.38 0.41 0.59 0.66 0.43
11 0.33 0.71 0.60 0.65 0.39 0.48 0.57 0.50 0.43 0.39
12 0.43 0.59 0.60 0.51 0.50 0.52 0.64 0.42 0.42 0.38 0.26
13 0.52 0.52 0.54 0.75 0.29 0.45 0.53 0.58 0.22 0.36 0.34 0.23
14 0.29 0.59 0.61 0.37 0.52 0.37 0.50 0.46 0.44 0.36 0.28 0.16 0.38
15 0.62 0.39 0.53 0.55 0.46 0.51 0.33 0.45 0.25 0.42 0.48 0.43 0.32 0.44
16 0.69 0.36 0.62 0.34 0.73 0.51 0.44 0.64 0.65 0.35 0.74 0.61 0.59 0.46 0.39
17 0.78 0.19 0.58 0.42 0.69 0.59 0.52 0.47 0.61 0.31 0.70 0.56 0.55 0.54 0.35 0.17
18 0.46 0.45 0.72 0.44 0.48 0.64 0.47 0.14 0.52 0.81 0.54 0.55 0.57 0.57 0.51 0.78 0.61

Metric :  mixed ;  Types = N, N, N, N, O, O, I, I
Number of objects : 18
```

Funciones en R para distancias IV

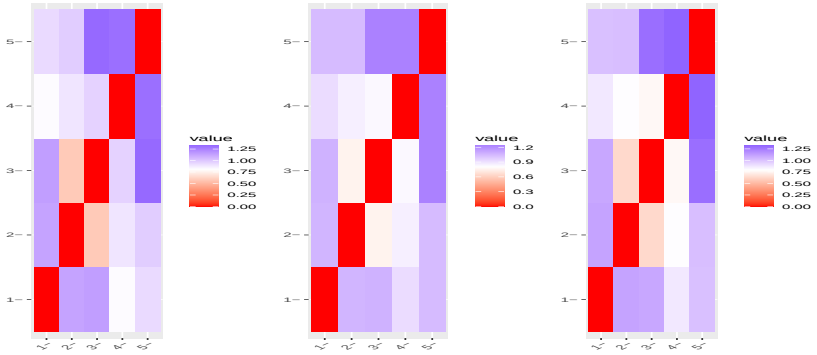
- La función `get_dist` del paquete `factoextra` permite calcular funciones basadas en correlación, incluyendo `pearson`, `kendall` y `spearman`

```
library(factoextra)
(dp <- get_dist(x, method = "pearson"))
          1         2         3         4
2 1.1173922
3 1.1379748 0.5810641
4 0.8157676 0.8904975 0.9552434
5 0.9276055 0.9731682 1.3224905 1.3022945
(dk <- get_dist(x, method = "kendall"))
          1         2         3         4
2 1.0526316
3 1.0631579 0.7473684
4 0.9157895 0.8526316 0.8210526
5 1.0315789 1.0315789 1.2315789 1.2315789
(ds <- get_dist(x, method = "spearman"))
          1         2         3         4
2 1.1022556
3 1.0872180 0.6330827
4 0.8661654 0.7954887 0.7609023
5 1.0015038 1.0075188 1.2842105 1.3187970
```

Funciones en R para distancias V

- Podemos visualizar la matriz de distancias con la función `fviz_dist`:

```
library(cowplot)  # Para arreglos de gráficas generadas por ggplot
plot_grid( fviz_dist(dp, order = F),
           fviz_dist(dk, order = F),
           fviz_dist(ds, order = F), ncol = 3)
```



Métodos de conglomerados jerárquicos

Métodos de conglomerados jerárquicos I

- El número de formas de particionar un conjunto de n elementos en g grupos puede aproximarse por el número $\#(n, g) \approx g^n/g!$. Por ejemplo, $\#(25, 10) \approx 2.8 \times 10^{18}$.
- Por lo anterior es necesario emplear un método eficiente que nos permita encontrar soluciones razonables sin necesidad de probar todas las configuraciones posibles.
- Los métodos jerárquicos aplican una serie de uniones o divisiones sucesivas.
  - Los métodos aglomerativos comienzan con los items individuales formando n clusters individuales, y se procede hasta que todos los items quedan en un sólo cluster.
  - Los métodos divisivos se mueven en dirección opuesta, de 1 cluster a n clusters.
- Los resultados de las divisiones o uniones sucesivas se grafican en un *dendrograma* (*dendron*, ($\delta\varepsilon\nu\delta\rho o\nu$), árbol)
- A continuación revisaremos ejemplos de algoritmos aglomerativos.

Algoritmos aglomerativos I

Puede usar distancia o similitud. Requiere un método de *enlace* (linkage). Comenzando con N items o variables,

1. Se consideran N clusters con un sólo item, y una respectiva matriz de distancias **D** $= \{d_{ij}\}$.
2. Buscar en **D** los pares más cercanos (similares) de clusters, U y V con distancia d_{UV}.
3. Se unen los clusters U y V y se etiqueta como un nuevo cluster (UV), y se actualiza la matriz para eliminar los renglones y columnas de distancias U y V y se agregan las correspondientes de (UV) a todos los otros clusters.
4. Repetir los pasos 1-3, $N - 1$ veces, ya que todos los items quedarán aglomerados en un sólo cluster. Registrar en cada iteración la identidad de los clusters que se unieron y los niveles de distancia en donde se hicieron las uniones.

- El criterio de enlace que se aplica en el paso 3 se puede definir de diferentes maneras:
  - **Enlace completo**: la distancia máxima entre pares de items en los diferentes clusters,

$$\text{máx}\{d_{uv}, u \in U, v \in V\}.$$

Algoritmos aglomerativos II

- **Enlace sencillo**: la distancia mínima entre pares de items en los diferentes clusters,

$$\text{mín}\{d_{uv}, u \in U, v \in V\}.$$

- **Enlace promedio**: la distancia promedio entre los items de los diferentes clusters,

$$\frac{\sum_{v \in V} \sum_{u \in U} d_{uv}}{|U||V|}.$$

Ejemplo conglomerados jerárquicos Aglomerativos I

Ejemplo. [Clustering con enlace sencillo]

- Supongamos que se tienen los 5 items: *a, b, c, d, e* con las siguientes distancias:

$$\mathbf{D} = \begin{bmatrix} \mathsf{a} & \mathsf{b} & \mathsf{c} & \mathsf{d} & \mathsf{e} \\ 0 & & & & \\ 9 & 0 & & & \\ 3 & 7 & 0 & & \\ 6 & 5 & 9 & 0 & \\ 11 & 10 & 2 & 8 & 0 \end{bmatrix}$$

Ejemplo conglomerados jerárquicos Aglomerativos II

- En la primera iteración se tienen los clusters: $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$ y $\{e\}$. Los más cercanos son e y c, así que se crea el cluster $\{ce\}$ y se actualiza la matriz de distancias, considerando el mínimo de las distancias de los clusters $\{a\}$, $\{b\}$ y $\{d\}$ al cluster $\{ce\}$. Agrego al final de la matriz el nuevo cluster y elimino los renglones correspondientes a $\{c\}$ y $\{e\}$:

$$\mathbf{D} = \begin{bmatrix} \text{a} & \text{b} & \text{d} & \text{ce} \\ 0 & & & \\ 9 & 0 & & \\ 6 & 5 & 0 & \\ 3 & 7 & 8 & 0 \end{bmatrix}$$

  Por ejemplo: $d(\{ce\}, a) = \text{mín}\{11, 3\} = 3$, $d(\{ce\}, b) = \text{mín}\{10, 7\} = 7$, $d(\{ce\}, d) = \text{mín}\{9, 8\} = 8$.

- En la segunda iteración se tienen los clusters: $\{a\}$, $\{b\}$, $\{d\}$ y $\{ce\}$. Los más cercanos ahora son $\{a\}$ y $\{ce\}$, así que se crea el cluster $\{ace\}$ y se actualiza la matriz de distancias, recalculando las distancias a $\{b\}$ y $\{d\}$ del cluster $\{ce\}$.

$$\mathbf{D} = \begin{bmatrix} \text{b} & \text{d} & \text{ace} \\ 0 & & \\ 5 & 0 & \\ 7 & 8 & 0 \end{bmatrix}$$

Ejemplo conglomerados jerárquicos Aglomerativos III

Por ejemplo: $d(\{ce\}, a) = \text{mín}\{11, 3\} = 3$, $d(\{ce\}, b) = \text{mín}\{10, 7\} = 7$, $d(\{ce\}, d) = \text{mín}\{9, 8\} = 8$.

- En la tercera iteración la distancia más corta es la de b a d, así que se crea el cluster $\{bd\}$ y se actualizan distancias a $\{ace\}$.

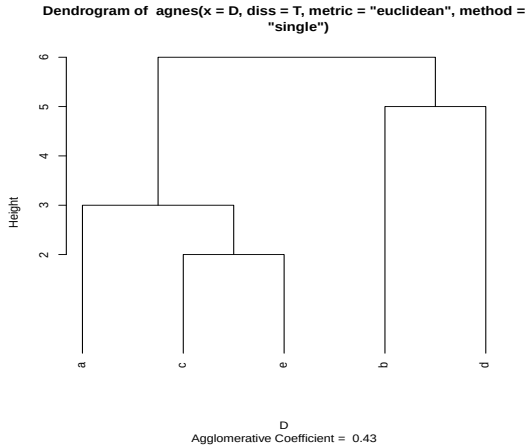
$$\mathbf{D} = \begin{bmatrix} \text{bb} & \text{ace} \\ 0 & \\ 6 & 0 \end{bmatrix}$$

En R se puede hacer el problema del siguiente modo (contempla los tres métodos de enlace que mencioné antes, más otros tres métodos más sofisticados):

Ejemplo conglomerados jerárquicos Aglomerativos IV

