Mineria de Datos

Clustering I

Dr. Edgar Acuna
Departmento de Matematicas

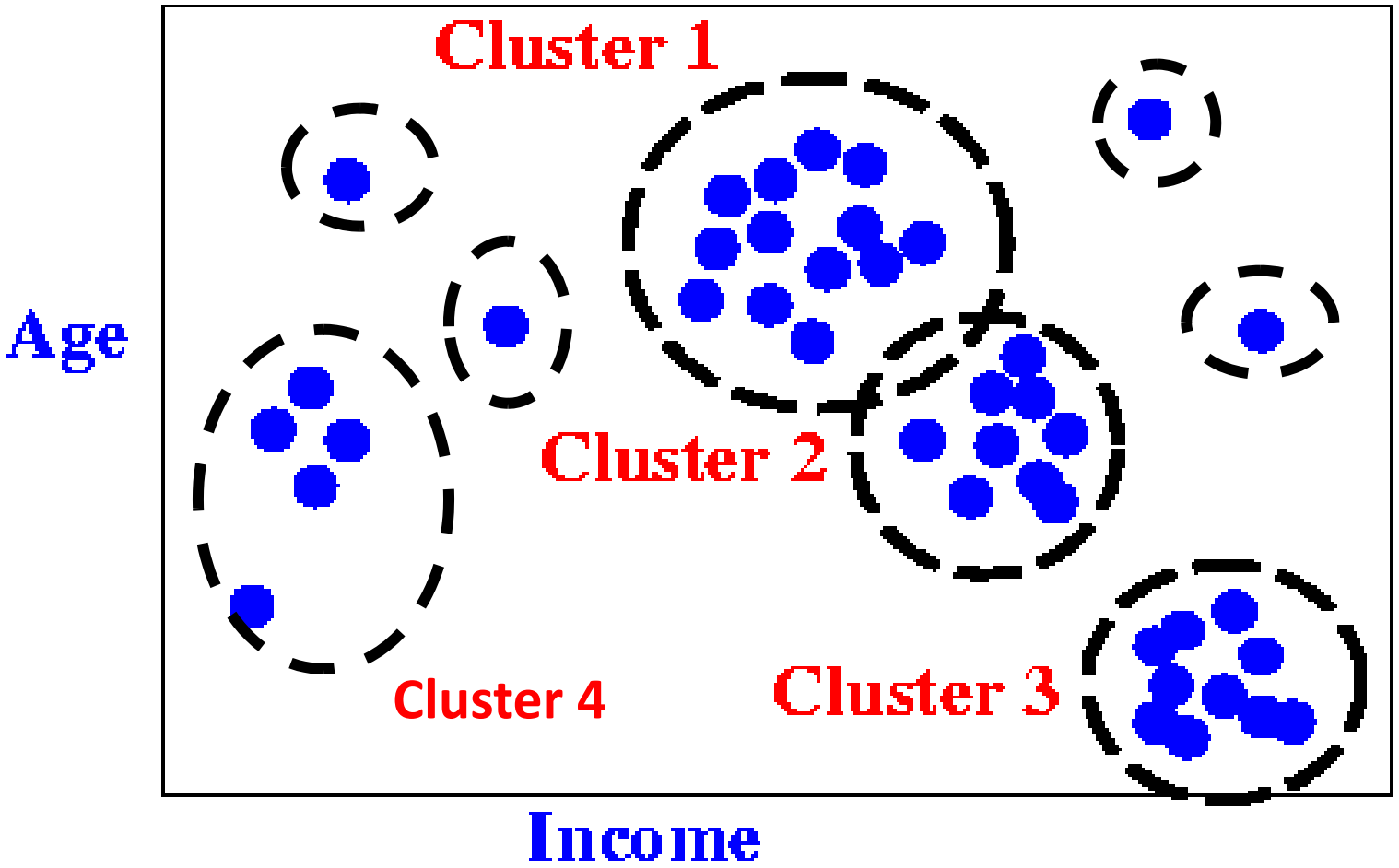
Universidad de Puerto Rico- Mayaguez

math.uprrm.edu/~edgar

Introduccion

- La idea de analisis de conglomerados (“clustering”) es agrupar muestras (filas) o features (columnas) o ambos a la vez, de acuerdo a la separación entre ellas determinada por una medida de distancia dada, llamada **medida de dissimilaridad**. Se supone que las clases a las que pertenecen las muestras no son conocidas.

- También es conocido con el nombre clasificacion no supervisado o segmentacion.


