

Aprendizaje No Supervisado: Conglomerados (Clustering)

1EST17-Aprendizaje Estadístico 1

Mg. Enver Gerald Tarazona Vargas
enver.tarazona@pucp.edu.pe

Maestría en Estadística

Escuela de Posgrado



Resumen I

1

Introducción

- Definiciones Generales
- Clasificación
- Medidas de similaridad y distancias

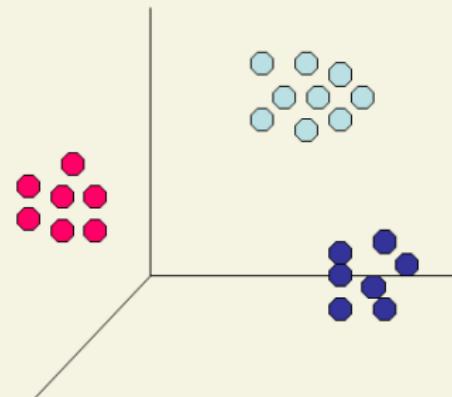
Grandes dimensiones de datos

Dada una nube de observaciones queremos entender su estructura.



¿Qué es el análisis cluster?

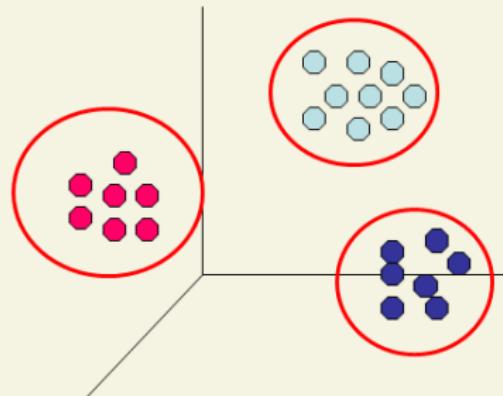
Encontrar grupos de objetos de tal forma que los objetos en un grupo sean similares (o relacionados) unos a otros y distintos de (o poco relacionados) otros objetos en otros grupos



Los clusters son descubiertos durante el análisis.

¿Qué es el análisis cluster?

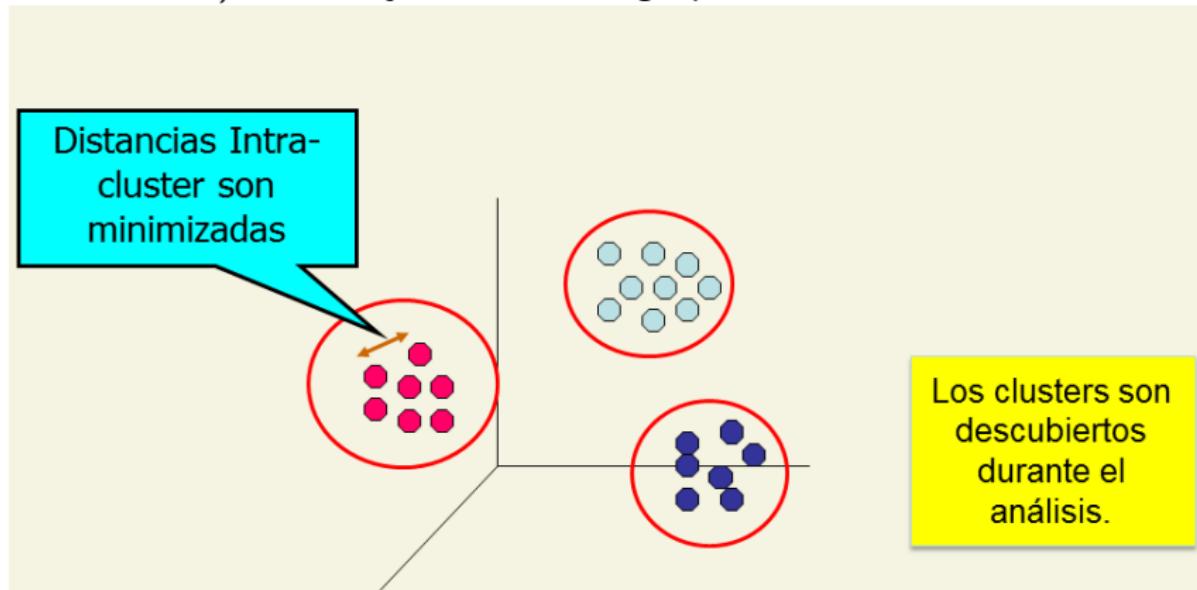
Encontrar grupos de objetos de tal forma que los objetos en un grupo sean similares (o relacionados) unos a otros y distintos de (o poco relacionados) otros objetos en otros grupos



Los clusters son descubiertos durante el análisis.

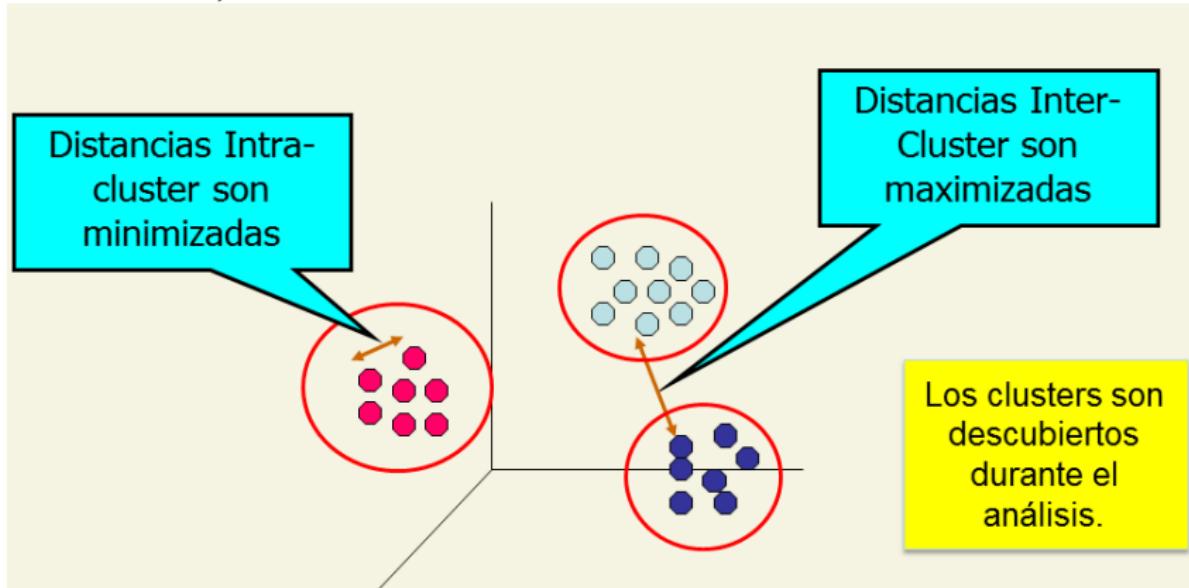
¿Qué es el análisis cluster?

Encontrar grupos de objetos de tal forma que los objetos en un grupo sean similares (o relacionados) unos a otros y distintos de (o poco relacionados) otros objetos en otros grupos



¿Qué es el análisis cluster?

Encontrar grupos de objetos de tal forma que los objetos en un grupo sean similares (o relacionados) unos a otros y distintos de (o poco relacionados) otros objetos en otros grupos



Introducción I

- El clustering está referido a un amplio conjunto de técnicas para encontrar subgrupos o *clusters* (conglomerados) de observaciones (y variables) en una base de datos.
- Es uno de los métodos descriptivos más conocidos y usados en Minería de Datos.
- Tiene como objetivo principal encontrar subconjuntos homogéneos.
- Cuando se agrupan las observaciones de una base de datos, se busca particionar las observaciones en grupos de tal forma que las observaciones dentro de cada grupo sean muy similares unas de otras, mientras que las observaciones de grupos distintos deben ser muy diferentes.
- Los términos similares y diferentes deben ser definidos. Usualmente esta es una consideración específica a un dominio que debe basarse en el conocimiento de los datos a ser estudiados.

Introducción II

- No es lo mismo que clasificación: No hay una variable dependiente.
- El clustering es una técnica no supervisada debido a que estamos tratando de descubrir una estructura sobre la base del conjunto de datos.
- Por lo general no es adecuado comparar algoritmos de cluster distintos.
- Cluster de documentos = Text Mining y Web Mining

Aplicaciones del Análisis Cluster

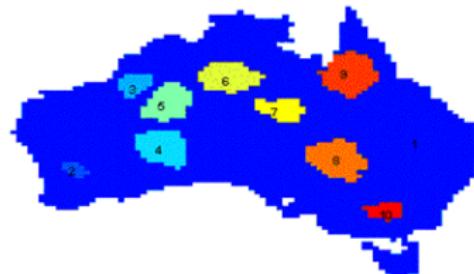
Conocimiento

- Grupos de documentos relacionados para búsquedas, grupos de genes y proteínas que tienen similar funcionalidad, grupos de acciones con similares fluctuaciones en sus valores

Reducción

- Reducir el tamaño de un conjunto de datos.

	<i>Discovered Clusters</i>	<i>Industry Group</i>
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN,Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN,DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN,Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down,Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN,Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN,ADV-Micro-Device-DOWN,Andrew-Corp-DOWN,Computer-Assoc-DOWN,Circuit-City-DOWN,Compaq-DOWN,EMC-Corp-DOWN,Gen-Inst-DOWN,Motorola-DOWN,Microssoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN,MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP,Louisiana-Land-UP,Phillips-Petro-UP,Uncal-UP,Schlumberger-UP	Oil-UP



Aplicaciones del Análisis Cluster

Marketing

- Perfiles de clientes.

Ventas

- Agrupar almacenes en grupos de establecimientos que sean homogéneos en cuanto a tipo de cliente, facturación, tamaño de tienda.

Sociología

- Dividir a la población en grupos de individuos con estilos de vida y expectativas similares.



¿Qué no es análisis cluster?

- Clasificación supervisada
 - Tener información de la etiqueta de la clase
- Segmentación simple
 - Divido a los estudiantes en diferentes grupos de registros por orden alfabético de los apellidos.
- Resultado de una consulta
 - Agrupamientos son los resultados de una especificación externa.
- Partición gráfica.

La noción de un cluster es muy ambigua



¿Cuántos clusters?

La noción de un cluster es muy ambigua

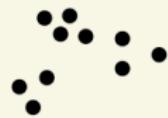


¿Cuántos clusters?

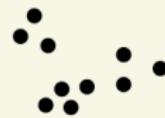


Dos clusters

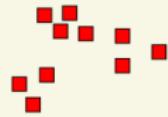
La noción de un cluster es muy ambigua



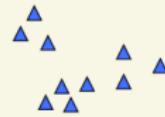
¿Cuántos clusters?



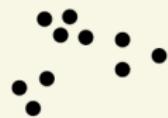
Seis clusters



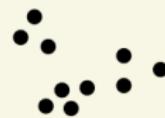
Dos clusters



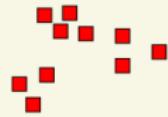
La noción de un cluster es muy ambigua



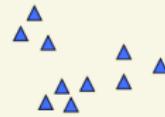
¿Cuántos clusters?



Seis clusters



Dos clusters



Cuatro Clusters



Clustering es una tarea difícil



¿Por qué es una tarea difícil?

- Clustering en dos dimensiones parece fácil.
- Clustering con una pequeña cantidad de datos parece fácil.
- En la mayoría de los casos las apariencias engañan.
- Muchas aplicaciones no envuelven 2, sino 10 o incluso 10000 dimensiones.
- Los espacios de grandes dimensiones luce distinto: La mayoría de pares de observaciones se encuentran a una distancia similar.

Tipos de Análisis Cluster

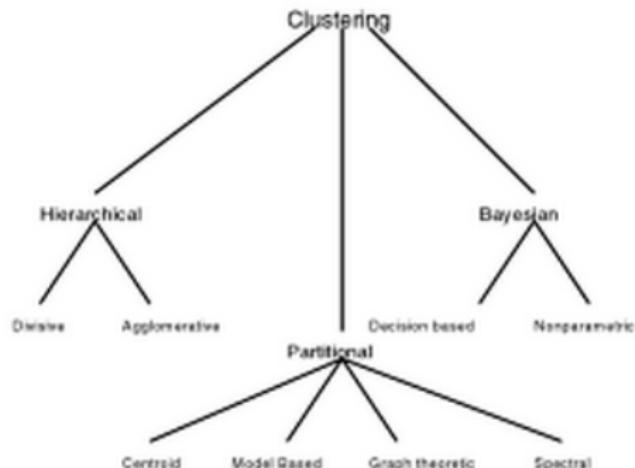


Figura: La colección de técnicas de clustering pueden ser agrupadas dentro de tres grupos -jerárquicas, particionales, y Bayesianas - y además subdivididas. Tomar en cuenta que las hojas no son exclusivas. Por ejemplo, hay técnicas que son de manera primaria espectrales pero arrojan un cluster jerárquico.

Clusters Jerárquicos I

- La característica de una técnica jerárquica es que genera una secuencia anidada de conglomerados.
- Usualmente uno debe de elegir un umbral que indique hasta donde debe de proseguirse en el procedimiento para encontrar la mejor secuencia de agrupamiento.
- Los anidamientos pueden ser crecientes o decrecientes.
- Si son decrecientes, usualmente se inicia considerando cada punto como un cluster. Estos procedimientos son considerados como aglomerativos.
- Si son crecientes, usualmente inician considerando a todo el conjunto de datos como un gran cluster que se descompone en otros más pequeños. Estos procedimientos son conocidos como divisivos.

Clusters Jerárquicos II

- La técnica llamada árboles de expansión mínima (minimal spanning trees) produce una secuencia anidada de las agrupaciones y puede ser considerado como jerárquica, a pesar de que algunos autores lo clasifican como una técnica gráfico-teórica.
- Los cluster jerárquicos generalmente requieren una medida de disimilitud,

Clusters Particionales I

- Usualmente requieren que se defina un número K de clusters y un agrupamiento inicial como entrada del procedimiento, el cuál tratará de mejorar la asignación inicial de estos datos.
- Algorítmicamente, la diferencia entre métodos jerárquicos y particionales es que en los algoritmos jerárquicos los grupos son encontrados usando cluster previamente establecidos, mientras que en métodos particionales se trata de determinar todos los clusters de manera óptima de un sólo tiro.