```
library(cluster)

X <- c("a", "b", "c", "d", "e")
D <- as.data.frame(matrix(c(0,9,3,6,11,9,0,7,5,10,3,7,0,9,2,6,5,9,0,8,11,10,2,8,0), nrow = 5),
                   row.names = X)
names(D) <- X
singlelink <- agnes(D, metric = "euclidean", method = "single", diss = T)
singlelink

Call:  agnes(x = D, diss = T, metric = "euclidean", method = "single")
Agglomerative coefficient:  0.4333333
Order of objects:
[1] a c e b d
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.00    2.75    4.00    4.00    5.25    6.00

Available components:
[1] "order"     "height"    "ac"        "merge"     "diss"      "call"      "method"    "order.lab"

plot(singlelink, which.plots = 2, hang = -1)
```

Ejemplo conglomerados jerárquicos Aglomerativos V



Ejemplo conglomerados jerárquicos Aglomerativos VI

El *coeficiente aglomerativo* o *divisivo* mide la estructura de conglomerados que se obtuvo. Mientras más cerca esté de 1, los datos estarán mejor agrupados. Se calcula con la siguiente fórmula:

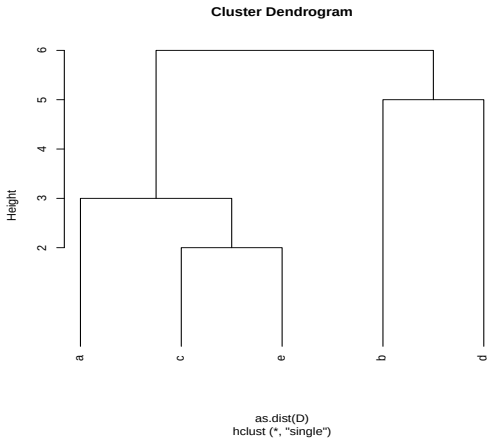
$$1 - \frac{\sum_{i=1}^{n} m(i)}{n}$$

donde $m(i)$ es la distancia al primer cluster con que se une la observación i, dividida por la distancia del cluster al que se une en el último paso del algoritmo.

Alternativamente, se puede usar la función `hclust`:

```
plot(hclust(d = as.dist(D), method = "single"), hang = -1)
```

Ejemplo conglomerados jerárquicos Aglomerativos VII



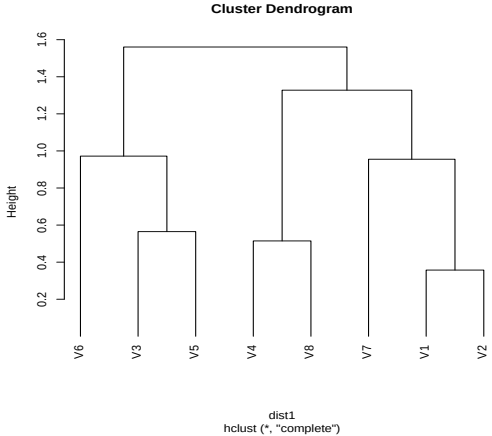
□

Ejemplo: Compañías de servicios I

En este ejercicio, se realiza la agrupación de variables, no de items. Noten que se tiene que transponer la matriz de datos

```
library(factoextra)
datos <- read.delim("https://raw.githubusercontent.com/jvega68/EA3/master/datos/J%26W/T12-4.DAT",sep = "",header = F)
dist1 <- get_dist(t(datos[,-9]),method = "pearson",diag = T) # Se obtienen las correlaciones como distancias entre las variables.
mod <- hclust(dist1, method = "complete")
plot(mod, hang = -1)
```

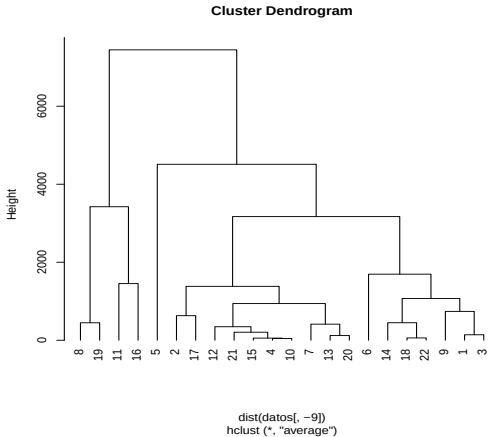
Ejemplo: Compañías de servicios II



Considerando ahora las compañías con distancia euclideana y utilizando el método de enlace promedio

Ejemplo: Compañías de servicios III