Cluster 1
Age
Cluster 2
Cluster 4
Cluster 3
Income

- Para cada muestra (fila) existe un vector de mediciones $\mathbf{X}=(X_1,\ldots X_p)$.

- El objetivo es identificar grupos de muestras similares basado en n mediciones observadas $\mathbf{X}_1=\mathbf{x}_1,\ldots., \mathbf{X}_n=\mathbf{x}_n$.

- Por ejemplo,

  Si las X's representan preguntas a un cuestionario a consumidores, uno podria identificar consumidores por sus estilos de compra.

  Si las X's representan variables socioeconomicas de paises, entonces uno podria formar grupos similares de paises de acuerdo a su bienestar socio-economico.

- Cuando el numero de columnas es bastantes grande se pueden formar tambien grupos de columnas con similar comportamiento. En consecuencia, se puede reducir la dimensionalidad , que es muy conveniente si se quiere usar luego un modelo para hacer predicciones. Pues es mucho mas conveniente predecir con 10 variables que con 100.

- También se puede aplicar conglomerados simultaneamente a filas y columnas (Bi-clustering) (ver paper de Alon, et al, 1999, Getz et al , 2000 y Lazaeronni y Owen, 2000)

Aspectos importantes en el analisis de conglomerados

i) Que variables usar?

ii) Que medida de dissimilaridad usar?.

iii) Qué método o algoritmo de conglomerado usar?.

iv) Cuantos conglomerados se deben formar?

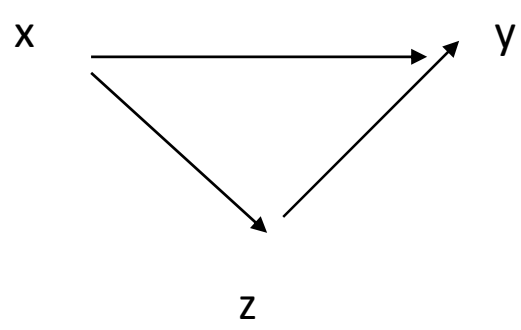
v) Cómo asignar las filas (o columnas) a los conglomerados?

vi) Cómo validar los conglomerados que se han formado?

Propiedades de medidas de dissimilaridad

- No-negatividad: $d(x,y) \geq 0$
- La distancia de una instancia asi mismo es 0, $d(x,x) = 0$
- Simetria: $d(x,y) = d(y,x)$
- Desigualdad Triangular:

  $d(x,y) \leq d(x,z) + d(z,y)$



Medidas de Dissimilaridad(variables continuas)

a) **Distancia Minkowski o norma Lp.**

$$D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{M}(x_i - y_i)^p\right)^{1/p}$$

Caso particulares:

- Distancia Euclideana: p=2, $D_2 = \sqrt{\left(\sum_{i=1}^{M}(x_i - y_i)^2\right)}$
-
- Distancia Manhattan o City-Block: $D_1 = \sum_{i=1}^{M}| x_i - y_i |$
- Distancia Chebychev, p=∝, $D_\infty = \max_{1\le i\le M}| x_i - y_i |$
- La distancia ponderada Minkoswki, será: $D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{M} w_i(x_i - y_i)^p\right)^{1/p}$

Medidas de Dissimilaridad (Cont)

b) Distancias basadas en formas cuadráticas.

$Q=(Q_{ij})$ es una matriz cuadrada MxM definida positiva de pesos entonces la distancia cuadrática entre x y y está dada por:

$$DQ(x,y) = [(x-y)'Q(x-y)]^{1/2} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{M}(x_i - y_i)Q_{ij}(x_j - y_j)}$$

Si $Q=V^{-1}$, V matríz de covarianza entre x y y se obtiene la distancia Mahalanobis.

c) Distancia Camberra.

$$D_{Can}(x,y) = \sum_{i=1}^{M}\frac{|x_i - y_i|}{|x_i + y_i|}$$

Si x_i y y_i son ambos ceros entonces el i-ésimo término de la suma se considera como cero.

Medidas de dissimilaridad (variables nominales)

Distancia Hamming. Sean **x** y **y** dos vectores de la misma dimension y con valores en Ω={0,1,2,……..k-1}, entonces la distancia Hamming (D_H) entre ellos se define como el número de entradas diferentes que tienen los dos vectores. Por ejemplo si x=(1,1,0,0) y y=(0,1,0,1) entonces D_H=2.

- Tambien se pueden usar para variables no binarias.Por ejemplo, si x=(0,1,3,2,1,0,1) y y=(1,1,2,2,3,0,2) entonces D_H(**x,y**)=4 .
- En el caso de variables binarias , la distancia Hamming, la distancia L2 y la distancia L1 coinciden.