Clusters Bayesianos I

- Las técnicas de cluster bayesianas difieren de las anteriores porque tratan de generar una distribución a posteriori sobre la colección de todas las particiones de datos.
- En algunos casos puede ser necesario especificar el valor de K .
- La moda de la distribución a posterior es el cluster óptimo.
- Las técnicas bayesianas son más cercanas a las jerárquicas que a las particionales porque usualmente hay un orden en las particiones.
- Cualquiera de estas técnicas requieren la especificación de una a priori y usualmente se llega a cierta interpretación a través de una prueba de hipótesis.

Problemas comunes I

- Dos problemas que se deben enfrentar en la mayoría de las técnicas son la aglutinación y la disección.
- La aglutinación ocurre cuando un objeto se ajusta a uno o más clusters, que por lo tanto estarían superponiéndose.
- Un ejemplo es la recuperación de documentos: Una misma palabra puede tener dos diferentes significados, por lo que un texto puede no encajar fácilmente en un cluster.
- La disección ocurre cuando existe una sola población la cual no contiene clusters significativos, pero el objetivo es todavía agrupar a los datos para algún otro propósito.
- Por ejemplo, puede no haber una manera significativa de encontrar agrupamientos en una ciudad homogénea, pero la oficina postal puede querer dividir la ciudad en regiones administrativas para que el envío de correspondencia sea más eficiente.

Similaridad, distancia y proximidad

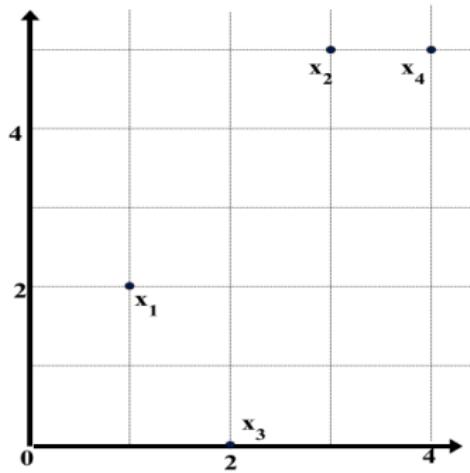
- Medida o función de similaridad:
 - Función que cuantifica la similaridad entre dos objetos asignando un valor real.
 - Medida de que tan parecidos son dos objetos de datos: A mayor valor, más similares.
 - Usualmente cae en el rango de $[0, 1]$: 0 indica no similaridad y 1 que son completamente similares.
- Distancia (medida de disimilaridad):
 - Medida numérica que indica que tan diferentes son dos objetos de datos.
 - En cierto sentido, es la inversa de la similaridad: Menor valor, más parecidos.
 - El mínimo valor de disimilaridad es usualmente 0 (completamente similares).
 - Rango $[0, 1]$ o $[0, \infty)$, dependiendo de la definición.
- Proximidad: Se refiere usualmente tanto a la similaridad como a la disimilaridad.

Matriz de distancias (disimilaridad)

$$\begin{pmatrix} 0 & & & \\ d(2, 1) & 0 & & \\ \vdots & \vdots & \ddots & \\ d(n, 1) & d(n, 2) & \dots & 0 \end{pmatrix}$$

- n observaciones, pero se registran solo las distancias $d(i, j)$ (por lo general métrica)
- Usualmente simétrica.
- Las funciones de distancias son usualmente distintas para variables de tipo cuantitativa, categórica (nominal u ordinal), vectores, etc.

Ejemplo - Matriz de distancias (disimilaridad)



Data Matrix

point	attribute1	attribute2
x_1	1	2
x_2	3	5
x_3	2	0
x_4	4	5

Dissimilarity Matrix (by Euclidean Distance)

	x_1	x_2	x_3	x_4
x_1	0			
x_2		3.61		
x_3			2.24	5.1
x_4				0
				4.24

Distancias con Datos Numéricos: Distancias de Minkowski

$$d(i, j) = \left[\sum_{k=1}^M |x_{ik} - x_{jk}|^p \right]^{\frac{1}{p}}$$

donde $i = (x_{i1}, \dots, x_{iM})$ y $j = (x_{j1}, \dots, x_{jM})$ son objetos de datos de dimensión M , y p es el orden de la distancia (L_p norm).

Propiedades:

- $d(i, j) > 0$ si $i \neq j$, y $d(i, i) = 0$ (no negatividad)
- $d(i, j) = d(j, i)$ (simetría)
- $d(i, j) \leq d(i, k) + d(k, j)$ (desigualdad triangular)

Distancias con Datos Numéricos I

Casos especiales de la distancias de Minkowski:

- $p = 1$: (L_1 norm) Distancia de Manhattan (City Block)

$$d(i, j) = \sum_{k=1}^M |x_{ik} - x_{jk}|$$

- $p = 2$: (L_2 norm) Distancia euclidiana

$$d(i, j) = \left[\sum_{k=1}^M |x_{ik} - x_{jk}|^2 \right]^{\frac{1}{2}}$$

- $p \rightarrow \infty$: (L_{max} norm) Distancia suprema (Chebychev)

$$d(i, j) = \max_{1 \leq k \leq M} |x_{ik} - x_{jk}|$$

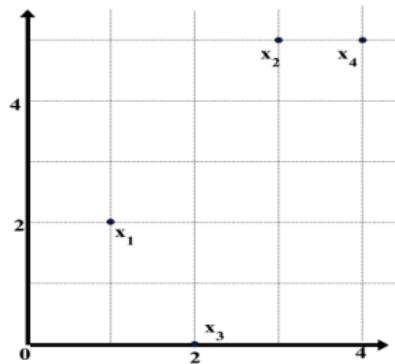
Distancias con Datos Numéricos II

Distancia de Canberra:

$$d(i, j) = \sum_{k=1}^M \frac{|x_{ik} - x_{jk}|}{|x_{ik} + x_{jk}|}$$

Ejemplo - Distancias con datos numéricos

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum (L_∞)

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Distancias con Datos Mixtos

- Gower: $d(i, j) = \sum_{k=1}^M w_k |x_{ik} - x_{jk}|$

donde, si el k -ésimo atributo es nominal $|x_{ik} - x_{jk}| = 0$ cuando ambos valores son iguales, $|x_{ik} - x_{jk}| = 1$ si son distintos y $w_k = \frac{1}{M}$. Si el k -ésimo atributo es continuo entonces

$$w_k = \frac{1}{M \times \text{rango}(k\text{-ésimo atributo})}$$

Distancias con Datos Numéricos

- La función `dist` calcula la matriz de disimilaridad utilizando diversos tipos de distancia.

```
dist(x, method = "euclidean", p = 2)
```

`x`: matriz con los datos.

`method`: tipo de distancia: "euclidean", "maximum",
"manhattan", "canberra" or "minkowski".

`p`: La potencia de la distancia de Minkowski.

- Ejemplo:

```
x =matrix(rnorm(20), nrow=5)
```

```
dist(x)
```

```
dist(x, method= "manhattan",diag = TRUE)
```

```
dist(x, method= "maximum",upper = TRUE)
```

Distancias con Datos Mixtos

- La función `daisy` de la librería `cluster` calcula la matriz de disimilitud incluye la distancia gower.

```
daisy(x, metric = "euclidean", )  
x:      matriz con los datos.  
metric: tipo de distancia: "euclidean" (the default),  
        "manhattan" and "gower"
```

- Ejemplo:

```
x =matrix(rnorm(20), nrow=5)  
daisy(x)  
x = cbind(rnorm(10),sample(1:3,10,replace=T))  
x<-as.data.frame(x)  
x[,2]<-as.factor(x[,2])  
daisy(x)
```

Métodos de Particionamiento

- El conjunto de datos es particionado en un número de conglomerados k previamente especificado.
- Luego, iterativamente se va reasignando las observaciones a los conglomerados hasta que algún criterio de parada (función a optimizar) se satisface (suma de cuadrados dentro de los conglomerados sea la más pequeña).

Notación

Sea C_1, \dots, C_K conjuntos conteniendo los índices de las observaciones en cada cluster. Estos conjuntos satisfacen las siguientes propiedades

- ① $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$. En otras palabras, cada observación pertenece a por lo menos uno de los K clusters.
- ② $C_k \cap C_{k'} = \emptyset$ para todo $k \neq k'$. En otras palabras, los clusters no se translapan: ninguna observación pertenece a más de un cluster

Por ejemplo, si la i -ésima observación está en el k -ésimo cluster, entonces $i \in C_k$.

Método Moving Centres I

- Propuesto por Forgy (1965)
- El algoritmo es sencillo:
 - ① K individuos son seleccionados como centros iniciales del cluster (selección al azar, elección de los K primeros individuos, elección de 1 de cada n/k , etc.)
 - ② Se calculan las distancias entre cada individuo y el centro C_i de la etapa anterior y cada individuo es asignado al centroide más cercano de tal manera que se definen K grupos.
 - ③ Los K centros C_i son reemplazados con los centros de gravedad de los K clusters definidos en la etapa 2 (los centros de gravedad no son necesariamente individuos)
 - ④ Se verifica si los centros son estables (no cambian) o si el número de iteraciones fue completada.
 - Si la respuesta es que si, el proceso se detiene (usualmente luego de al menos 10 iteraciones)
 - Si no, vuelve al paso 2.

Variantes del método Moving Centres I

- Con el método de las K-medias de MacQueen (1967), los centros de gravedad de cada grupo son recalculados para cada nuevo individuo introducido en el grupo, en vez de esperar para la asignación de todos los individuos antes de recalcular los centros de gravedad. La convergencia es más rápida y puede ser completada en una sola iteración, pero depende del conjunto de datos.
- Con el método de dynamic clouds de Diday (1971) los clusters no están representados por un centro de gravedad sino por un subconjunto del grupo, llamado el Kernel, el cual si está bien estructurado será más representativo del cluster que un centro de gravedad.

Método de las K-Medias

- Es un enfoque simple y elegante de particionar un conjunto de datos en K clusters no traslapados.
- Para ejecutar el método de las K-Medias se debe de especificar primero el número deseado de clusters K . De esta manera el algoritmo asignará cada observación exactamente en un cluster.
- La idea detrás de este método es que un buen clustering es aquel en el que la variación dentro del cluster es lo más pequeña posible.
- El objetivo es minimizar la disimilaridad de los elementos dentro de cada conglomerado y maximizar la disimilaridad de los elementos que caen en diferentes conglomerados.

Variación dentro del cluster I

La variación dentro del cluster para el cluster C_k es una medida $W(C_k)$ de la cantidad en que las observaciones dentro del cluster difieren unas de otras. De aquí que se quiere resolver el problema

$$\underset{C_1, \dots, C_k}{\text{Minimizar}} \left\{ \sum_{k=1}^K W(C_k) \right\} \quad (1)$$

En otras palabras la fórmula indica que se desea particionar las observaciones en K clusters de tal forma que la variación total dentro del cluster sea lo más pequeña posible.

Resolver (1) parece razonable, pero es necesario definir la variación dentro del cluster.

Variación dentro del cluster II

Existen formas diversas para definir este concepto, pero la alternativa más común es la distancia euclíadiana:

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

donde $|C_k|$ denota el número de observaciones en el k -ésimo cluster.

Combinando (1) y (2) se obtiene el problema de optimización que define el método de K-medias,

$$\underset{C_1, \dots, C_K}{\text{Minimizar}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\} \quad (3)$$

Algoritmo del Método K-Medias I

Resolver la ecuación (3) es complicado, dado que existen casi K^n formas distintas de particionar n observaciones en K cluster. Un algoritmo que garantice un óptimo local es el siguiente

Algoritmo: Método de las K-Medias

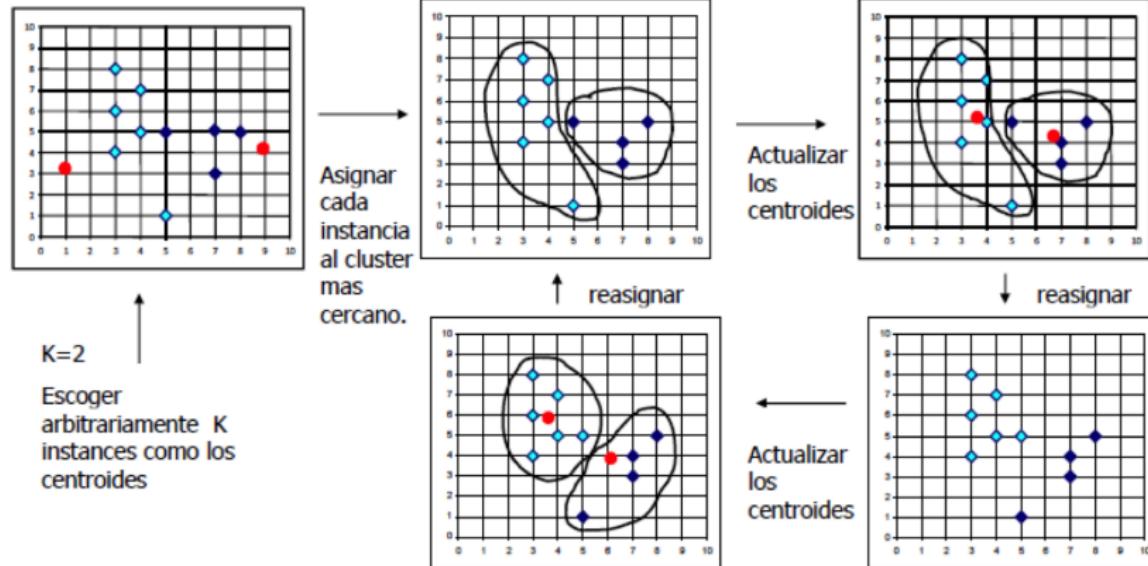
- ① Asigna aleatoriamente un número, de 1 a K , a cada una de las observaciones.
- ② Iterar hasta que la asignación de los cluster deje de cambiar
 - a) Para cada uno de los K cluster, calcular el centroide. El k -ésimo centroide es el vector con las p medias de las variables para las observaciones en el k -ésimo cluster.
 - b) Asignar cada observación al cluster donde el centroide esté más cerca (donde cercanía se encuentra definida por la distancia Euclidiana).