```
mod <- hclust(dist(datos[,-9]), method = "average")
plot(mod, hang = -1)
```



Método de aglomeración jerárquica de Ward (1963) I

- Minimiza "pérdida de información" al unir dos grupos (Suma de errores al cuadrado de todos los clusters, $ESS = \sum_{i=1}^{k} ESS_i$, donde $ESS_i = \sum_{j \in C_i} (\mathbf{x}_j - \bar{\mathbf{x}}_i)'(\mathbf{x}_j - \bar{\mathbf{x}}_i) = \sum_{j \in C_i} ||\mathbf{x}_j - \bar{\mathbf{x}}_i||^2$) se conoce como la *inercia*.
- Parte de que cada item es un cluster inicial, y en ese caso $ESS = 0$.
- En este método, se consideran los $\binom{K}{2}$ pares de clusters y se combinan los que dan un incremento mínimo a ESS
- En el paso final, cuando se juntan todos los N items en un cluster se obtiene:

$$ESS = \sum_{j=1}^{N} (\mathbf{x}_j - \bar{\mathbf{x}})'(\mathbf{x}_j - \bar{\mathbf{x}}) = ||\mathbf{x}_j - \bar{\mathbf{x}}||^2$$

- Los valores de ESS_i pueden graficarse y usarse para decidir cuántos clusters 'naturales' se pueden considerar. En el dendrograma se mide el eje vertical con ESS.
- Realmente Ward describe en su artículo un procedimiento muy general. Habla de funciones objetivo a minimizar, pero no especifica ninguna en particular. La documentación de R enfatiza lo siguiente:

Método de aglomeración jerárquica de Ward (1963) II

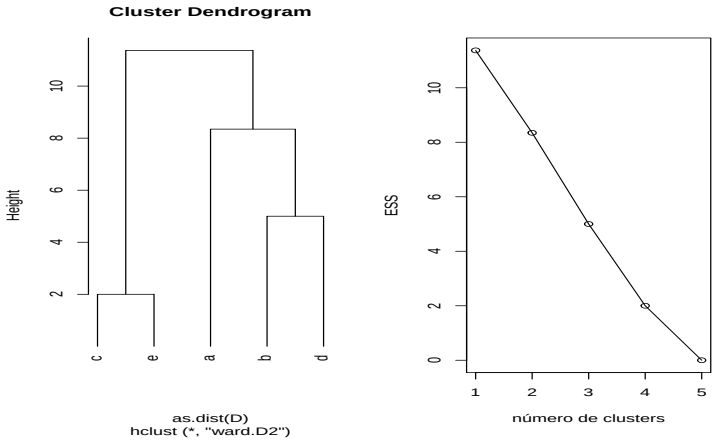
```
Two different algorithms are found in the literature for Ward clustering.  The one used by option ‘"ward.D"’
(equivalent to the only Ward option  ‘"ward"’ in R versions <= 3.0.3) does not implement Ward's (1963)
clustering criterion, whereas option  ‘"ward.D2"’ implements that criterion (Murtagh and Legendre 2014).
With the latter, the dissimilarities are squared before cluster updating.  Note that
‘agnes(*, method="ward")’ corresponds to ‘hclust(*, "ward.D2")’.
```

Ejemplo. [método de Ward]

Podemos realizar el ejercicio anterior con el método de Ward, para comparar resultados.

```
(m.ward <- hclust(d = as.dist(D), method = "ward.D2")) # ward.D2 es el método original de Ward (1963)


Call:
hclust(d = as.dist(D), method = "ward.D2")

Cluster method   : ward.D2
Number of objects: 5

par(mfrow = c(1,2))
plot(m.ward, hang = -1)
plot(5:1, c(0, m.ward$height), type = "o", ylab = "ESS", xlab = "número de clusters")
```

Método de aglomeración jerárquica de Ward (1963) III



□

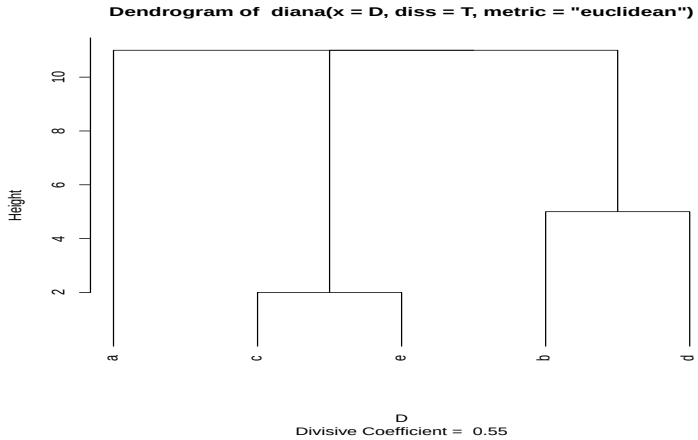
Ejemplo Método divisivo (Diana) I

Diana (DIvisive ANAlysis Clustering) utililza un coeficiente divisivo que mide la cantidad de estructura de cluster encontrada. No usa enlaces como en el caso del método aglomerativo. Ver detalles en capítulo 6 de Kaufman y Rousseeuw (1990).

Ejemplo Método divisivo (Diana) II

```
library(cluster)

X <- c("a", "b", "c", "d", "e")
D <- as.data.frame(matrix(c(0,9,3,6,11,9,0,7,5,10,3,7,0,9,2,6,5,9,0,8,11,10,2,8,0), nrow = 5),
row.names = X)
names(D) <- X
singlelink <- diana(D, metric = "euclidean", diss = T)
singlelink

Merge:
     [,1] [,2]
[1,]   -3   -5
[2,]   -2   -4
[3,]    1    2
[4,]   -1    3
Order of objects:
[1] a c e b d
Height:
[1] 11  2 11  5
Divisive coefficient:
[1] 0.5454545

Available components:
[1] "order"     "height"    "dc"        "merge"     "diss"      "call"      "order.lab"

plot(singlelink, which.plots = 2, hang = -1)
```

Ejemplo Método divisivo (Diana) III



Ejemplo de aplicación: Posicionamiento de marca en Marketing I

El siguiente ejemplo muestra cómo se puede aplicar el análisis de conglomerados sobre las *variables* y no directamente sobre las observaciones.

- Los siguientes datos provienen de una encuesta de percepción de marca. Los datos reflejan ratings de consumidores de marcas con *adjetivos perceptuales* como se expresa en las preguntas de la encuesta, como la siguiente:

  En una escala de 1 a 10, donde 1 es menos y 10 es mas, ¿qué tan [ADJETIVO] es [MARCA A]? Por ejemplo: ¿Qué tan amargo es Starbucks? o ¿Qué tan trendy es Benneton?

  Los datos que están en el archivo `brands8.csv` consisten en ratings de 10 marcas (‘a’ a ‘j’) sobre 9 adjetivos para $N = 100$ encuestados. Como hay 100 encuestados en 10 marcas, hay 1,000 renglones en los datos.

Ejemplo de aplicación: Posicionamiento de marca en Marketing II

```
ratings.marcas <- read.csv("https://raw.githubusercontent.com/jvega68/EA3/master/datos/brands8.csv")  # con file.choose(), R pregunta
head(ratings.marcas,3) # principio del archivo
```

```
  perform leader latest fun serious bargain value trendy rebuy brand
1       2      4      8   8       2       9     7      4     6     a
2       1      1      4   7       1       1     1      2     2     a
3       2      3      5   9       2       9     5      1     6     a
```

```
tail(ratings.marcas,3) # fin del archivo
```

```
     perform leader latest fun serious bargain value trendy rebuy brand
998        1      1     10  10       1       6     5      5     2     j
999        1      1      7   5       1       1     2      5     1     j
1000       7      4      7   8       4       1     2      5     1     j
```

```
summary(ratings.marcas)
```

```
    perform          leader          latest           fun            serious     
 Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
 1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 2.000  
 Median : 4.000   Median : 4.000   Median : 7.000   Median : 6.000   Median : 4.000  
 Mean   : 4.488   Mean   : 4.417   Mean   : 6.195   Mean   : 6.068   Mean   : 4.323  
 3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.: 9.000   3rd Qu.: 8.000   3rd Qu.: 6.000  
 Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
    bargain          value           trendy          rebuy           brand          
 Min.   : 1.000   Min.   : 1.000   Min.   : 1.00   Min.   : 1.000   Length:1000       
 1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 3.00   1st Qu.: 1.000   Class :character  
 Median : 4.000   Median : 4.000   Median : 5.00   Median : 3.000   Mode  :character  
 Mean   : 4.259   Mean   : 4.337   Mean   : 5.22   Mean   : 3.727                     
 3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 7.00   3rd Qu.: 5.000                     
 Max.   :10.000   Max.   :10.000   Max.   :10.00   Max.   :10.000                     
```

- Las etiquetas de cada columna muestran los adjtivos usados:

Ejemplo de aplicación: Posicionamiento de marca en Marketing III

- *perform*: la marca tiene un desempeño fuerte
- *leader*: la marca es lider en su segmento
- *latest*: la marca tiene los productos más recientes
- *fun*: la marca es divertida
- *serious*: la marca es seria
- *bargain*: los productos de esta marca son baratos
- *value*: los productos de la marca tienen un buen valor
- *trendy*: la marca impone moda
- *rebuy*: Yo compraría de nuevo la Marca.

- Para analizar los datos, primero escalamos los datos:

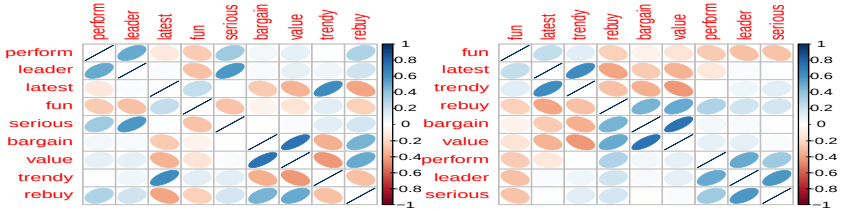
```
# Excluimos el nombre de la marca
ratings.sc <- ratings.marcas # hacemos una copia de los datos para no afectar la fuente
ratings.sc[,1:9] <- scale(ratings.sc[,1:9])  # escala todas las variables de una sola vez
```

Ejemplo de aplicación: Posicionamiento de marca en Marketing IV

- Interpretamos correlaciones: el parámetro `hclust` reordena as variables por la similitud entre variables basado en la correlación. ¿Cuántos clusters se perciben y quienes los componen?