Medidas de similaridad (Variables continuas))

a) Medida de correlación:

$$r(x,y)=\frac{\sum_{i=1}^{M}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{M}(x_i-\bar{x})^2\sum_{i=1}^{M}(y_i-\bar{y})^2}} \quad =\frac{(x-\bar{x})'(y-\bar{y})}{\|x-\bar{x}\|\|y-\bar{y}\|}$$

Nota: 1-s(x,y) puede ser considerado como una medida de dissimilaridad

b) Medida de Tanimoto. Se define por

$$S_T(x,y)=\frac{x'y}{\|x\|^2+\|y\|^2-x'y}$$

Tambien puede ser usada para variables nominales

Medidas de similaridad (variables nominales)[1]

i)La medida de Tanimoto. Sean X y Y dos vectors de longitud n_x y n_y respectivamente. Sea $n_{X \cap Y}$ que representa la cardinalidad de $X \cap Y$, entonces la medida de Tanimoto se define por

$$\frac{n_{X \cap Y}}{n_X + n_Y - n_{X \cap Y}}$$

Es decir, la medida de Tanimono es la razón del número de elementos que los vectores tienen en común entre el número de elementos distintos. En el caso de variables binarias, la medida de Tanimoto se reduce a

$$\frac{a+d}{a+2(c+b)+d}$$

Donde a representa el número de posiciones donde los vectores X y Y coinciden en tomar el valor 0, d representa el número de coincidencias donde X y Y valen ambos 1. Mientras que c representa el numero de parejas donde X=0 y Y=1 y b representa el número de parejas donde X=1 y Y=0.

Medidas de Similaridad(variables nominales) [2]

ii) **El coeficientes de coincidencias simple**. Definido por

$$\frac{a+d}{a+b+c+d}$$

El problema de esta medida es que incluye a a en el numerador, la cual indica ausencia de ambos factores.

iii) **La medida de Jaccard-Tanimoto**. Definida por

$$\frac{d}{b+c+d}$$

iv) **La medida de Russel -Rao**.

$$\frac{d}{a+b+c+d}$$

v) **La medida de Dice-Czekanowski**. Similar a la de Jaccard pero asigna peso doble a las coincidencias. Es decir

$$\frac{2d}{b+c+2d}$$

Ejemplo:

Si X=(1,0,1,0,0,1) y Y=(0,1,1,1,0,1).

	X=0	X=1
Y=0	1	1
Y=1	2	2

Entonces a=1, b=2, c=1 y d=2.
Luego la distancia de Tanimoto entre X y Y sera 3/9=1/3.
El coeficiente de coincidencias simples sera: 3/6=.5
La medida de Jaccard-Tanimoto sera: 2/5=
La medida de Russel-Rao sera:2/6=1/3
La medida de Dice-Czekanowski sera 4/7.

Librerias y funciones en R para hallar distancias

La libreria **stats** de R tiene una funcion **dist** que calcula la matríz distancia de una matríz de datos usando las distancias Euclideana, Manhattan, Chebychev, Camberra, Mimkowski y binary..

```
librería(stats)
x =matrix(rnorm(20), nrow=5)
    dist(x)
    dist(x, method=”manhattan”,diag = TRUE)
    dist(x, method=”maximum”,upper = TRUE)
# Ejemplo de distancia Camberra entre dos vectores
    x = c(0, 0, 1, 1, 1, 1)
    y = c(1, 0, 1, 1, 0, 1)
    dist(rbind(x,y), method="canberra")
```

La libreria **cluster** de R tiene una función **daisy** que calcula la

matriz distancia de una matriz de datos usando

las distancias euclideana y manhattan y considerando ademas distintos tipos de variables mediante el uso del coeficiente de Gower.

La distancia Gower entre los registros (filas) **x** y **y** de una matriz de dimension n por p, se define como

$$D_G(x, y) = \sum_{i=1}^{p} w_i ! x_i - y_i !$$

Donde, si el i-esimo atributo es nominal xi-yi=0, cuando ambos valores son iguales y 1 si son distintos, y los pesos w_i=1/p. Si el i-esimo atributo es continuo entonces w_i=1/(p*rango de i-esima columa

La libreria **cluster** de R tiene una función **daisy** que calcula la matriz distancia de una matriz de datos usando las distancias euclideana y manhattan y considerando ademas distintos tipos de variables mediante el uso del coeficiente de Gower.