Algoritmo del Método K-Medias II

El algoritmo anterior garantiza que el valor del objetivo dado en (3) decrezca en cada etapa. Para entender el por qué, se evalúa la siguiente expresión:

$$\frac{1}{|C_K|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2 \quad (4)$$

Método de las K-Medias



K medias: Desventajas

- No se satisface el criterio de optimización globalmente, solo produce un óptimo local.
- El algoritmo de K-medias es computacionalmente rápido.
- Es sensible a la elección de los centroides iniciales.
- Es sensible a “outliers” .

Aplicación con R

- En R esta implementado en la función kmeans.

```
kmeans(x, centers)
```

x : es el conjunto de datos

centers : es el numero de conglomerados.

```
a=kmeans(iris[,1:4],3)
```

```
a$cluster
```

```
table(a$cluster)
```

```
table(a$cluster,iris[,5])
```

Aplicación con R

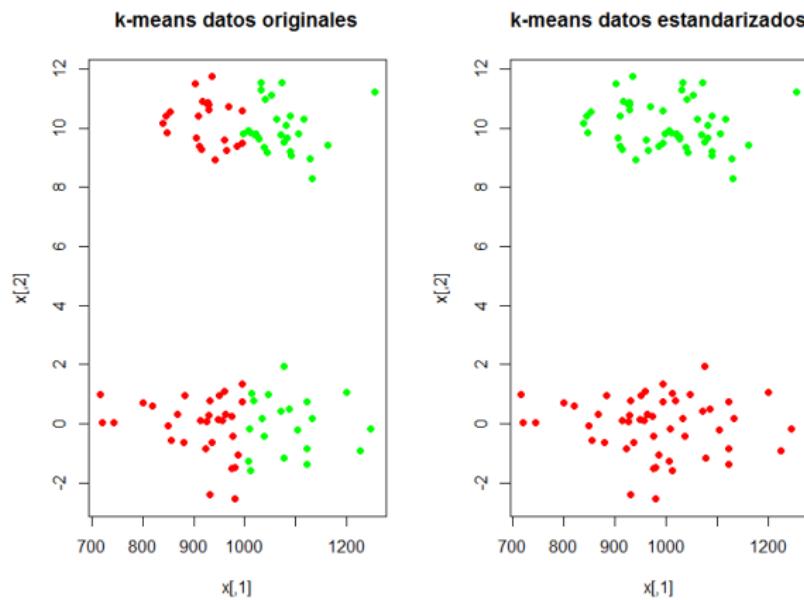
```
# Simulación
x=cbind(rnorm(100,1000,100),
c(rnorm(50),rnorm(50,10,1)))
plot(x,pch=16)

#k-means datos originales
res=kmeans(x,2)
plot(x,col=c("green","red")[res$cluster],pch=16)

# Estandarización
xs=scale(x)
plot(xs,pch=16)

#k-means datos estandarizados
res=kmeans(xs,2)
plot(x,col=c("green","red")[res$cluster],pch=16)
```

Aplicación con R



Aplicación con R

- Como ejemplo vamos a considerar el archivo distritos.sav el cual contiene 7 indicadores para los distritos de Lima.
 - distrito: Distrito
 - ocu_vivi: Hogares en cada vivienda
 - pobpjov: Porcentaje de población en pueblos jóvenes
 - sinelect: Porcentaje de población sin electricidad
 - sinagua: Porcentaje de población sin agua
 - pea1619: PEA entre 16 y 19 años
 - peam15: PEA menores de 15 años

Aplicación con R

```
library(foreign)
distritos=read.spss("distritos.sav",
use.value.labels=TRUE, max.value.labels=Inf, to.data.frame=TRUE)
colnames(distritos) <- tolower(colnames(distritos))
nombres=distritos[,1]
distritos=distritos[,-1]
rownames(distritos)=nombres
head(distritos)

    ocu_vivi pobpjov sinelect sinagua pea1619 pocprin peam15
Ate      1.15     5.3    27.60   51.10     3.9     1.1   63.48
Barranco 1.09     4.5    1.59    8.32     0.8     3.9   33.48
Breña    1.08     4.4    2.20    23.15     0.9     4.0   37.89
Carabayllo 1.10     5.1   30.13   38.09     4.5    12.6   63.65
Comas    1.20     5.9   10.92   24.27     3.8     9.4   60.37
Chorrillos 1.15     5.5   16.77   37.11     3.2    10.6   18.78
```

Aplicación con R

```
kmeans(scale(distritos),2)
```

```
K-means clustering with 2 clusters of sizes 15, 19
```

```
Cluster means:
```

	ocu_vivi	pobpjov	sinelect	sinagua	pea1619	pocprin	peam15
1	0.6635316	0.7536352	0.8957621	0.7981240	0.9978041	0.7677947	0.8420238
2	-0.5238407	-0.5949751	-0.7071806	-0.6300979	-0.7877401	-0.6061537	-0.6647556

```
Within cluster sum of squares by cluster:
```

```
[1] 67.25282 36.46842  
(between_SS / total_SS = 55.1 %)
```

Aplicación con R

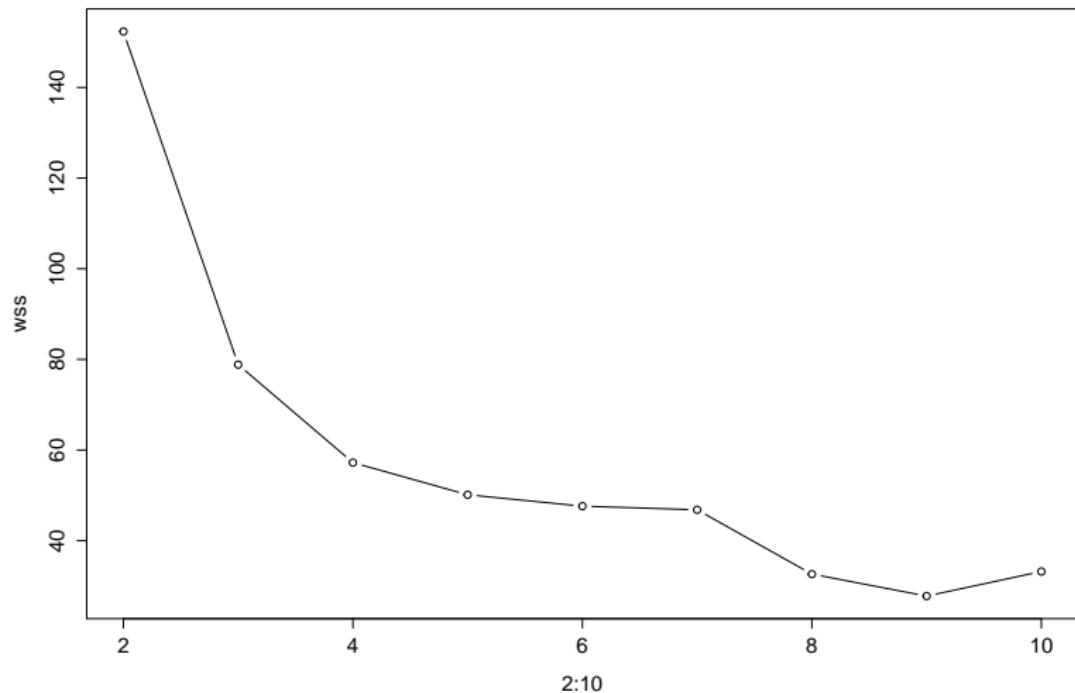
- Una medida para decidir el número de cluster es la suma de cuadrados entre clusters.

```
data(iris)

wss<-numeric()
for(h in 2:10){
  b<-kmeans(iris[,1:4],h)
  wss[h-1]<-b$tot.withinss
}

plot(2:10,wss,type="b")
```

K-means



Silueta

- El ancho de la silueta de la i -ésima observación es definida por:

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

- Donde, a_i denota la distancia promedio entre la observación i y todas las otras que están en el mismo conglomerado de i , y b_i denota la distancia promedio mínima de i a las observaciones que están en otros conglomerados.
- El valor de la silhouette varía entre -1 y +1.
- La medida de silueta será dada por $\bar{s} = \frac{1}{n} \sum_{i=1}^n sil_i$

Aplicación con R

- Se encuentra implementada en la función *silhouette* de la librería *cluster*.

```
silhouette(x, dist,...)
```

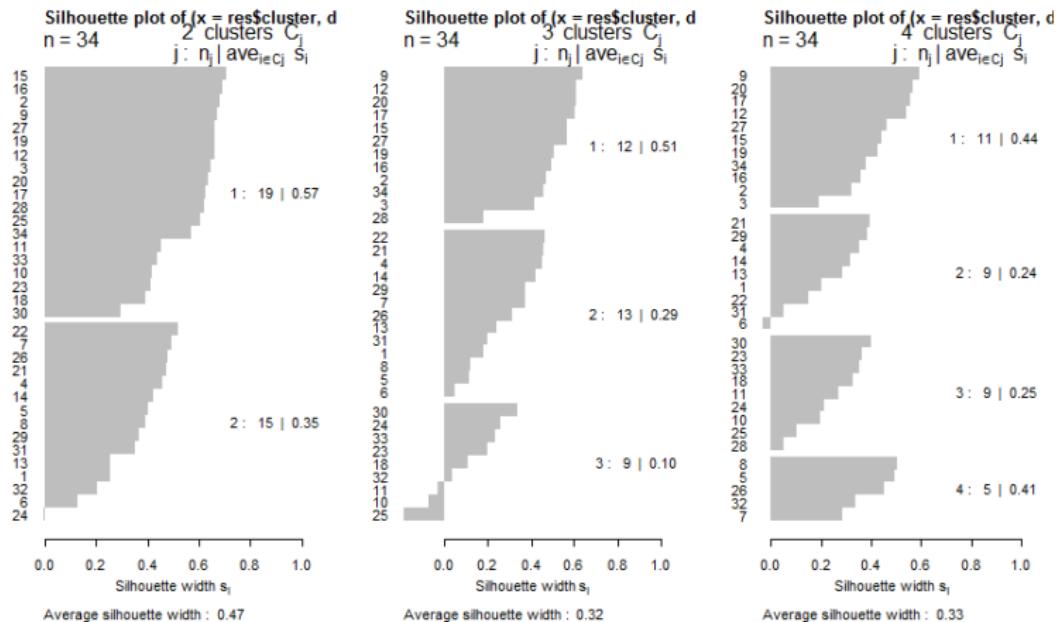
x: vector con códigos de los conglomerados

dist: matriz de disimilaridades

Aplicación con R

```
diss.distritos=daisy(scale(distritos))
par(mfrow=c(1,3))
for(h in 2:4){
  res=kmeans(scale(distritos),h)
  plot(silhouette(res$cluster,diss.distritos))
}
```

Aplicación con R



Criterio de Calinski-Harabasz

- Criterio de Calinski-Harabasz

$$\frac{(n - k) \times \text{Suma de cuadrados entre los grupos}}{(k - 1) \times \text{Suma de cuadrados dentro de los grupos}}$$

- Se encuentra implementado en la función calinhara de la librería fpc.

`calinhara(x, clustering, cn=max(clustering))`

x: conjunto de datos

clustering: vector con los conglomerados

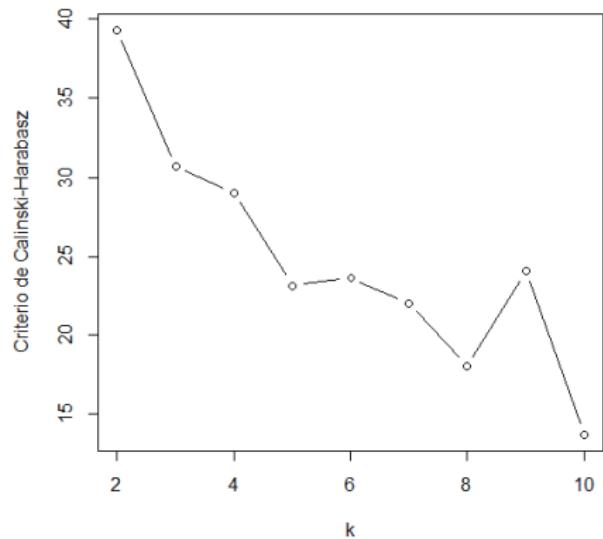
cn: número de conglomerados

Aplicación con R

```
ch<-numeric()
for(h in 2:10){
  res<-kmeans(scale(distritos),h)
  ch[h-1]<-calinhara(scale(distritos),res$cluster)
}

plot(2:10,ch,type="b",xlab="k",
ylab="Criterio de Calinski-Harabasz")
```

Aplicación con R



Aplicación con R

- Para elegir el número de clusters tambien podemos utilizar la función `kmeansruns` de la librería `fpc`.
- Esta función considera diversos puntos iniciales para el algoritmo de k-medias y estima el número de conglomerados utilizando diversos criterios.

```
kmeansruns(data,krange=2:10,criterion="ch",...)
```

`data`: conjunto de datos

`krange`: número de conglomerados a ser comparados

`criterion`: "ch" para el Criterio de Calinski-Harabasz y
"asw" para el criterio de promedio de ancho de siluetas

Aplicación con R

```
kmeansruns(scale(distritos),criterion="ch")
K-means clustering with 2 clusters of sizes 15, 19

Cluster means:
    ocu_vivi      pobpjov     sinelect     sinagua     pea1619     pocprin     peam15
1  0.6635316   0.7536352   0.8957621   0.7981240   0.9978041   0.7677947   0.8420238
2 -0.5238407 -0.5949751 -0.7071806 -0.6300979 -0.7877401 -0.6061537 -0.6647556

Within cluster sum of squares by cluster:
[1] 67.25282 36.46842
(between_SS / total_SS =  55.1 %)
```

Aplicación con R

```
kmeansruns(scale(distritos),criterion="asw")
K-means clustering with 2 clusters of sizes 15, 19

Cluster means:
    ocu_vivi      pobpjov     sinelect     sinagua     pea1619     pocprin     peam15
1  0.6635316   0.7536352   0.8957621   0.7981240   0.9978041   0.7677947   0.8420238
2 -0.5238407 -0.5949751 -0.7071806 -0.6300979 -0.7877401 -0.6061537 -0.6647556

Within cluster sum of squares by cluster:
[1] 67.25282 36.46842
(between_SS / total_SS =  55.1 %)
```

Aplicación con R

- La función `plotcluster` de la librería `fpc` genera un gráfico considerando diversos métodos de proyección.