```
library(corrplot)
par(mfrow=c(1,2))
corrplot(cor(ratings.sc[,1:9]), method = "ellipse")
corrplot(cor(ratings.sc[,1:9]), order = "hclust", method = "ellipse")
```

Ejemplo de aplicación: Posicionamiento de marca en Marketing V



Posicionamiento de marca (cont) I

- Agregamos los ratings medios por marca.
  La pregunta más fácil con estos datos es: ¿Cuál es la posición promedio de la marca en cada adjetivo?
  Lo podemos hacer de dos maneras:
  1. Herramientas básicas: `aggregate`

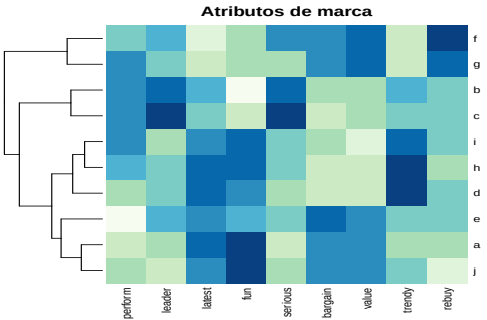
```
options(width=140)
# El punto en la fórmula es para indicar que queremos usar todas las variables que están en el data.frame
marca.media <- aggregate(. ~ brand, data = ratings.sc, FUN = mean)
# Housekeeping: Para tener un mejor arreglo de los datos, podemos quitar la columna de la marca y usarla como nombre para los renglones:
rownames(marca.media) <- marca.media$brand
marca.media$brand <- NULL
marca.media
```

```
      perform     leader     latest        fun     serious     bargain       value      trendy       rebuy
a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067  0.21409609  0.18469264 -0.52514473 -0.59616642
b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061  0.04161938  0.15133957  0.74030819  0.23697320
c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461 -0.60704302 -0.44067747  0.02552787 -0.13243776
d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505 -0.88075605 -0.93263529  0.73666135 -0.49398892
e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361  0.55155051  0.41816415  0.13857986  0.03654811
f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066  0.87400696  1.02268859 -0.81324496  1.35699580
g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906  0.89650392  1.25616009 -1.27639344  1.36092571
h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855 -0.73827529 -0.78254646  0.86430070 -0.60402622
i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746 -0.25459062 -0.80339213  0.59078782 -0.20317603
j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870 -0.09711188 -0.07379367 -0.48138267 -0.96164748
```

Posicionamiento de marca (cont) II

2. Podemos usar una gráfica de calor para mostrar los resultados en colores.

```
library(RColorBrewer) # paletas de colores
heatmap(as.matrix(marca.media), Colv = NA, col = brewer.pal(9, "GnBu"), main = "Atributos de marca")
```



Posicionamiento de marca (cont) III

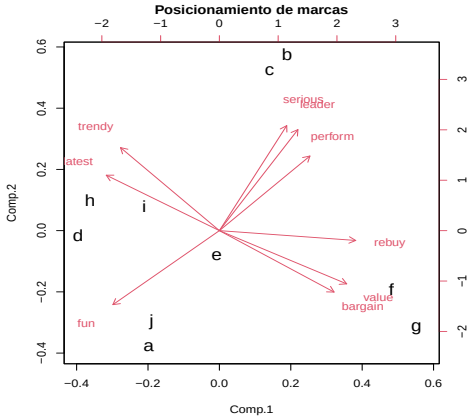
- Podemos intentar con los agregados hacer una gráfica de componentes principales para ver el posicionamiento de las marcas.

```
marca.media.pc <- princomp(marca.media,cor = T)
summary(marca.media.pc, loadings = T)
```

```
Importance of components:
                          Comp.1    Comp.2     Comp.3     Comp.4    Comp.5     Comp.6      Comp.7      Comp.8       Comp.9
Standard deviation     2.134521 1.7349473 0.76898915 0.61498280 0.5098261 0.36661576 0.215062433 0.145882355 0.0486674686
Proportion of Variance 0.506242 0.3344491 0.06570492 0.04202265 0.0288803 0.01493412 0.005139094 0.002364629 0.0002631692
Cumulative Proportion  0.506242 0.8406911 0.90639603 0.94841868 0.9772990 0.99223311 0.997372202 0.999736831 1.0000000000

Loadings:
        Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
perform  0.285  0.337  0.481  0.470  0.396  0.435
leader   0.247  0.457 -0.317 -0.191         0.119 -0.610         0.451
latest  -0.356  0.251 -0.496  0.275  0.461        -0.196 -0.119 -0.466
fun     -0.336 -0.335 -0.152  0.324 -0.388  0.636 -0.246  0.179
serious  0.212  0.475 -0.244 -0.212 -0.394  0.334  0.439        -0.407
bargain  0.361 -0.278 -0.459  0.291  0.112  0.127  0.319 -0.513  0.321
value    0.401 -0.241 -0.336         0.206                0.778
trendy  -0.311  0.375         0.484 -0.273 -0.339  0.322  0.243  0.410
rebuy    0.430                0.442 -0.438 -0.368 -0.352 -0.142 -0.372
```