La distancia Gower entre los registros (filas) **x** y **y** de una matriz de dimension n por p, se define como

$$D_G(x, y) = \sum_{i=1}^{p} w_i ! x_i - y_i !$$

Donde, si el i-esimo atributo es nominal xi-yi=0, cuando ambos valores son iguales y 1 si son distintos, y los pesos w_i=1/p. Si el i-esimo atributo es continuo entonces w_i=1/(p*rango de i-esima columa

```
> matx
   col1 col2      col3    col4
1 Hombre  1.3 Desempleo Soltero
2 Hombre  2.7 Desempleo  Casado
3  Mujer  4.5    Empleo   Viudo
4 Hombre  4.9    Empleo  Casado
> daisy(matx,metric="gower")
Dissimilarities :
         1         2         3
2 0.3472222
3 0.9722222 0.8750000
4 0.7500000 0.4027778 0.5277778
Metric :  mixed ;  Types = N, I, N, N
Number of objects : 4
```

La función **cor** de R calcula la matríz de correlaciones entre las columnas de una matríz y

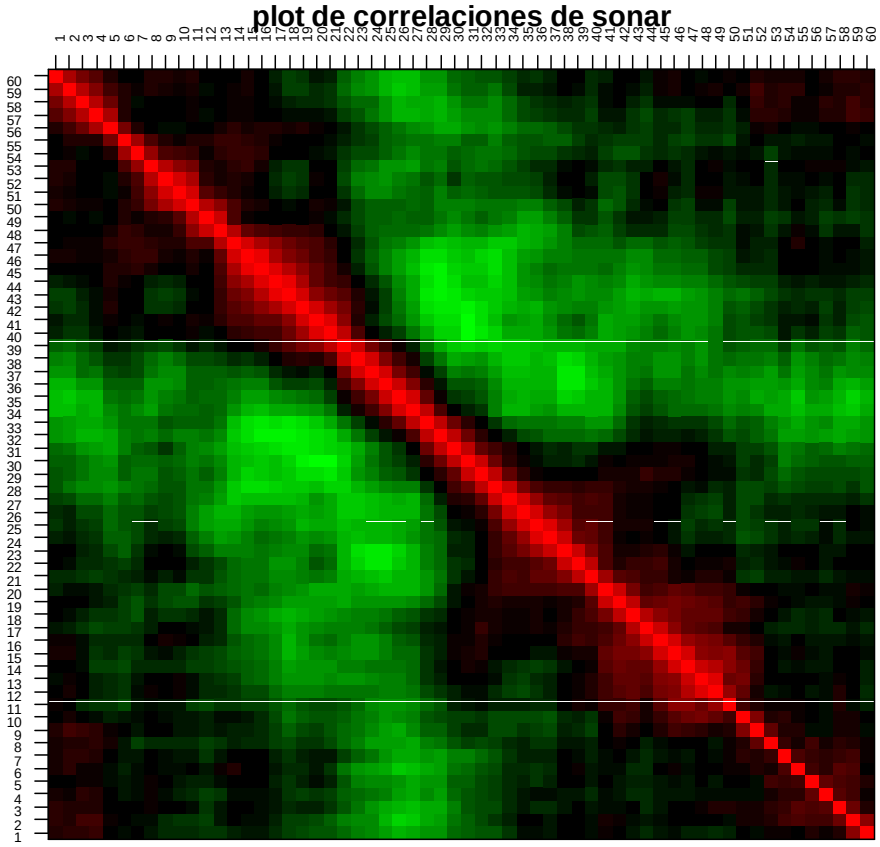
la función **plot.cor** de la librería **sma** permite hacer un "heatmap" de las correlaciones.

```
>library(sma)
>plot.cor(as.matrix(sonar[,-61]),new=T,labels=T)
> help(plot.cor)
```

Plot de la matríz de corrrelaciones (sonar)

- Sonar, tiene 60 columnas y 208 filas.

- La parte **roja** de la grafica indica que la correlación es bien alta y positiva, la parte **verde** indica que es bien alta pero negativa y la parte **negra** indica que la correlación es **cerca de cero**.



Distancias entre clusters

Despues de elegir la medida de similaridad se transforma la matriz de datos n x p en una matriz distancia o de dissimilaridad

D = (d_{ij}) de orden n x n para las n muestras a ser agrupadas. Basado en esta matriz se debe determinar una medida de distancia entre dos conglomerados (clusters) cualesquiera. Entre estas medidas están:

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:

$$\partial(S,T)=\frac{1}{|S||T|}\sum_{x\in S, y\in T} d(x,y)$$

Donde |S|, y |T| representan la cardinalidad de S y T.