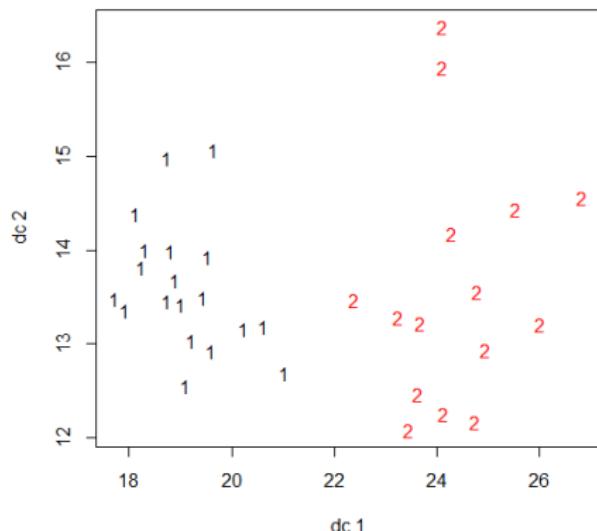
```
plotcluster(x, clvecd,...)
```

`x`: conjunto de datos

`clvecd`: vector con códigos de los conglomerados

Aplicación con R

```
res=kmeans(scale(distritos),2)  
plotcluster(distritos,res$cluster)
```



PAM

- El algoritmo de Particionamiento Alrededor de Medoides (PAM) se basa en buscar k observaciones representativas entre todas las observaciones del conjunto de datos.
- Estas observaciones son denominadas de medoides.
- Una vez encontrados los medoides se construyen los k conglomerados asignando cada observación al medoide más cercano

PAM

- El algoritmo PAM (Partitioning Around Medoids) tiene como objetivo encontrar medoides de modo que la disimilaridad total de todas las observaciones con respecto a su medoide más cercano sea mínima.
- Esto es, el objetivo es encontrar $\{m_1, \dots, m_k\} \subset \{1, \dots, n\}$ que minimice la función objetivo

$$\sum_{i=1}^n \min_{t=1, \dots, k} d(i, m_t)$$

PAM

- Luego, cada objeto es asignado al conglomerado correspondiente al medoide más cercano.
- Esto es la observación i es incluida en el conglomerado v_i cuando el medoide m_{v_i} es el más cercano a i que cualquier otro medoide m_w , esto es

$$d(i, m_{v_i}) \leq d(i, m_w) \forall w = 1, \dots, k$$

PAM

El algoritmo PAM considera dos pasos:

① BUILD

Construir los k medoides iniciales

- m_1 es la observación con la menor $\sum_{i=1}^n d(i, m_1)$
- m_2 minimiza la función objetivo lo más posible

⋮

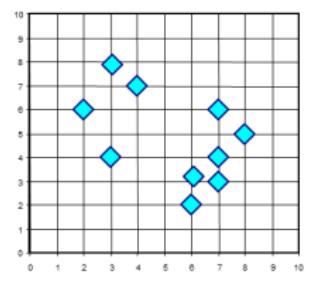
- m_k minimiza la función objetivo lo más posible

② SWAP

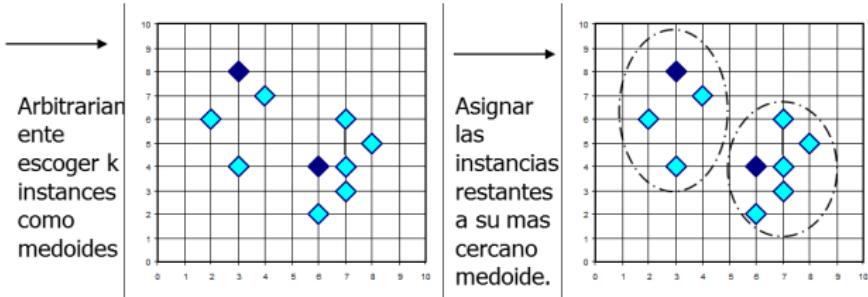
Considera todos los pares de objetos (i, j) donde

$$i \in \{m_1, \dots, m_k\} \text{ y } j \notin \{m_1, \dots, m_k\}$$

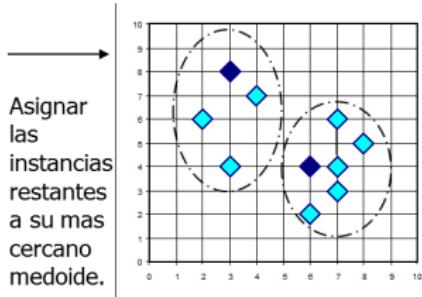
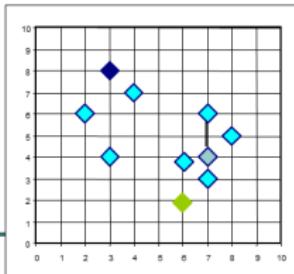
y se intercambia $i \leftrightarrow j$ que disminuya la función objetivo lo más posible.

**K=2**

Hacer el loop hasta que no haya cambios

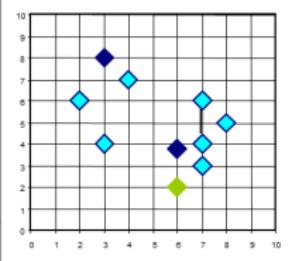


Total Cost = 26



Swapping O y O_{random}
Si se mejora la calidad

Calcular el costo total de swapping

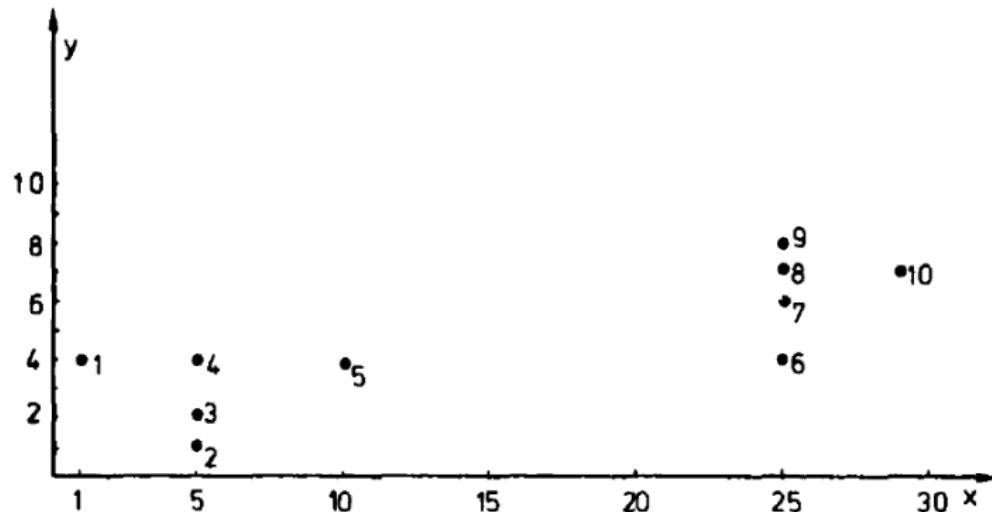


PAM

Table 1 Coordinates of the Objects of the Example of Figure 1

Number	x Coordinate	y Coordinate
1	1.0	4.0
2	5.0	1.0
3	5.0	2.0
4	5.0	4.0
5	10.0	4.0
6	25.0	4.0
7	25.0	6.0
8	25.0	7.0
9	25.0	8.0
10	29.0	7.0

PAM



PAM

Object Number	Dissimilarity from Object 1	Dissimilarity from Object 5	Minimal Dissimilarity	Closest Representative Object
1	0.00	9.00	0.00	1
2	5.00	5.83	5.00	1
3	4.47	5.39	4.47	1
4	4.00	5.00	4.00	1
5	9.00	0.00	0.00	5
6	24.00	15.00	15.00	5
7	24.08	15.13	15.13	5
8	24.19	15.30	15.30	5
9	24.33	15.52	15.52	5
10	28.16	19.24	<u>19.24</u>	5
Average 9.37				

PAM

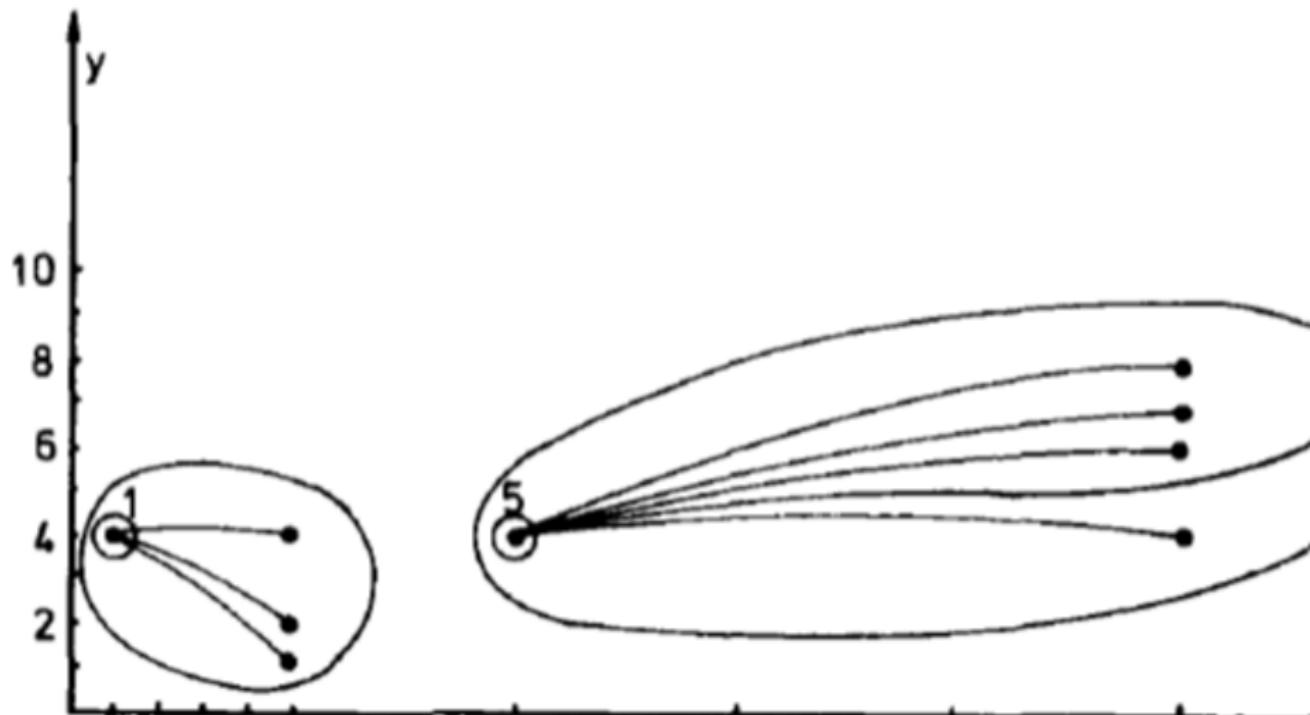


Table 1 Coordinates of the Objects of the Example of Figure 1

Number	x Coordinate	y Coordinate
1	1.0	4.0
2	5.0	1.0
3	5.0	2.0
4	5.0	4.0
5	10.0	4.0
6	25.0	4.0
7	25.0	6.0
8	25.0	7.0
9	25.0	8.0
10	29.0	7.0

Object Number	Dissimilarity from Object 1	Dissimilarity from Object 5	Minimal Dissimilarity	Closest Representative Object
1	0.00	9.00	0.00	1
2	5.00	5.83	5.00	1
3	4.47	5.39	4.47	1
4	4.00	5.00	4.00	1
5	9.00	0.00	0.00	5
6	24.00	15.00	15.00	5
7	24.08	15.13	15.13	5
8	24.19	15.30	15.30	5
9	24.33	15.52	15.52	5
10	28.16	19.24	19.24	5
Average 9.37				

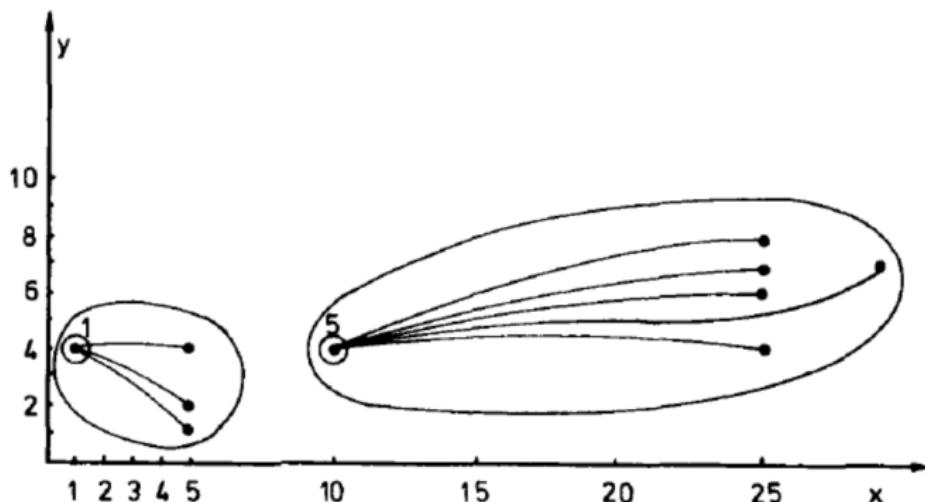
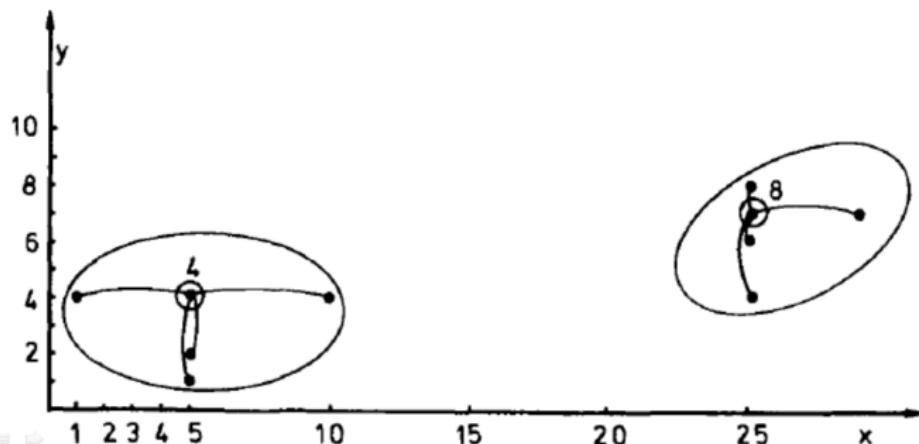


Table 3 Assignment of Objects to Two Other Representative Objects

Object Number	Dissimilarity from Object 4	Dissimilarity from Object 8	Minimal Dissimilarity	Closest Representative Object
1	4.00	24.19	4.00	4
2	3.00	20.88	3.00	4
3	2.00	20.62	2.00	4
4	0.00	20.22	0.00	4
5	5.00	15.30	5.00	4
6	20.00	3.00	3.00	8
7	20.10	1.00	1.00	8
8	20.22	0.00	0.00	8
9	20.40	1.00	1.00	8
10	24.19	4.00	4.00	8
Average 2.30				



PAM

- La función pam de la librería cluster implementa este método.

```
pam(x, k, diss = inherits(x, "dist"),  
metric = "euclidean",...)
```

x : matriz de datos o de disimilaridades

k : número de clusters

diss : Si es TRUE x es una matriz de disimilaridades

metric : "euclidean" ó "manhattan".