```
biplot(marca.media.pc, main="Posicionamiento de marcas", cex = c(1.5,1))
```

Posicionamiento de marca (cont) IV



¿Qué conclusiones se pueden obtener de la gráfica? ¿Cómo podemos tomar algunas decisiones respecto a la marca **e**?

Métodos de conglomerados no jerárquicos

Características generales de los métodos no jerárquicos

- Están diseñados para agrupar a las observaciones, más que a las variables, en K grupos.
- El número de grupos K se puede determinar de antemano o como parte del procedimiento de estimación. Sin embargo, no se recomienda fijar de antemano el número de grupos:
  - Puede haber grupos no bien diferenciados cuando se eligen valores iniciales dentro de uno de los grupos 'naturales'.
  - La existencia de valores extremos puede afectar la configuración final, creando grupos artificiales.
  - Se pueden obtener grupos poco intuitivos.
- Sin embargo, en la práctica varios de los algoritmos piden de antemano el número de clusters.
- Se requiere definir una partición inicial de los items en grupos, o bien de un conjunto inicial de puntos semillas que formarán los núcleos de los clusters. La solución final dependerá de las condiciones iniciales.
- Otras características de estos métodos:
  - No se requiere especificar la matriz de similitudes o distancias.
  - Mucho más eficientes computacionalmente que los métodos jerárquicos, por lo que se puede trabajar con conjuntos de datos mucho más grandes.

Métodos a considerar

Los procedimientos que se considerarán son los siguientes:

- K-medias
- K-modas
- Partición alrededor de medioides (Pam)
- Análisis difuso (Fanny)
- Análisis monotético (Mona)

K-medias I

- Originalmente desarrollado por MacQueen en 1967. Requiere los datos, no la matriz de distancias o similaridades.
- El algoritmo general se basa en asignar cada item de acuerdo a:
  - intra-clusters: máxima similaridad
  - inter-clusters: máxima separación

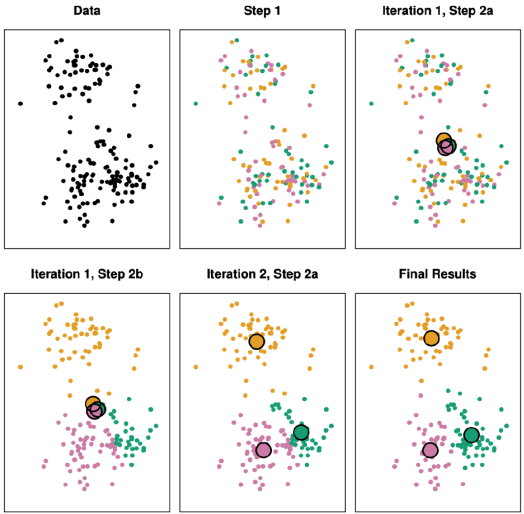
Algoritmo de k-medias

1. Se seleccionan K puntos como centros de los grupos iniciales, de alguna de estas formas:
   A. Asigna aleatoriamente los items a los grupos y se toman los centroides de los grupos formados,
   B. Selecciona K puntos al azar como los centroides.
2. Calcular las distancias euclídeas de cada item a los centroides de los K grupos, y se asigna ese item al más próximo, de manera secuencial. Con cada asignación, se recalcula el centro del grupo al que el item fue asignado y del grupo que perdió el item.
3. A partir de algún criterio de optimalidad, se verifica si con la reasignación de los items mejora el criterio. Si no es posible mejorar el criterio, terminar el proceso.

K-medias II

- Hay varias versiones de este algoritmo que se encuentran programados en la función `kmeans`: Hartigan & Wong (1979), MacQueen (1967), Lloyd (1957, 1982), (basado en regiones de Voronoi), y Forgy (1965).
  Video para visualizar el algoritmo (hay muchos)
- Hay que tomar en cuenta que el algoritmo puede dar diferentes soluciones dependiendo de las condiciones iniciales.

K-medias en acción



Ejemplo 1 I

Consideraremos los datos `iris` para comparar con los métodos de clasificación que vimos antes.

```
data("iris")
iris2 <- iris
iris2$Species <- NULL
(mod1 <- kmeans(iris2,3))

K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3
 [68] 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 1 1 3 3 1 1 1 1 3 1 1 1 1 3 3
[135] 1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

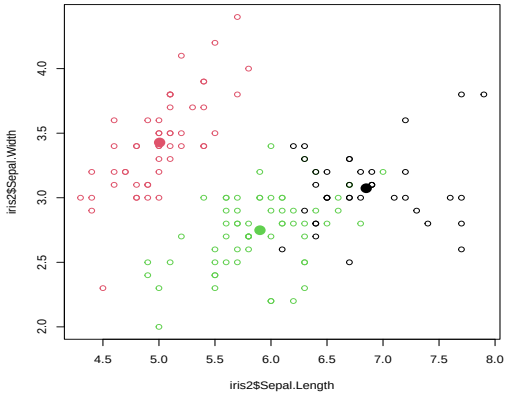
Ejemplo 1 II

Podemos comparar el resultado de clasificación con los datos originales

```
plot(iris2$Sepal.Length,iris2$Sepal.Width, col=mod1$cluster)
points(mod1$centers[,c("Sepal.Length","Sepal.Width")], col=1:3, pch=16,cex=2)
```

```
table(iris$Species,mod1$cluster)
```

```
            1  2  3
  setosa    0 50  0
  versicolor  2  0 48
  virginica  36  0 14
```



Ejemplo 2 I

Podemos aplicar el algoritmo a los encuestados del ejemplo de marketing, aunque no está muy claro si reproduciremos las marcas que se evaluaron. Esto más bien nos daría qué clientes son los que evalúan aproximadamente igual los mismos conceptos: por ejemplo aquellos que son muy exigentes, o los que son indiferentes, etc.

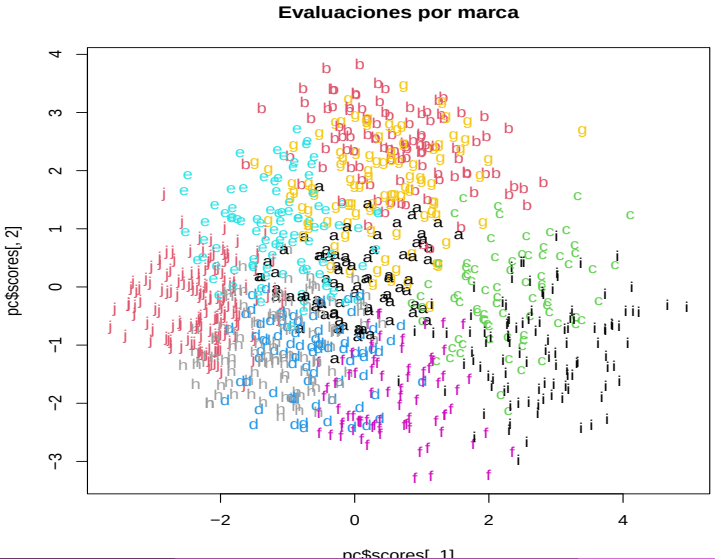
```
mod2 <- kmeans(ratings.sc[,1:9],10)
```

Para poder visualizar los resultados, podemos hacer un ejercicio de reducción de dimensión con componentes principales y ver los grupos en dos dimensiones

Ejemplo 2 II

```
pc <- princomp(ratings.sc[,1:9], cor = T)
summary(pc, loadings = T)

Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4     Comp.5     Comp.6     Comp.7     Comp.8     Comp.9
Standard deviation     1.7260636 1.4479474 1.0388719 0.85276667 0.79846472 0.73132977 0.62458341 0.55861115 0.49309927
Proportion of Variance 0.3310328 0.2329502 0.1199172 0.08080122 0.07083844 0.05942703 0.04334494 0.03467182 0.02701632
Cumulative Proportion  0.3310328 0.5639830 0.6839002 0.76470146 0.83553989 0.89496692 0.93831185 0.97298368 1.00000000

Loadings:
        Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
perform  0.237  0.420         0.526  0.468  0.337  0.364  0.144
leader   0.206  0.524               -0.295  0.297 -0.614 -0.288 -0.179
latest  -0.370  0.201 -0.533 -0.214  0.106  0.174 -0.185  0.643
fun     -0.251 -0.250 -0.418  0.751 -0.331 -0.141
serious  0.160  0.510               -0.555 -0.392  0.445  0.184
bargain  0.399 -0.218 -0.490 -0.167         0.139  0.288        -0.647
value    0.447 -0.190 -0.369 -0.151         0.220        -0.148  0.728
trendy  -0.351  0.318 -0.371 -0.168  0.366 -0.266  0.154 -0.615
rebuy    0.439        -0.125  0.130  0.356 -0.675 -0.389  0.202


plot(pc$scores[,1], pc$scores[,2],
     col = as.factor(mod2$cluster),
     pch = letters[1:10][as.factor(mod2$cluster)],
     main = "Evaluaciones por marca")
```

Ejemplo 2 III



K-Modas I

- El algoritmo de k-modas es una extensión de k-medias para agrupar grandes conjuntos de datos con variables categóricas.
- Basado en el paper de Huang (1998).
- El algoritmo utiliza una medida de disimilaridad simple para trabajar con datos categóricos, reemplaza los centroides de los grupos con modas y usa un método basado en frecuencias para actualizar las modas de los grupos en el proceso de reducir la función de costo objetivo.
- Una extensión adicional, el algoritmo de k-prototipos, combina datos categóricos y numéricos para encontrar las agrupaciones.[1]
- Tiene la ventaja de poder trabajar con datos faltantes.