Linkage centroide: $\delta(S,T)= d(\bar{x},\bar{y})$

Donde $\bar{x}$ y $\bar{y}$ representan los cenroides de S y T respectivamente

Distancias entre clusters

Linkage Mediana: $\delta(S,T)=\text{median}_{\{x\in S, y\in T\}} d(x,y)$

Linkage de Mc Quitty:

$$\partial(S,T)=\frac{1}{|S|+|T|}[\sum_{x\in S} d(x,\bar{x})+\sum_{y\in T} d(y,\bar{y})$$

Linkage de Ward:

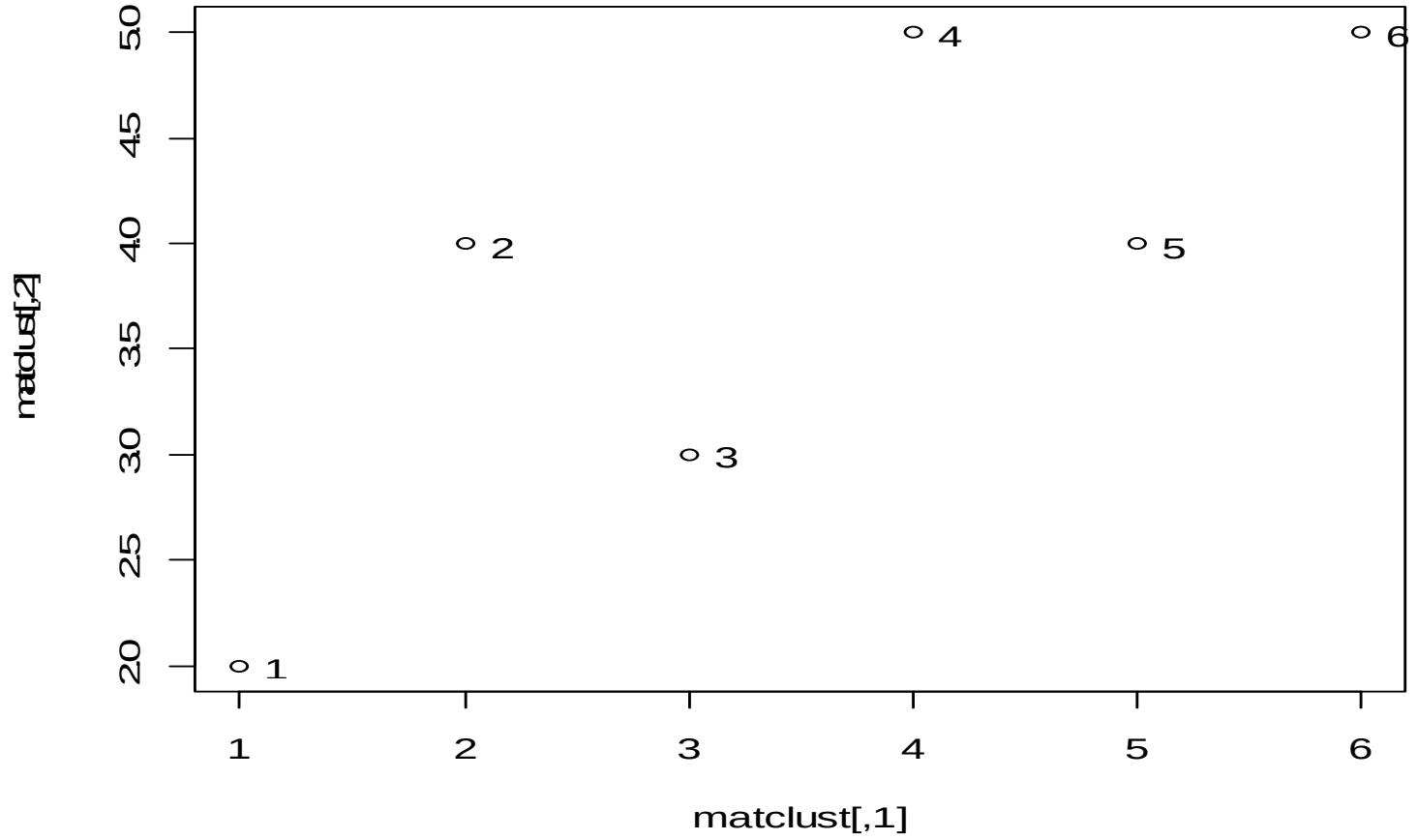
$$\partial(S,T)=\sum_{x\in S} d^2(x,\bar{x})+\sum_{y\in T} d^2(y,\bar{y})$$

Aqui se junta el par de grupos que produce la varianza mas pequena en el grupo juntado.