Aplicación con R

```
res=pam(scale(distritos),2)
res
```

Medoids:

ID	ocu_vivi	pobpjov	siselect	sinagua	pea1619	pocprin	peam15
SJM 22	0.5141271	0.5714769	1.0211988	1.1196499	1.2455772	1.0645749	0.9067534
Mag 15	-0.5317041	-0.7947101	-0.8085748	-0.6737858	-0.8939102	-0.8848637	-0.8586552

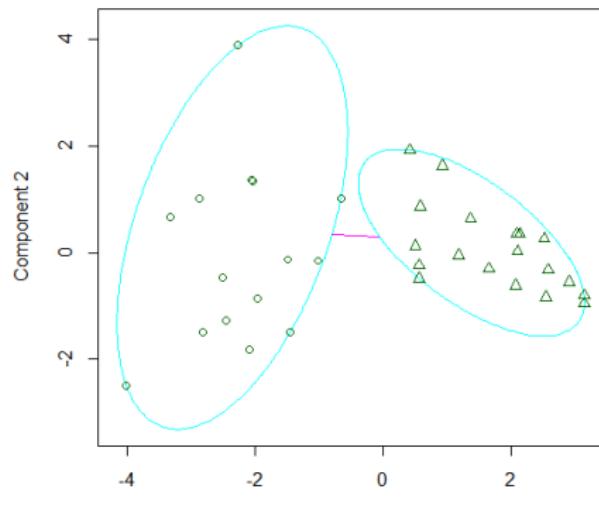
Objective function:

build	swap
1.648834	1.598154

```
plot(res)
```

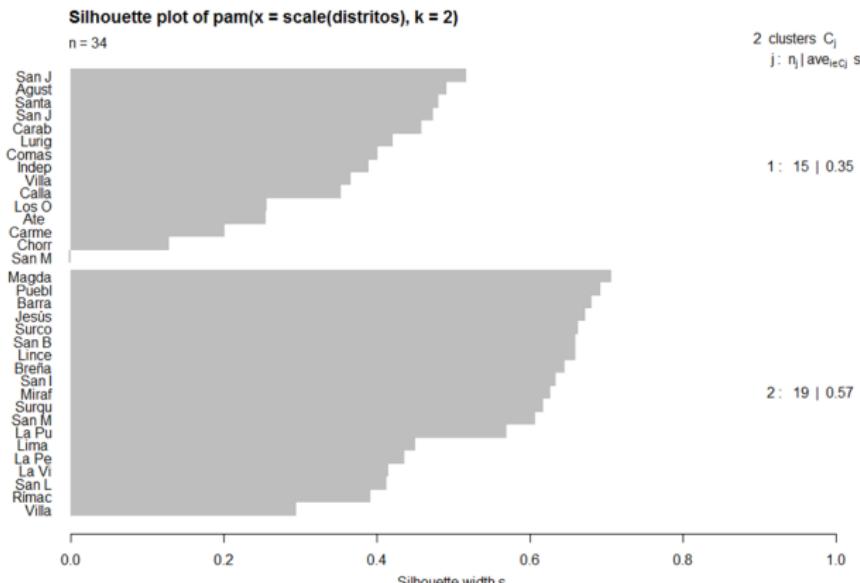
Aplicación con R

```
clusplot(pam(x = scale(distritos), k = 2))
```



These two components explain 86.96 % of the point variability.

Aplicación con R

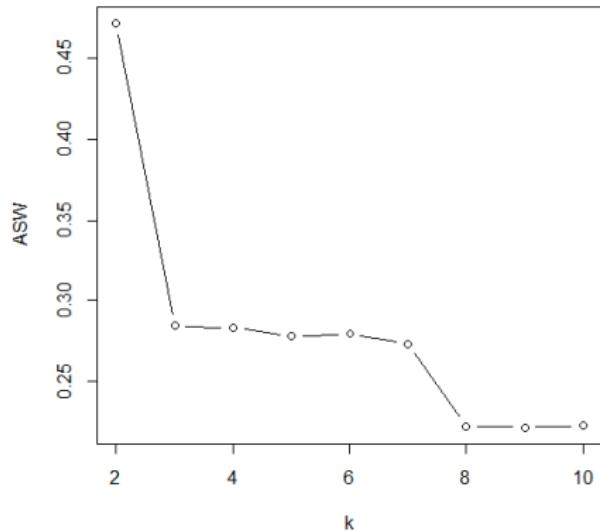


Aplicación con R

```
asw<-numeric()
for(h in 2:10){
  res<-pam(scale(distritos),h)
  asw[h-1]<-res$silinfo$avg.width
}

plot(2:10,asw,type="b",xlab="k",ylab="ASW")
```

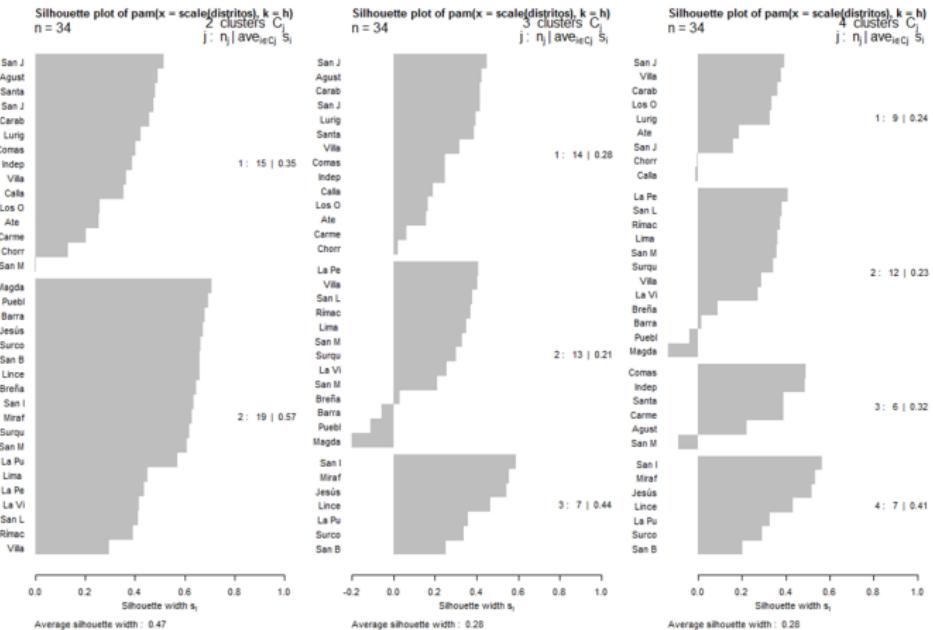
Aplicación con R



Aplicación con R

```
par(mfrow=c(1,3))
for(h in 2:4){
  res=pam(scale(distritos),h)
  plot(res,which.plots=2)
}
```

Aplicación con R

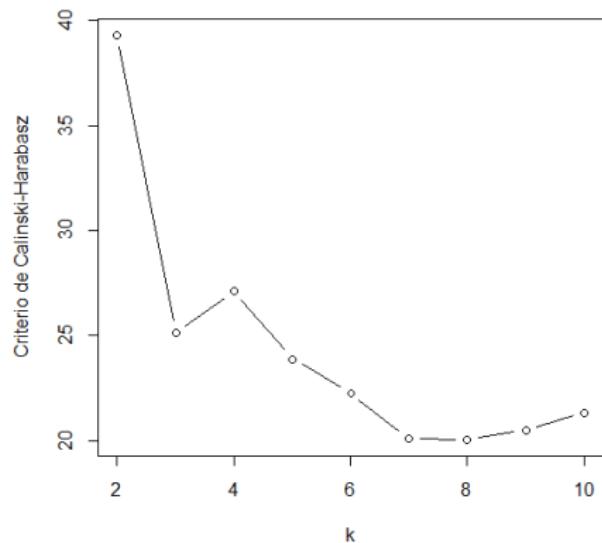


Aplicación con R

```
ch<-numeric()
for(h in 2:10){
  res<-pam(scale(distritos),h)
  ch[h-1]<-calinhara(scale(distritos),res$clustering)
}

plot(2:10,ch,type="b",xlab="k",
  ylab="Criterio de Calinski-Harabasz")
```

Aplicación con R



Aplicación con R

- Para elegir el número de clusters tambien podemos utilizar la función pamk de la librería fpc.
- Esta función considera diversos puntos iniciales para el algoritmo de k-medias y estima el número de conglomerados utilizando diversos criterios.

```
pamk(data,krange=2:10,criterion="asw",...)
```

data: conjunto de datos

krange: número de conglomerados a ser comparados

criterion: "ch" para el Criterio de Calinski-Harabasz y
"asw" para el criterio de promedio de ancho de siluetas

Aplicación con R

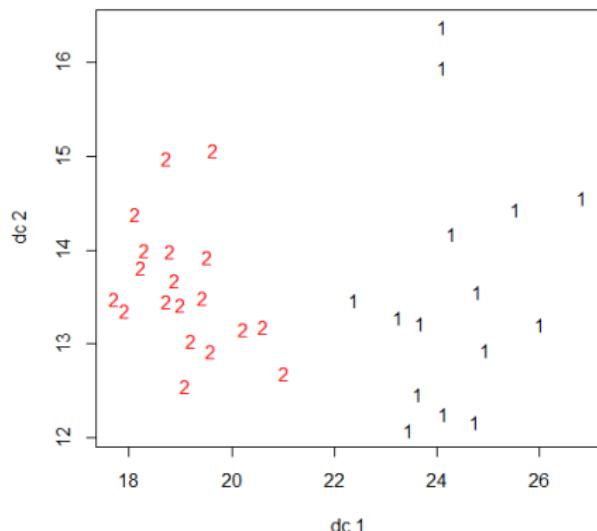
```
pamk(scale(distritos),criterion="asw")  
  
$pamobject  
Medoids:  
  
 ID   ocu_vivi    pobpjov   sinelect    sinagua    pea1619    pocprin    peam15  
SJM  22  0.5141271  0.5714769  1.0211988  1.1196499  1.2455772  1.0645749  0.9067534  
Magd 15 -0.5317041 -0.7947101 -0.8085748 -0.6737858 -0.8939102 -0.8848637 -0.8586552  
  
Objective function:  
  build      swap  
1.648834 1.598154  
$nc  
[1] 2  
  
$crit  
[1] 0.0000000 0.4716626 0.2841849 0.2830814 0.2777934 0.2791337  
    0.2730802 0.2222431 0.2214481 0.2224316
```

Aplicación con R

```
pamk(scale(distritos),criterion="ch")  
  
$pamobject  
Medoids:  
  
 ID   ocu_vivi    pobpjov   sinelect    sinagua    pea1619    pocprin    peam15  
SJM  22  0.5141271  0.5714769  1.0211988  1.1196499  1.2455772  1.0645749  0.9067534  
Magd 15 -0.5317041 -0.7947101 -0.8085748 -0.6737858 -0.8939102 -0.8848637 -0.8586552  
  
Objective function:  
  build      swap  
1.648834 1.598154  
$nc  
[1] 2  
  
$crit  
[1] 0.00000 39.26795 25.11968 27.06260 23.85990 22.21844  
     20.06436 20.04850 20.48795 21.32444
```

Aplicación con R

```
res=pam(scale(distritos),2)  
plotcluster(distritos,res$clustering)
```



CLARA

- El algoritmo PAM necesita construir la matriz de disimilaridades completa, por lo que para conjuntos de datos grandes se puede volver impráctico.
- Para evitar este problema se propuso el algoritmo CLARA (Clustering Large Applications) que no considera la matriz de disimilaridades completa.

CLARA

El algoritmo CLARA considera los siguientes pasos:

- ① Repetir s veces
 - Extraer una muestra de tamaño r
 - Aplicar PAM para encontrar los medoides $\{m_1, \dots, m_k\}$
 - Calcular la función objetivo $\sum_{i=1}^n \min_{t=1, \dots, k} d(i, m_t)$
 - Guardar $\{m_1, \dots, m_k\}$ si la función objetivo < actual mejor valor obtenido de la función objetivo
- ② Asignar las n observaciones a los $\{m_1, \dots, m_k\}$ medoides genera la partición final.