[1] Aunque no lo veremos, en `R` se encuentra en el paquete `clustMixType`, bajo la función `kproto`.

Ejemplo k-modas

```r
library(klaR) # k-modas para datos categóricos
personas <- data.frame(cabello = c("rubio", "castaño", "rojo", "negro", "castaño", "negro", "rojo", "negro"),
                       ojos = c("ambar","gris","verdes","avellana","ambar","gris","verdes", "avellana"),
                       piel = c("clara", "oscura", "oscura", "oscura", "clara","oscura","clara","clara"))
set.seed(123) # se genera semilla aleatoria, pues el algoritmo tiene una asignación inicial aleatoria.
kmodes(personas, modes = 3)

K-modes clustering with 3 clusters of sizes 3, 2, 3

Cluster modes:
  cabello     ojos   piel
1 castaño    ambar  clara
2   negro avellana  clara
3 castaño     gris oscura

Clustering vector:
[1] 1 3 3 2 1 3 1 2

Within cluster simple-matching distance by cluster:
[1] 3 1 3

Available components:
[1] "cluster"    "size"       "modes"      "withindiff" "iterations" "weighted"
```

Partición alrededor de medioides (Pam) I

- Similar a k-medias, no busca centroides, sino *medioides*, que son k items representativos de los clusters, y usa disimilaridad en lugar de distancias euclídeas.
- El medioide de un cluster se define como el item en el cluster, cuya suma de disimilaridades de todos los objetos en el cluster es mínima.
- Es más robusto que k-medias.
- Puede usarse con la matriz de disimilaridades y una configuración inicial de k grupos, pero utiliza el mismo algoritmo que k-medias.
- Además, considera un procedimiento de *intercambio* de medioides dentro del cluster, siempre y cuando ese cambio reduzca la función objetivo. Es decir, para cada item, se intercambia con el medioide y se hace un swap cuando se reduce la función objetivo.
- No es eficiente para muestras muy grandes.

Ejemplo Pam

```
df <- scale(USArrests) # Escala los datos
head(df, n = 3)

            Murder   Assault   UrbanPop         Rape
Alabama 1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska  0.50786248 1.1068225 -1.2117642  2.484202941
Arizona 0.07163341 1.4788032  0.9989801  1.042878388

pam.res <- pam(df, k = 2)
print(pam.res)

Medoids:
           ID     Murder    Assault   UrbanPop       Rape
New Mexico 31  0.8292944  1.3708088  0.3081225  1.1603196
Nebraska   27 -0.8008247 -0.8250772 -0.2445636 -0.5052109
Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California       Colorado    Connecticut       Delaware        Florida 
             1              1              1              2              1              1              2              2              1 
       Georgia         Hawaii          Idaho       Illinois        Indiana           Iowa         Kansas       Kentucky      Louisiana 
             1              2              2              1              2              2              2              2              1 
         Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi       Missouri        Montana       Nebraska 
             2              1              2              1              2              1              1              2              2 
        Nevada  New Hampshire     New Jersey     New Mexico       New York North Carolina   North Dakota           Ohio       Oklahoma 
             1              2              2              1              1              1              2              2              2 
        Oregon   Pennsylvania   Rhode Island South Carolina   South Dakota      Tennessee          Texas           Utah        Vermont 
             2              2              2              1              2              1              1              2              2 
      Virginia     Washington  West Virginia      Wisconsin        Wyoming 
             2              2              2              2              2 
Objective function:
   build     swap 
1.441358 1.368969

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"  "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Ejemplo Pam

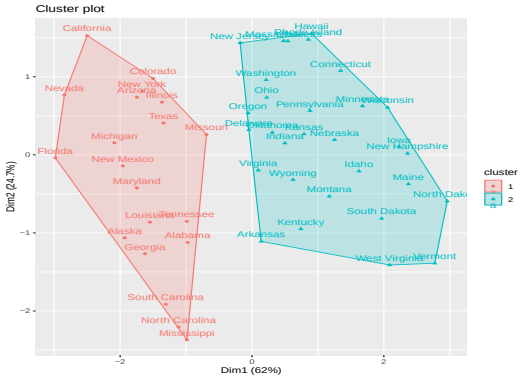
Adicionalmente, la función `pamk` del paquete `fpc` calcula Pam sin necesidad de especificar el número de clusters, lo hace como parte del problema.

```
library(fpc)
(mk <- pamk(scale(USArrests)))

$pamobject
Medoids:
           ID     Murder    Assault   UrbanPop       Rape
New Mexico 31  0.8292944  1.3708088  0.3081225  1.1603196
Nebraska   27 -0.8008247 -0.8250772 -0.2445636 -0.5052109
Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California       Colorado    Connecticut       Delaware        Florida 
             1              1              1              2              1              1              2              2              1 
       Georgia         Hawaii          Idaho       Illinois        Indiana           Iowa         Kansas       Kentucky      Louisiana 
             1              2              2              1              2              2              2              2              1 
         Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi       Missouri        Montana       Nebraska 
             2              1              2              1              2              1              1              2              2 
        Nevada  New Hampshire     New Jersey     New Mexico       New York North Carolina   North Dakota           Ohio       Oklahoma 
             1              2              2              1              1              1              2              2              2 
        Oregon   Pennsylvania   Rhode Island South Carolina   South Dakota      Tennessee          Texas           Utah        Vermont 
             2              2              2              1              2              1              1              2              2 
      Virginia     Washington  West Virginia      Wisconsin        Wyoming 
             2              2              2              2              2 
Objective function:
   build     swap 
1.441358 1.368969 

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"  "clusinfo"   "silinfo"    "diss"       "call"       "data"      

$nc
[1] 2
```

Ejemplo Pam

Para visualizar los clusters

```
library(factoextra)
fviz_cluster(pam.res)
```



Análisis difuso (Fanny) I

- En fuzzy clustering, se asignan probabilidades de pertenencia para cada uno de los k grupos.
- Se define $u_{ik} =$ fuerza de pertenencia del item i en el grupo k. Además, se requiere que para cada item i, $\{u_{ik}\}$, $k = 1, \dots, K$ se comporten como probabilidades:
  1. $u_{ik} \geq 0$ para todo i, k
  2. $\sum_{k=1}^{K} u_{ik} = 1$ para cada i.
- La función objetivo a minimizar es una especie de dispersión:

$$\sum_{k=1}^{K} \frac{\sum_i \sum_j u_{ik}^2 u_{jk}^2 d_{ij}}{2 \sum_l u_{lk}^2}$$

  La ecuación anterior se obtiene de hacer una ponderación de las disimilaridades por los pesos u_{ik}.
- La solución se obtiene usando las restricciones y un algoritmo iterativo. Esto impone la restricción adicional de que el número de grupos no puede ser mayor que $n/2$.

Nivel de dureza-difusión de los grupos I

- Algunas agrupaciones son más difusas y otras son muy duras:
  - Si cada item pertenece a todos los clusters ($u_{ik} = 1/k$) se tiene máxima difusión.
  - Si cada item pertenece a un sólo cluster ($u_{ik} = I(X_i \in C_k)$, es uno o cero), se tiene más dureza.
- El coeficiente de Dunn mide qué tan dura es la difusión:

$$F_k = \sum_{i=1}^{n} \sum_{\nu=1}^{k} \frac{u_{i\nu}^2}{n} \in (1/k, 1)$$

- La versión normalizada:

$$F_k' = \frac{F_k - (1/k)}{1 - (1/k)} = \frac{kF_k - 1}{k - 1} \in (0, 1)$$

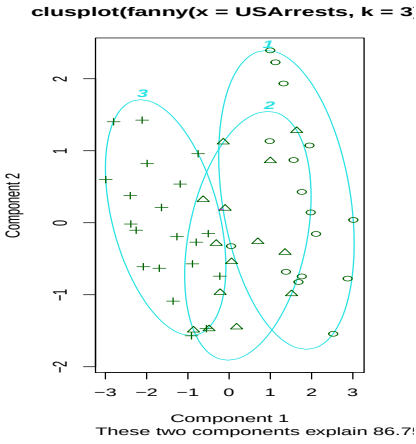
- Mientras más cercano a uno sea el índice de Dunn, más dura es la agrupación.

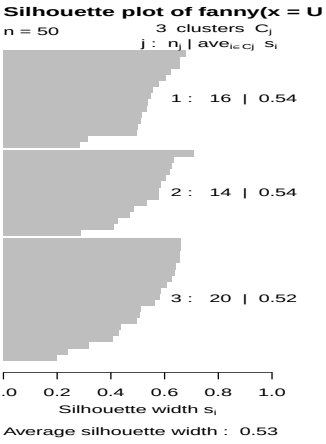
Ejemplo