Distancias entre clusters- Ejemplo

```
> matclust
     [,1] [,2]
[1,]    1    2
[2,]    2    4
[3,]    3    3
[4,]    4    5
[5,]    5    4
[6,]    6    5
> daisy(matclust,metric="euclidean")
Dissimilarities :
         1        2        3        4        5
2 2.236068
3 2.236068 1.414214
4 4.242641 2.236068 2.236068
5 4.472136 3.000000 2.236068 1.414214
6 5.830952 4.123106 3.605551 2.000000 1.414214
Metric :  euclidean
Number of objects : 6
```

Distancias entre clusters- Ejemplo



plot(matclust); text(matclust[,1],matclust[,2],adj=-1, 1:6)

Distancias entre clusters- Ejemplo

Si consideramos que los puntos 1,2 y3 forman el lcuster 1 y que los puntos 4,5 y6 forman el cluster 2. Entonces,

el linkage simple dara2.236068

El linkage completo dara 5.830952

El linkage promedio dara 3.553621

El linkage Mediana sera 3.000

El centroide del cluster 1 es (2,3) y del cluster 2 es (5,4.67)

Asi que el linkage centroide sera 3.433

El Linkage de Mc Quitty sera (1/6)(3.41+ 2.77)=1.5742

El Linkage de Ward sera 4+2.67=6.67

Tipos de Algoritmos para Conglomerados ("Clustering")

I. Métodos de particionamiento
II. Metodos jerarquicos.
III. Metodos Basados en modelos
IV. Metodos basados en densidad
V. Metodos basados en grafos.

Tipos de Algoritmos para Conglomerados ("Clustering")

I. Métodos de particionamiento

El conjunto de datos es particionado en un número pre-especificado de conglomerados K, y 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).

Ejemplos: **K-means, PAM,** CLARA, **SOM**, Conglomerados basados en modelos de mezclas gausianas, Conglomerados difusos.

Tipos de Algoritmos para Conglomerados ("Clustering")

II. Metodos Jerárquicos.

En estos algoritmos se generan sucesiones ordenadas (jerarquias) de conglomerados. Puede ser juntando cluster pequenos en mas grande o dividiendo grandes clusters en otros mas pequenos. La estructura jerárquica es representada en forma de un árbol y es llamada **Dendograma.** Se dividen en dos tipos:

Algoritmos jerárquicos aglomerativos (bottom-up, inicialmente cada instancia es un cluster).AGNES

Algoritmos jerárquicos divisivos (top-down, inicialmente todas las instancias estan en un solo cluster. DIANA.

1-Algoritmo k-means (MacQueen, 1967).

El objetivo es minimizar la dissimiliridad de los elementos dentro de cada cluster y maximizar la disimilaridad de los elementos que caen en diferentes clusters.

INPUT: Un conjunto de datos S y k número de clusters a formar;

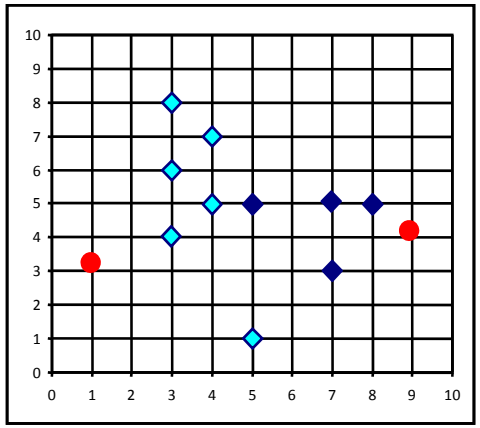
OUTPUT: L una lista de los clusters en que caen las observaciones de S.

1. Seleccionar los centroides iniciales de los K clusters: c_1, c_2, ..., c_K.
2. Asignar cada observación x_i de S al cluster C(i) cuyo centroide c(i) está mas cerca de x_i. Es decir, $C(i)=\text{argmin}_{1\le k\le K}\|\mathbf{x}_i-\mathbf{c}_k\|$
3. Para cada uno de los clusters se recalcula su centroide basado en los elementos que están contenidos en el cluster y minimizando la suma de cuadrados dentro del cluster. Es decir,

$$WSS=\sum_{k=1}^{K}\sum_{C(i)=k}\|\mathbf{x}_i-\mathbf{c}_k\|^2$$

Ir al paso 2 hasta que se consiga convergencia.