CLARA

- La función clara de la librería cluster implementa este método.

```
clara(x, k, metric = "euclidean", samples = 5,  
sampszie = min(n, 40 + 2 * k),...)
```

x : matriz de datos

k : número de clusters

metric : "euclidean" ó "manhattan?

samples : número de muestras a ser tomadas

sampszie : tamaño de cada muestra

CLARA

```
res=clara(scale(distritos),2)
res

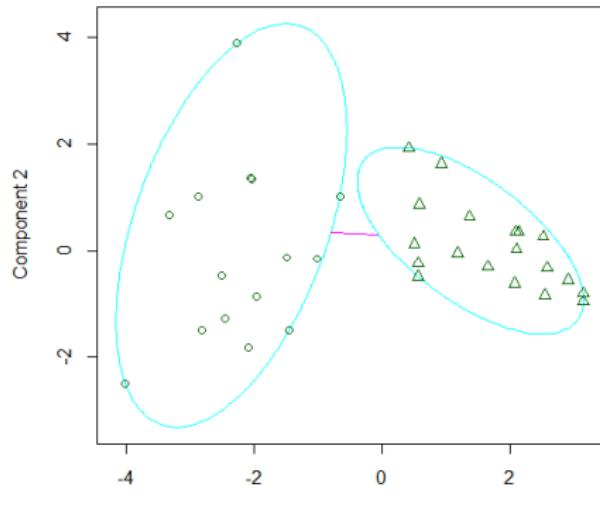
Medoids:
   ID   ocu_vivi   pobpjov   sinelect   sinagua   peal619   pocprin   peam15
SJM 22  0.5141271  0.5714769  1.0211988  1.1196499  1.2455772  1.0645749  0.9067534
Mag 15 -0.5317041 -0.7947101 -0.8085748 -0.6737858 -0.8939102 -0.8848637 -0.8586552

objective function:      1.598154
cluster sizes:          15 19
Best sample:

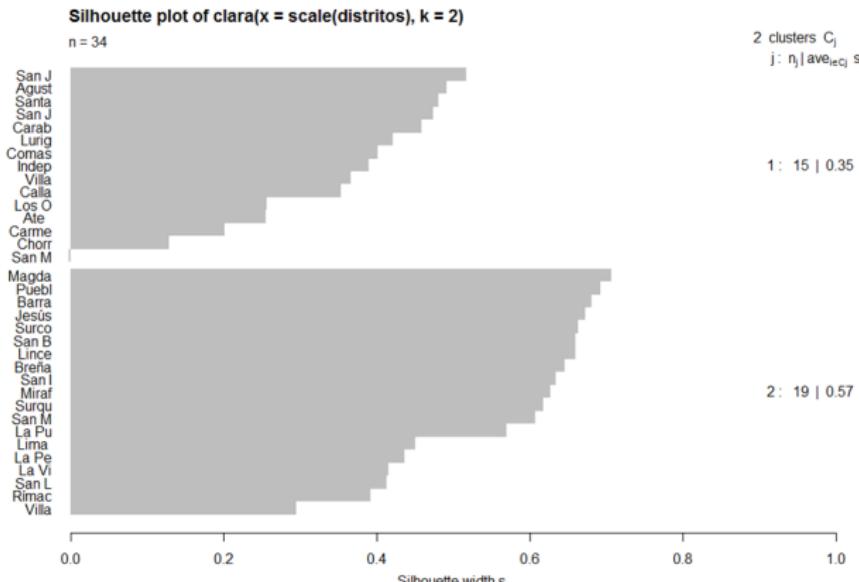
plot(res)
```

Aplicación con R

```
clusplot(clara(x = scale(districtos), k = 2))
```



Aplicación con R

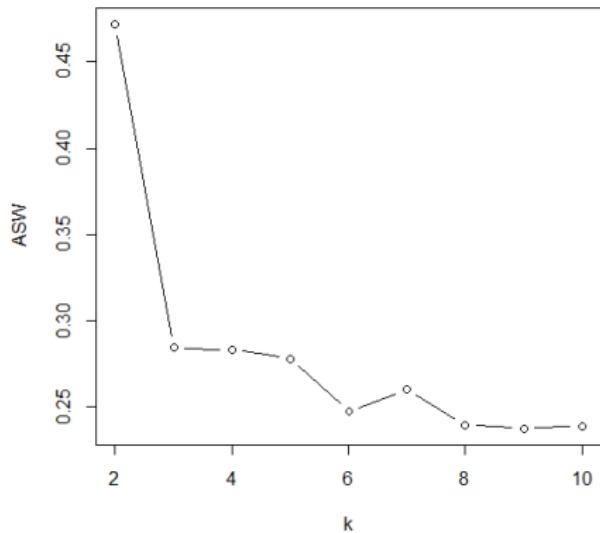


Aplicación con R

```
asw<-numeric()
for(h in 2:10){
  res<-clara(scale(distritos),h)
  asw[h-1]<-res$silinfo$avg.width
}

plot(2:10,asw,type="b",xlab="k",ylab="ASW")
```

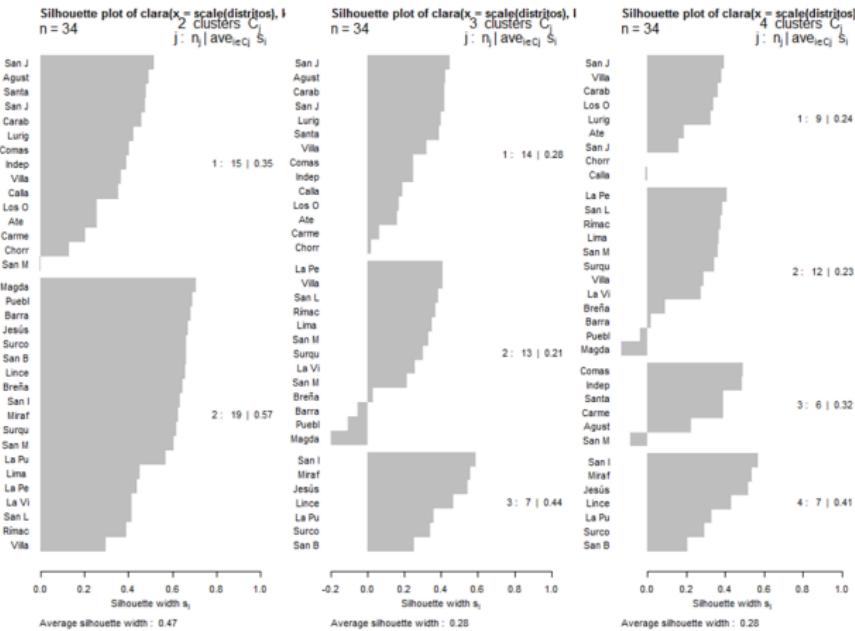
Aplicación con R



Aplicación con R

```
par(mfrow=c(1,3))
for(h in 2:4){
  res=clara(scale(distritos),h)
  plot(res,which.plots=2)
}
```

Aplicación con R

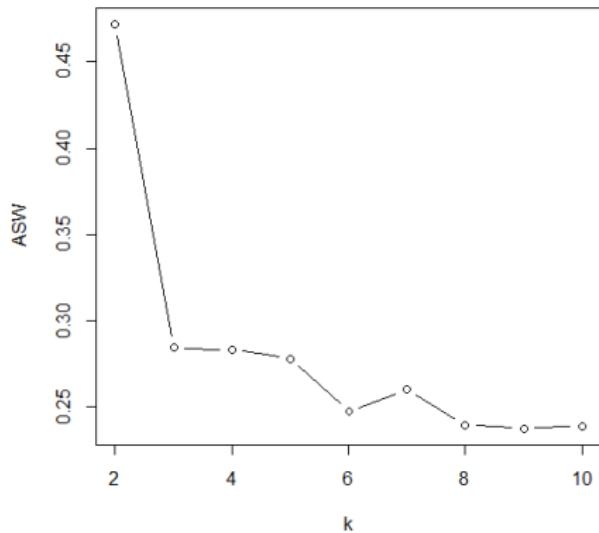


Aplicación con R

```
ch<-numeric()
for(h in 2:10){
  res<-clara(scale(distritos),h)
  ch[h-1]<-calinhara(scale(distritos),res$clustering)
}

plot(2:10,ch,type="b",xlab="k",
  ylab="Criterio de Calinski-Harabasz")
```

Aplicación con R



Aplicación con R

- Para elegir el número de clusters también podemos utilizar la función pamk de la librería fpc.
- Esta función considera diversos puntos iniciales para el algoritmo de k-medias y estima el número de conglomerados utilizando diversos criterios.

```
pamk(data,krange=2:10,criterion="asw",. usepam=TRUE...)
```

data: conjunto de datos

krange: número de conglomerados a ser comparados

criterion: "ch" para el Criterio de Calinski-Harabasz y "asw" para el criterio de promedio de ancho de siluetas

usepam: si es TRUE usa PAM si es FALSE usa CLARA

Aplicación con R

```
pamk(scale(distritos),criterion="asw",usepam=FALSE)

$pamobject
Medoids:

      ID   ocu_vivi   pobpjov  sinelect   sinagua   pea1619   pocprin   peam15
SJM  22  0.5141271  0.5714769  1.0211988  1.1196499  1.2455772  1.0645749  0.9067534
Magd 15 -0.5317041 -0.7947101 -0.8085748 -0.6737858 -0.8939102 -0.8848637 -0.8586552

Objective function:      1.598154
Clustering vector:
Cluster sizes:          15 19
Best sample:
$nc
[1] 2

$crit
[1] 0.0000000 0.4716626 0.2841849 0.2830814 0.2777934 0.2473110
    0.2599120 0.2396704 0.2376063 0.2385898
```

Aplicación con R

```
pamk(scale(distritos),criterion="ch",usepam=FALSE)

$pamobject
Medoids:

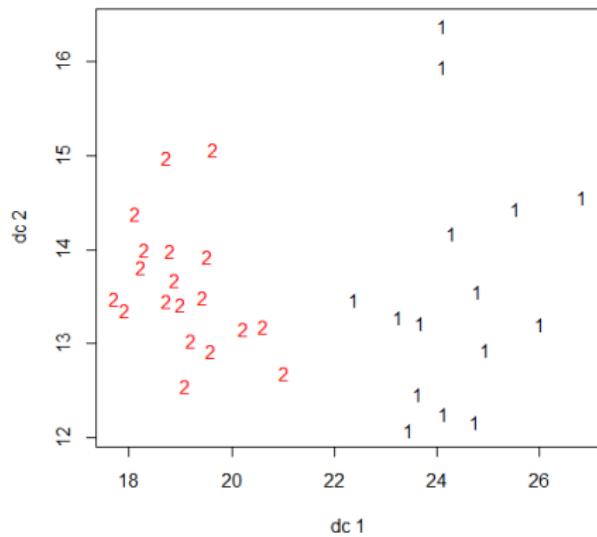
      ID   ocu_vivi    pobpjov   sinelect    sinagua    pea1619    pocprin    peam15
SJM  22  0.5141271  0.5714769  1.0211988  1.1196499  1.2455772  1.0645749  0.9067534
Magd 15 -0.5317041 -0.7947101 -0.8085748 -0.6737858 -0.8939102 -0.8848637 -0.8586552

Objective function:      1.598154
Clustering vector:
Cluster sizes:          15 19
Best sample:
$nc
[1] 2

$crit
[1] 0.00000 39.26795 25.11968 27.06260 23.85990 21.33027
     19.45147 20.34892 20.84134 21.75316
```

Aplicación con R

```
res=clara(scale(distritos),2)  
plotcluster(distritos,res$clustering)
```



FANNY

- Los algoritmos PAM y CLARA asignan cada observación a único conglomerado. Por ejemplo, un observación que este exactamente entre dos conglomerados será asignada solamente a uno de ellos.
- Sin embargo, con el método fuzzy divide cada objeto entre los conglomerados. Esta idea está implementada en el algoritmo FANNY (Fuzzy Analysis).

FANNY

- Para cada observación i y cada cluster v existirá un valor u_{iv} que indicará que tan fuertemente el objeto i pertenece al conglomerado v , estos valores deben cumplir
 - $u_{iv} \geq 0 \quad \forall i = 1, \dots, n \text{ y } \forall v = 1, \dots, k$
 - $\sum_{v=1}^k u_{iv} = 1, \quad \forall i = 1, \dots, n$

FANNY

- Luego, los valores u_{iv} serán estimados por la minimización de la función objetivo

$$\sum_{v=1}^k \frac{\sum_{i,j=1}^n u_{jv}^2 u_{iv}^2 d(ij)}{2 \sum_{j=1}^n u_{jv}^2}$$

- En esta expresión las disimilitudes $d(i, j)$ son conocidas y los u_{iv} son desconocidos.
- La minimización es llevada a cabo a través de un algoritmo iterativo, tomando en cuenta las restricciones mediante el uso de multiplicadores de Lagrange.

FANNY

- Cuando cada observación tiene la misma pertenencia a todos los conglomerados, estos es todos son $1/k$, se dice que el análisis de conglomerados es difuso completamente.
- Por otro lado si cada observación pertenece solamente a un conglomerado, se dice que el análisis de conglomerados es una partición.

FANNY

- Para medir este comportamiento se usa el indice de Dunn dado por

$$F_k = \sum_{i=1}^n \sum_{v=1}^k u_{iv}^2 / n$$

que toma su valor mínimo $1/k$ si la solución es completamente difusa y su valor máximo 1 si tenemos una partición.

- Se suele utilizar su versión normalizada

$$F'_k = \frac{kF_k - 1}{k - 1}$$

que toma valores entre 0 y 1.