```
modelo <- fanny(USArrests, k = 3)
    modelo$coeff  # Coeficiente de Dunn

dunn_coeff normalized
 0.5509585  0.3264378

    head(modelo$membership,10)  # pertenencia

                [,1]      [,2]       [,3]
Alabama     0.65708031 0.2381836 0.10473610
Alaska      0.74250646 0.1672458 0.09024773
Arizona     0.77552510 0.1411409 0.08333405
Arkansas    0.24041554 0.5929564 0.16662802
California  0.76825279 0.1486784 0.08306880
Colorado    0.35340691 0.4906862 0.15590690
Connecticut 0.08854613 0.3272313 0.58422258
Delaware    0.67219053 0.2274574 0.10035208
Florida     0.63642563 0.2189294 0.14464501
Georgia     0.41149634 0.4389330 0.14957071

    head(modelo$clustering, 10)

    Alabama      Alaska     Arizona    Arkansas  California    Colorado Connecticut    Delaware     Florida     Georgia
          1           1           1           2           1           2           3           1           1           2
```

Ejemplo (cont.)

```
par(mfrow=c(1,2))
plot(modelo, which = 1, labels = 5)
plot(modelo, which = 2)
```





Análisis monotético (Mona) I

- El análisis monotético se refiere a que usa sólo una variable a la vez para hacer las particiones.
- El análisis `MONA` es especialmente útil para vectores binarios, correspondientes a indicadoras de atributos.
- El procedimiento para este caso es el siguiente:
  1. Imputar la matriz si hay datos faltantes (en `R` se permiten un limitado número de valores faltantes, ver restricciones).
  2. Dividir cada cluster de acuerdo a una variable bien elegida X_j: un cluster es para todos los casos que tienen $X_{lJ} = 1$ y el otro cluster es para los que tienen $X_{mJ} = 0$. La variable usada para dividir un cluster es la variable con la mayor asociación a las otras variables.

     La asociación entre variables f y g está dada por $A_{fg} = |\det(Cont_{fg})|$ donde $Cont_{fg} = \begin{pmatrix} a_{fg} & b_{fg} \\ c_{fg} & d_{fg} \end{pmatrix}$ es la tabla de contingencia de las variables f y g y a_{fg} es la frecuencia de unos en ambas variables condicional al cluster que se va a dividir. La asociación total de una variable f es $A_f = \sum_{g \neq f} A_{fg}$. La variable h que satisface $A_h = \text{máx}_f A_f$ se selecciona para dividir el cluster.
  3. Repetir el paso 2 hasta que cada cluster consista de objetos que tengan idénticos valores para todas las variables

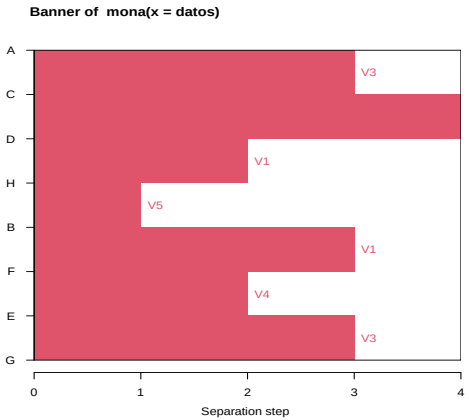
Ejemplo 1 I

```
datos <- data.frame(
        V1 = c(1,1,1,1,0,0,0,0),
        V2 = c(1,1,1,1,0,0,0,0),
        V3 = c(0,0,1,1,0,0,1,1),
        V4 = c(1,0,1,1,1,0,1,1),
        V5 = c(1,0,1,1,0,0,0,1),
        V6 = c(0,1,0,0,1,0,1,0))
row.names(datos) <- c("A","B","C","D","E","F","G","H")
datos
```

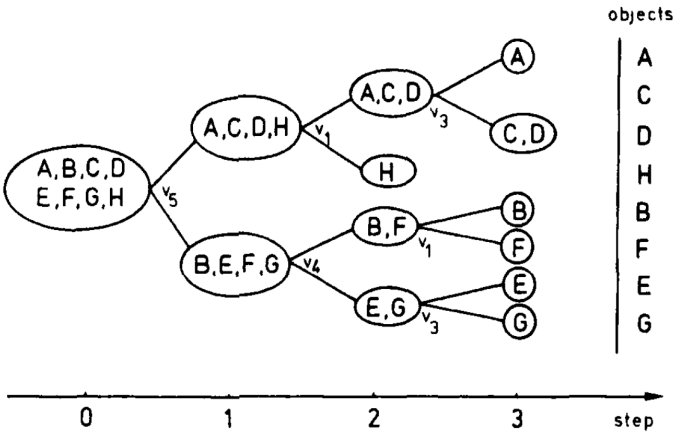
```
  V1 V2 V3 V4 V5 V6
A  1  1  0  1  1  0
B  1  1  0  0  0  1
C  1  1  1  1  1  0
D  1  1  1  1  1  0
E  0  0  0  1  0  1
F  0  0  0  0  0  0
G  0  0  1  1  0  1
H  0  0  1  1  1  0
```

```
m1 <- mona(datos)
plot(m1)
```

Ejemplo 1 II


Banner of mona(x = datos)
A
C
D
H
B
F
E
G
V3
V1
V5
V1
V4
V3
0
1
2
3
4
Separation step

Ejemplo 1


objects
A
C
D
H
B
F
E
G
A,B,C,D
E,F,G,H
A,C,D,H
A,C,D
B,E,F,G
B,F
E,G
C,D
v5
v1
v3
v4
0
1
2
3
step

Ejemplo 2 I

Por ejemplo, los datos en `animals`:

```
data(animals)
dim(animals)

[1] 20  6

head(animals)

    war fly ver end gro hai
ant   1   1   1   1   2   1
bee   1   2   1   1   2   2
cat   2   1   2   1   1   2
cpl   1   1   1   1   1   2
chi   2   1   2   2   2   2
cow   2   1   2   1   2   2
```

En este conjunto de datos cada renglón corresponde a un animal y las columnas son los siguientes atributos:

- `war` si es de sangre fría o caliente
- `fly` si vuela o no vuela
- `ver` si es vertebrado o invertebrado.
- `end` si está en peligro o no

Ejemplo 2 II

- `gro` si vive en grupos o no
- `hai` si tiene cabello o no.

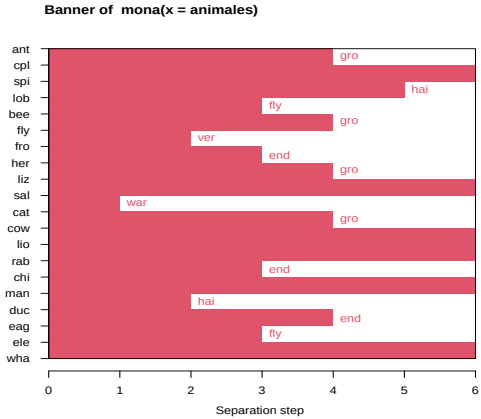
```
animales <- animals
animales <- animales -1
animales[is.na(animales)] <- 0 #imputación (se puede mejorar)
head(animales,3)

    war fly ver end gro hai
ant   0   0   0   0   1   0
bee   0   1   0   0   1   1
cat   1   0   1   0   0   1