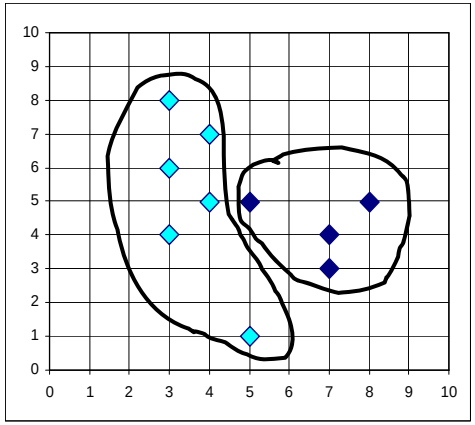
K-Means



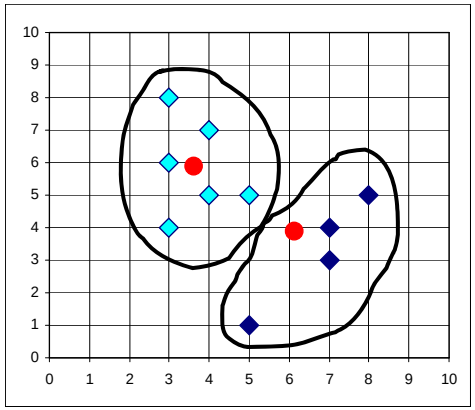
K=2

Escoger arbitrariamente K instances como los centroides

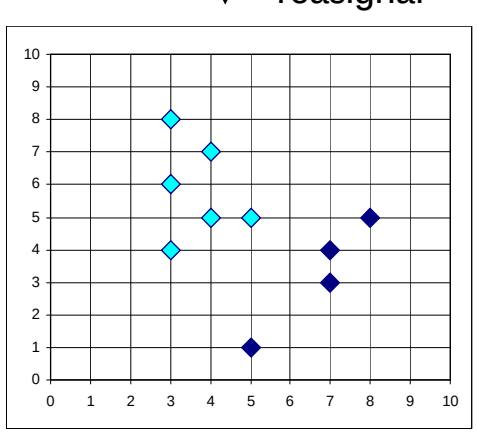
Asignar cada instancia al cluster mas cercano.



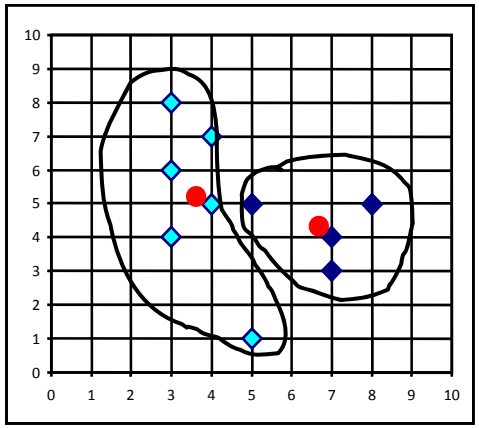
reasignar



Actualizar los centroides



reasignar



Actualizar los centroides

Alternativas para los k centroides iniciales

- Usando las primeras k observaciones
- Eligiendo aleatoriamente k observaciones.
- Tomando cualquier partición al azar en k clusters y calculando sus centroides.

Características del Algoritmo k-means

- No se satisface el criterio de optimizacion globalmente, solo produce un óptimo local.
- El algoritmo de k-means es computacionalmente rápida.
- Puede trabajar bien con datos faltantes (missing values).
- Es sensible a “outliers”.

R y la función **kmeans**

La librería **mva** de **R** tiene la función **kmeans** que ejecuta el algoritmo **kmeans**.

Ejemplo: Consideremos conjunto de datos **bupa (**6 variables predictoras y 345 observaciones).

```
>a=kmeans(bupa[,1:6],2)
>a$cluster
> # identificando los tamanos de los clusters formados
> table(a$cluster)
  1   2
308 37
> #  Los tamanos de los grupos verdaderos
>table(bupa[,7])
  1   2
145 200
 >·Comparando los grupos verdaderos con los clusters
table(a$cluster,bupa[,7])
     1   2
  1 131 177
  2  14  23
```

R y la función **kmeans**

El cluster 1 se parea con el grupo 2 y el cluster 2 con el grupo 1 para un total de 154 no coincidencias.

```
 > # Formando tres clusters
>kmeans(bupa[,1:6],3)
> b= kmeans(bupa[,1:6],3)
> table(b$cluster)
  1   2   3
 62 271  12
> #K-means eligiendo como centros iniciales las observaciones10 y 100 de Bupa
> medias<-bupa[c(10,100),1:6]
>am=kmeans(bupa[,1:6],medias)
> table(am$cluster)
  1   2
308  37
```