Aplicación con R

- La función `fanny` de la librería `cluster` implementa este método.

```
fanny(x, k, metric = "euclidean", memb.exp=2,...)
```

`x` : matriz de datos o de disimilaridades

`k` : número de clusters

`metric` : "euclidean" ó "manhattan"

`memb.exp` : exponente a ser utilizado en la definición
de la función objetivo

Aplicación con R

```
res=fanny(scale(distritos),2)
res
```

Fuzzy Clustering object of class 'fanny' :

m.ship.expon. 2

objective 26.52194

tolerance 1e-15

iterations 16

converged 1

maxit 500

n 34

Fuzzyness coefficients:

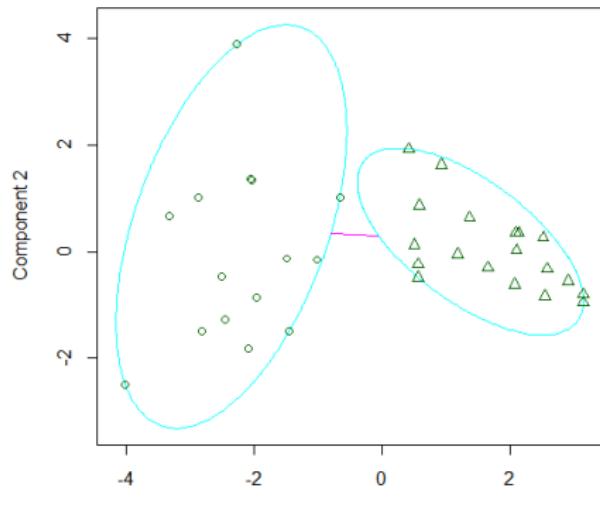
dunn_coeff normalized

0.6187843 0.2375687

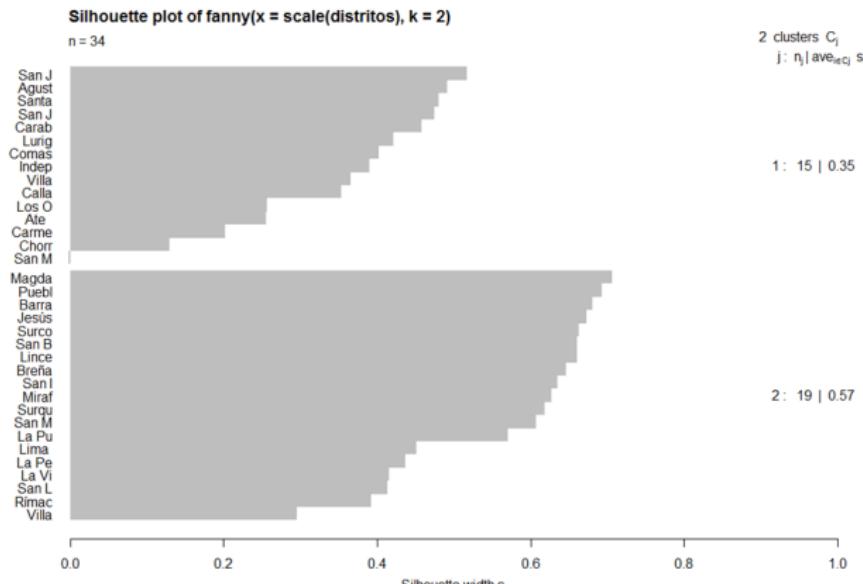
```
plot(res)
```

Aplicación con R

```
clusplot(fanny(x = scale(distritos), k = 2))
```



Aplicación con R

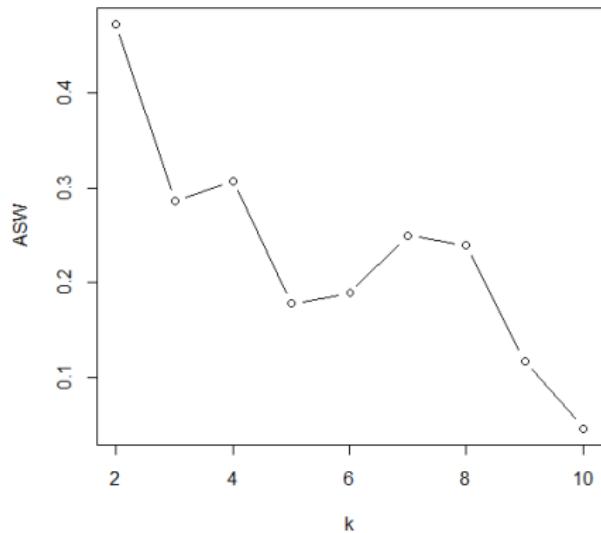


Aplicación con R

```
asw<-numeric()
for(h in 2:10){
  res<-fanny(scale(distritos),h,maxit=5000)
  asw[h-1]<-res$silinfo$avg.width
}

plot(2:10,asw,type="b",xlab="k",ylab="ASW")
```

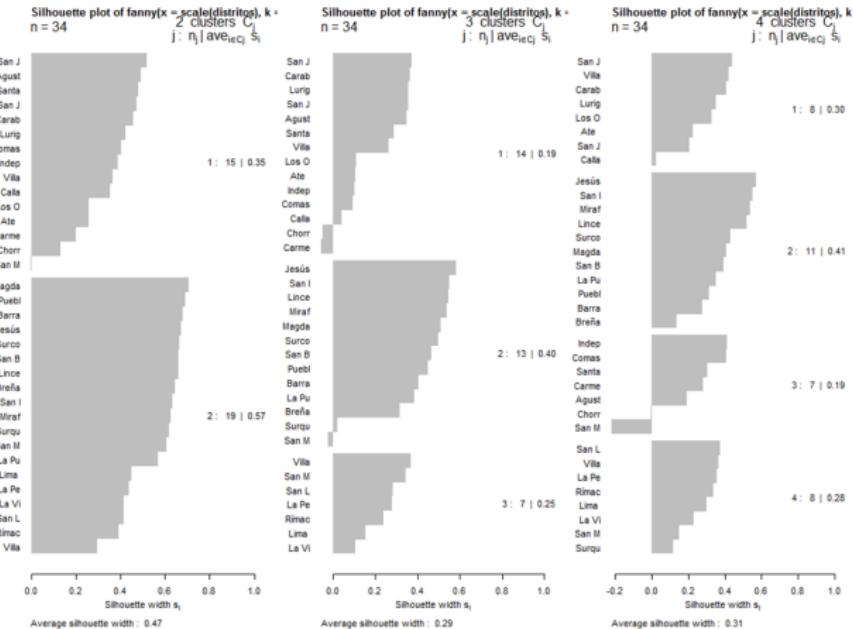
Aplicación con R



Aplicación con R

```
par(mfrow=c(1,3))
for(h in 2:4){
  res=fanny(scale(distritos),h)
  plot(res,which.plots=2)
}
```

Aplicación con R

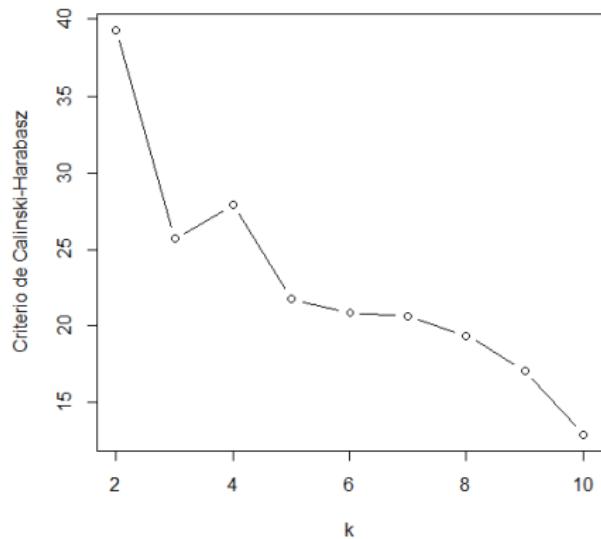


Aplicación con R

```
ch<-numeric()
for(h in 2:10){
  res<-fanny(scale(distritos),h,maxit=5000)
  ch[h-1]<-calinhara(scale(distritos),res$clustering)
}

plot(2:10,ch,type="b",xlab="k",
  ylab="Criterio de Calinski-Harabasz")
```

Aplicación con R



Métodos Jerárquicos

- En estos algoritmos se generan sucesiones ordenadas (jerarquías) de conglomerados. Puede ser juntando conglomerados pequeños en mas grandes o dividiendo grandes conglomerados en otros mas pequeños.
- La estructura jerárquica es representada en forma de un árbol y es llamada Dendograma.
- Se dividen en dos tipos:
 - Algoritmos jerárquicos aglomerativos. Inicialmente cada observación es un conglomerados.
 - Algoritmos jerárquicos divisivos. Inicialmente todas las observaciones están en un solo conglomerado.

Métodos Jerárquicos

- Suponiendo que tenemos datos de n individuos.
- Se empieza con n conglomerados.
- En cada paso se juntan los conglomerados más cercanos usando una medida de distancia entre conglomerados (linkage).

Métodos Jerárquicos

Sea $d(x, y)$ la distancia entre la observación x y la observación y . La distancia entre los conglomerados S y T puede ser calculada como:

- Linkage Simple: $\delta(S, T) = \min_{\{x \in S, y \in T\}} d(x, y)$
- Linkage Completo: $\delta(S, T) = \max_{\{x \in S, y \in T\}} d(x, y)$
- Linkage Promedio: $\delta(S, T) = \frac{1}{|S||T|} \sum_{\{x \in S, y \in T\}} d(x, y)$

Métodos Jerárquicos

- Linkage Centroide: $\delta(S, T) = d(\bar{x}, \bar{y})$
Donde \bar{x} y \bar{y} representan los centroides de S y T respectivamente.
- Linkage Mediana: $\delta(S, T) = \text{mediana}_{\{x \in S, y \in T\}} d(x, y)$

- Linkage de Mc Quitty: $\delta(S, T) = \frac{1}{|S|+|T|} \left[\sum_{x \in S} d(x, \bar{x}) + \sum_{y \in T} d(y, \bar{y}) \right]$
- Linkage de Ward: $\delta(S, T) = \sum_{x \in S} d(x, \bar{x})^2 + \sum_{y \in T} d(y, \bar{y})^2$

Se juntan el par conglomerados que produzcan la varianza más pequeña en el nuevo grupo.

Métodos Jerárquicos

- En R está implementado en la función `hclust`.

```
a=hclust(dist(scale(distritos)))
plot(a)
```

- La función `dist` calcula una matriz de disimilaridad para las instancias.

Métodos Jerárquicos

- Se puede recortar el dendrograma para encontrar el número de grupos deseado.

```
dis=dist(scale(distritos))
a=hclust(dis, method="ward.D")
plot(a)

b=cutree(a, k=3)
table(b)
```

Algoritmo Jerárquico Aglomerativo

Suponiendo que tenemos una matriz de datos $m \times n$. Se empieza con m conglomerados si se desea formar grupos de muestras (filas) o con n cluster si se quiere formar grupos de variables (columnas). En cada paso se juntan los cluster más cercanos usando una medida de distancia entre cluster (linkage), que ya se discutieron anteriormente. Entre estas distancias están:

- Linkage promedio: Promedio de las distancias de las observaciones de cada cluster.
- Linkage simple: La menor distancia entre las observaciones de cada cluster.
- Linkage completo: La mayor distancia entre las observaciones de cada cluster.

AGNES

- El algoritmo AGNES (Agglomerative Nesting) considera el método jerárquico aglomerativo, se encuentra implementado en la función agnes de la librería cluster.

```
agnes(x, metric = "euclidean", method = "average",
      ...)
x : conjunto de datos o matriz de dimilaridades
metric : "euclidean" o "manhattan"
method : linkage
```

AGNES

- El coeficiente de aglomeración permite medir la estructura de conglomerados de un conjunto de datos.
- Para cada observación i , denotemos por $m(i)$ la disimilaridad con el primer cluster con el cual fue unida, dividida por la disimilaridad de la última unión. Por lo tanto $0 \leq m(i) \leq 1$.
- El coeficiente de aglomeración se define como

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

AGNES

- Cuando el conjunto de datos posee una clara estructura de conglomerados, se espera que las disimilaridades entre conglomerados sean mucho más grandes que las disimilaridades dentro de los conglomerados. Por lo tanto, se espera que los valores de $m(i)$ sean bajos lo que hace que el valor de AC se acerque a su valor máximo 1.
- Se sabe que el coeficiente de aglomeración tiende a tener valores grandes a medida que el valor de n se incrementa, por lo tanto no debe ser usado para comparar conjuntos de datos de diferentes tamaños.

Algoritmo Jerárquico Aglomerativo

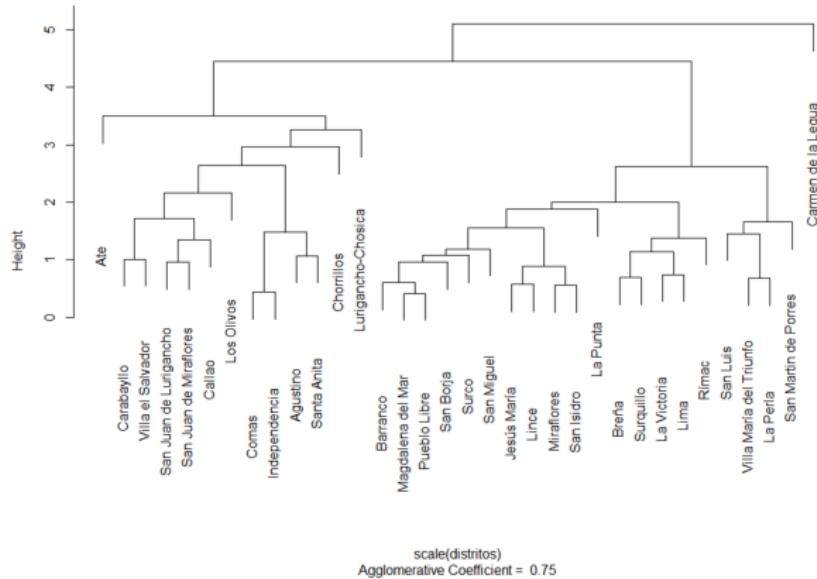
```
#Clustering jerarquico aglomerativo usando Agnes  
  
library(cluster)  
b<-agnes(iris[,1:4],metric="euclidean",method="ward")  
b=cutree(b,k=3)  
table(b,iris[,5])
```

Aplicación con R

```
res=agnes(scale(distritos),method="average")
res
Call: agnes(x = scale(distritos), method = "average")
Agglomerative coefficient:  0.7534594
Order of objects:
[1] Ate                      Carabayllo
[3] Villa el Salvador          San Juan de Lurigancho
[5] San Juan de Miraflores    Callao
[7] Los Olivos                 Comas
[9] Independencia              Agustino
[11] Santa Anita                Chorrillos
[13] Lurigancho-Chosica        Barranco
[15] Magdalena del Mar         Pueblo Libre
[17] San Borja                 Surco
[19] San Miguel                 Jesús María
[21] Lince                     Miraflores
[23] San Isidro                 La Punta
[25] Breña                     Surquillo
[27] La Victoria               Lima
[29] Rímac                     San Luis
[31] Villa María del Triunfo   La Perla
[33] San Martín de Porres      Carmen de la Legua
Height (summary):
  Min. 1st Qu. Median  Mean 3rd Qu. Max.
  0.4162  0.8894  1.3390  1.6390  1.9960  5.0970
plot(res)
```