(mona1 <- mona(animales))

mona(x, ..) fit;  x of dimension 20x6
Order of objects:
 [1] ant cpl spi lob bee fly fro her liz sal cat cow lio rab chi man duc eag ele wha
Variable used:
 [1] gro  NULL hai  fly  gro  ver  end  gro  NULL war  gro  NULL NULL end  NULL hai  end  fly  NULL
Separation step:
 [1] 4 0 5 3 4 2 3 4 0 1 4 0 0 3 0 2 4 3 0

Available components:
[1] "data"      "hasNA"     "order"     "variable"  "step"      "order.lab" "call"
```

Gráfica de Banner I

- Una gráfica de Banner muestra las divisiones sucesivas de izquierda a derecha. La longitud de una barra es el número de pasos divisivos necesarios para hacer una división. Junto a cada barra, se muestra cuál es la variable que causa la división. Una barra que continúa en el margen derecho, indica un cluster que no se puede dividir.

```
plot(mona1)
```

Gráfica de Banner II



Determinación óptima del número de clusters

Determinación óptima del número de clusters I

- Como se ha mencionado, sería ideal no determinar de antemano el número de clusters. Se sugiere hacer una evaluación adecuada para determinar cuál debería ser el número óptimo de clusters a considerar. Hay tres métodos que son comunes:
  - Método del codo
  - Método de la silueta
  - estadística gap
- Revisaremos aquí sólo los dos primeros casos. El tercero se puede consultar en el artículo de Tibishirani, Walther, Hastie (2001).

Método de codo I

Método de codo

1. Se evalúa el algoritmo de conglomerados (e.g. k-medias) para diferentes valores de k.
2. Para cada k se calcula $ESS(k)$[a].
3. Graficar $ESS(k)$ vs k
4. Si se forma un codo, es donde se especifica el número de clusters óptimo.

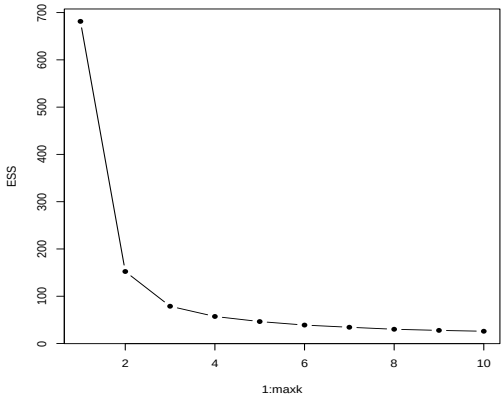
[a] Notar que $ESS(k) = ESS$ no se refiere a los componentes de ESS, que denotamos con ESS_k.

Ejemplo. [Cálculo para `iris`]

A continuación evaluamos el procedimiento indicado

```
maxk <- 10 #número de clusters a considerar
ESS <- sapply(1:maxk,function(k)kmeans(iris2,k,nstart=10)$tot.withinss)
plot(1:maxk,ESS, type="b",pch=16)
```

Método de codo II



□

Método de la silueta I

- A veces el método del codo es un poco ambiguo en la determinación del número óptimo de clusters. Por eso es conveniente tener más de una opción.
- El *análisis de silueta* mide qué tan bien una observación es conglomerada y estima la distancia promedio entre conglomerados. La gráfica de la silueta muestra una medida de que tan cerca cada punto en un cluster está de puntos en los clusters vecinos.

Algoritmo para silueta

Para cada observación i, el ancho de la silueta s_i se calcula de la siguiente manera:

1. Se calcula la disimilaridad promedio a_i entre i y todos los puntos del cluster al que pertenece el item i.
2. Para todos los otros clusters C a los que no pertenece i, se calcula la disimilaridad promedio $d(i, C)$ de i a todos los items de C. Se define $b_i = \text{mín}_C\, d(i, C)$, y se interpreta como la disimilaridad entre i y el cluster más cercano al que no pertenece i.
3. El ancho de la silueta del item i se define como $s_i = \frac{b_i - a_i}{\text{máx}\{a_i, b_i\}}$

Podemos interpretar el ancho de la silueta de la siguiente manera:

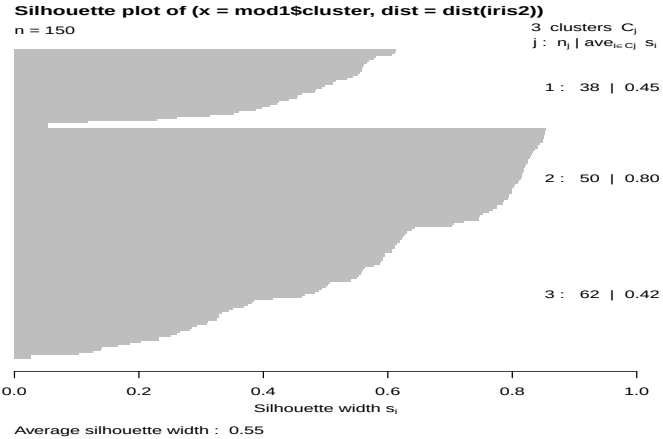
Método de la silueta II

- Los items con un valor grande de s_i (casi 1) están bien agrupados
- Los items con un valor pequeño de s_i (casi 0), significa que los items quedan entre dos clusters.
- Items con valores negativos de s_i están mal agrupados.
- En el paquete `cluster` se puede usar la función `silhouette`.
- En el paquete `factoextra` la función `fviz_nbclust` genera las gráficas de codo y de silueta.

Ejemplo. [silueta para `iris`]

```
plot(silhouette(mod1$cluster,dist(iris2)))
```

Método de la silueta III



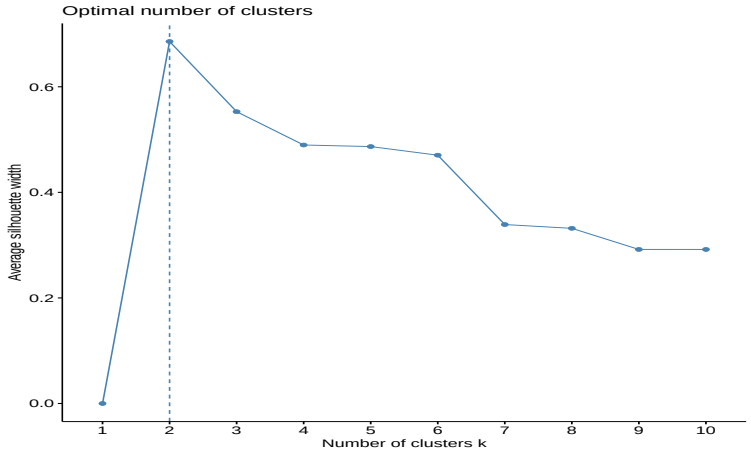
Otra alternativa utilizando `factoextra`. Se calcula el agrupamiento para diferente número k. Se calcula la silueta promedio de los clusters de acuerdo al número considerado k. La silueta promedio mide la calidad del agrupamiento,

Método de la silueta IV

con un alto valor de la silueta promedio indicando una buena agrupación. La k óptima es la que maximiza la silueta promedio sobre los valores de k.

```
fviz_nbclust(iris2, pam, method = "silhouette")
```

Método de la silueta V



□