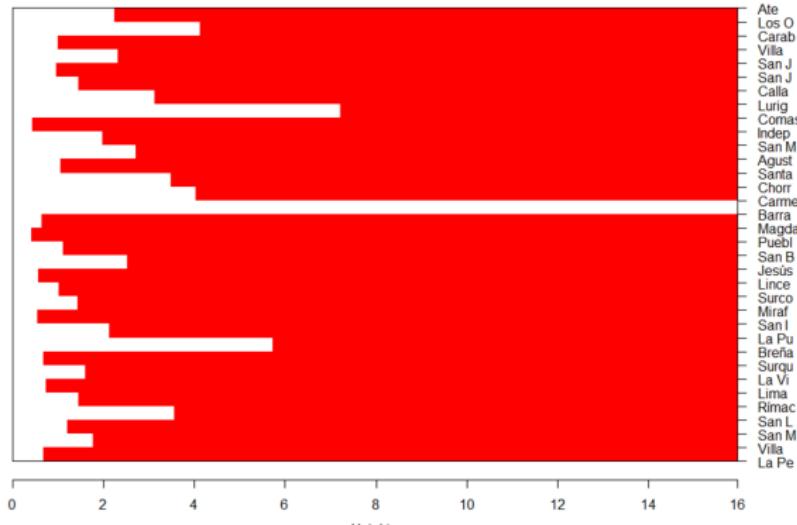
Aplicación con R

Dendrogram of agnes(x = scale(districtos), method = "average")



Aplicación con R

Banner of agnes(x = scale(districtos), method = "ward")



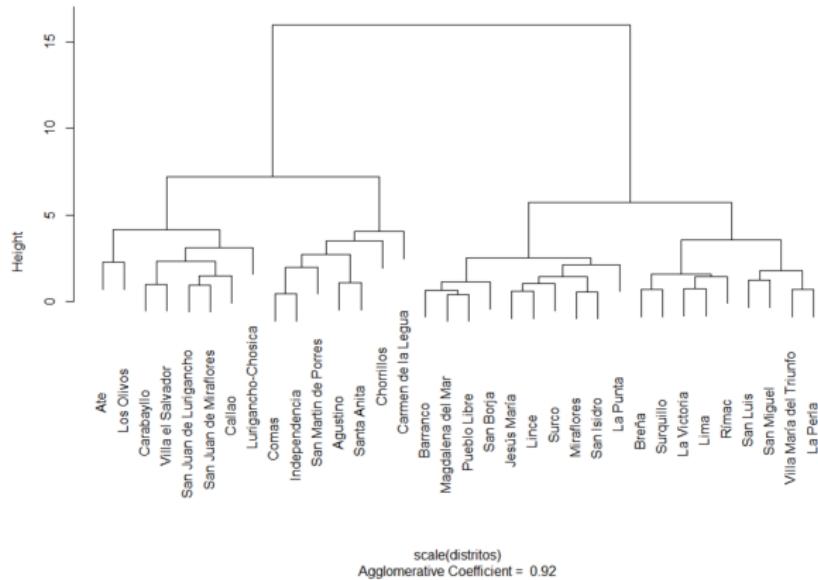
Agglomerative Coefficient = 0.92

Aplicación con R

```
res=agnes(scale(distritos),method="ward")
res
Call: agnes(x = scale(distritos), method = "ward")
Agglomerative coefficient:  0.9232787
Order of objects:
[1] Ate                      Los Olivos
[3] Carabayllo                Villa el Salvador
[5] San Juan de Lurigancho   San Juan de Miraflores
[7] Callao                     Lurigancho-Chosica
[9] Comas                      Independencia
[11] San Martin de Porres    Agustino
[13] Santa Anita               Chorrillos
[15] Carmen de la Legua       Barranco
[17] Magdalena del Mar        Pueblo Libre
[19] San Borja                 Jesús María
[21] Lince                      Surco
[23] Miraflores                San Isidro
[25] La Punta                  Breña
[27] Surquillo                 La Victoria
[29] Lima                       Rímac
[31] San Luis                   San Miguel
[33] Villa María del Triunfo  La Perla
Height (summary):
  Min. 1st Qu. Median  Mean 3rd Qu.  Max.
  0.4162  0.9582  1.4590  2.4260  2.7260 15.9500
plot(res)
```

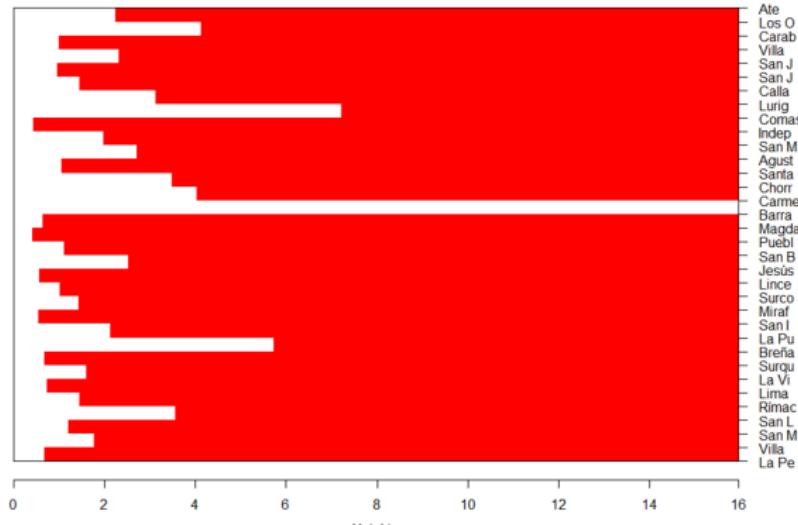
Aplicación con R

Dendrogram of agnes(x = scale(districtos), method = "ward")



Aplicación con R

Banner of agnes(x = scale(districtos), method = "ward")



Agglomerative Coefficient = 0.92

Aplicación con R

- La función `cutree` permite cortar un dendograma, en varios grupos.

```
cutree(tree, k = NULL, h = NULL)
```

tree : salida producida por agnes

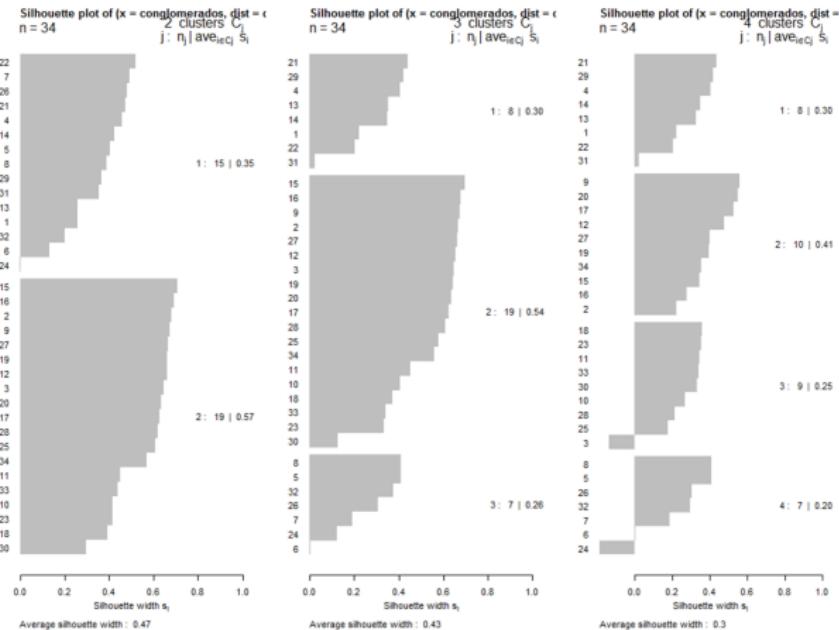
k : número de conglomerados

h : altura a la cual cortar el dendograma

Aplicación con R

```
diss.distritos=daisy(scale(distritos))
res=agnes(scale(distritos),method="ward")
par(mfrow=c(1,3))
for(h in 2:4){
conglomerados=cutree(res,h)
plot(silhouette(conglomerados,diss.distritos))
}
```

Aplicación con R



Algoritmo Jerárquico Divisivo: DIANA

- Se comienza con un solo cluster y se pasa a dividir los clusters en dos subgrupos cada vez.
- El algoritmo DIANA (Divisive Analysis) considera el método jerárquico divisivo, se encuentra implementado en la función diana de la librería cluster.

```
diana(x, metric = "euclidean",...)
```

x : conjunto de datos o matriz de dimilaridades

metric : "euclidean" o "manhattan"

DIANA: Ejemplo en R

```
b=diana(iris[,1:4],metric="euclidean")
plot(b,which=2)
b
b=cutree(b,k=3)
table(b)
table( b,iris[,5])
```

Aplicación con R

```
res=diana(scale(distritos))
> res
Order of objects:
[1] Ate                      Los Olivos
[3] Carabayllo                Villa el Salvador
[5] San Juan de Lurigancho   San Juan de Miraflores
[7] Lurigancho-Chosica       Comas
[9] Independencia            Agustino
[11] Santa Anita              Callao
[13] Chorrillos               Carmen de la Legua
[15] Barranco                 Magdalena del Mar
[17] Pueblo Libre              San Borja
[19] Surco                    Breña
[21] Jesús María              Lince
[23] Miraflores               San Isidro
[25] La Punta                 La Victoria
[27] Lima                     Surquillo
[29] Rímac                    San Luis
[31] San Miguel                Villa María del Triunfo
[33] La Perla                  San Martín de Porres
```

Aplicación con R

Height:

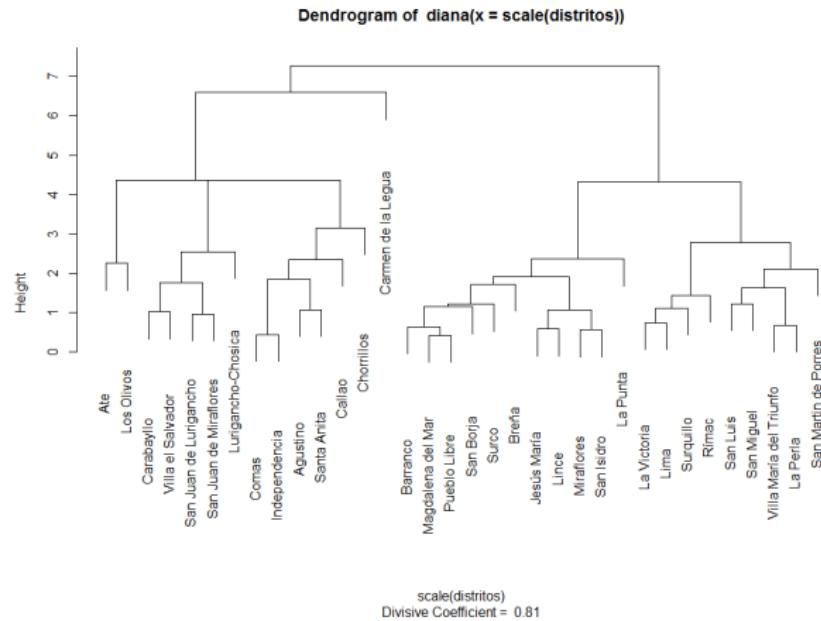
```
[1] 2.2483781 4.3582550 1.0098227 1.7546815 0.9582227 2.5449741 4.3582550  
[8] 0.4347900 1.8375651 1.0688382 2.3480693 3.1490065 6.5959718 7.2600487  
[15] 0.6308699 0.4162235 1.1375059 1.2025935 1.7143217 1.8960408 0.5711253  
[22] 1.0587986 0.5562151 2.3556905 4.3189741 0.7395195 1.1119242 1.4377573  
[29] 2.7748798 1.2138721 1.6338881 0.6754611 2.1027033
```

Divisive coefficient:

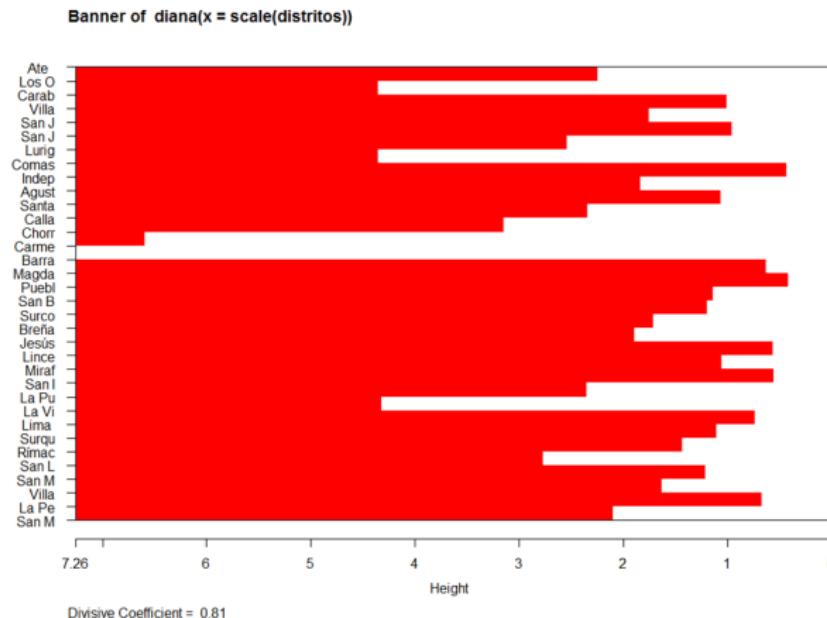
```
[1] 0.8131745
```

```
plot(res)
```

Aplicación con R



Aplicación con R



Aplicación con R

```
diss.distritos=daisy(scale(distritos))
res=diana(scale(distritos))
par(mfrow=c(1,3))
for(h in 2:4){
conglomerados=cutree(res,h)
plot(silhouette(conglomerados,diss.distritos))
}
```

Aplicación con R