Kmeans para census

```
 > #Kmeans para censusn con datos imputados
 >a=kmeans(census.mimp[,-14],2)
>#  los tamanos de los clusters formados
>table(a$cluster)

    1     2
 8339 24222
> # los tamanos de los grupos verdaderos
>table(censusn[,14])

    1     2
24720  7841
> # Comparando los clusters con los grupos verdaderos
>table(a$cluster,censusn[,14])

       1     2
  1  6445  1894
  2 18275  5947
```

Particionamiento alrededor de medoides (PAM)

Introducido por Kauffman y Rousseauw, 1987.

MEDOIDES, son instancias representativas de los clusters que se quieren formar.

Para un pre-especificado número de clusters K, el procedimiento PAM está basado en la búsqueda de los K MEDOIDES,

$\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_K)$ de todas las observaciones a clasificar

Para encontrar M hay que minimizar la suma de las distancias de las observaciones a su mas cercano Medoide.

$$M^* = \arg\min_M \sum_i \min_k d(x_i, m_k)$$

d es una medida de dissimilaridad

PAM C(cont.)

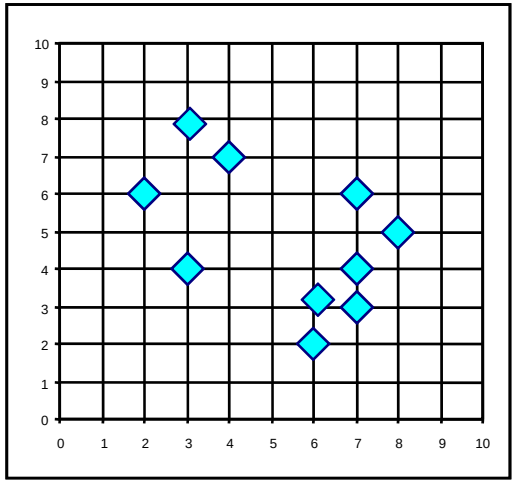
El algoritmo tienes dos fases:

En la primera fase llamada BUILT se eligen los k mediodes iniciales, siendo la observacion, cuya suma de disimilarides a los otras observaciones es la mas pequena de todas, el primer medoide elegido. Los restantes mediodes son elegidos de tal manera que la ganacia total sea la mas alta.

Sin embargo esta fase puede ser omitida , eligiendo los medoides iniciales al azar.

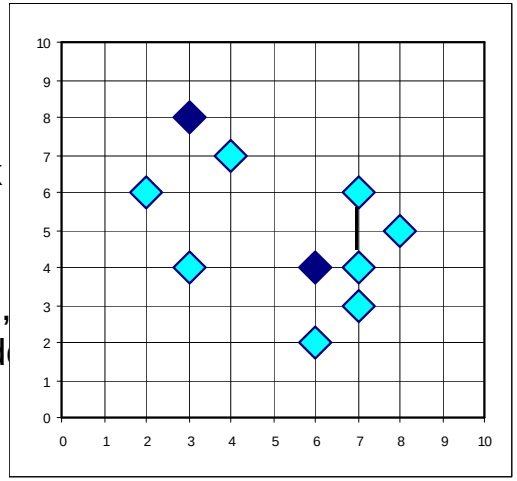
En la segunda fase llamada SWAP , se trata mejorar le rendimiento del algoritmo de clustering intercambiando los medoides iniciales por otras observaciones que no han sido elegidas aun.

PAM



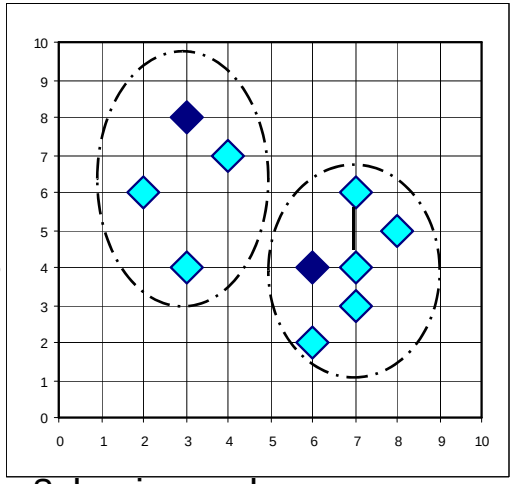
K=2

Escoger k instances como medoides, de acuerd al paso BUILT

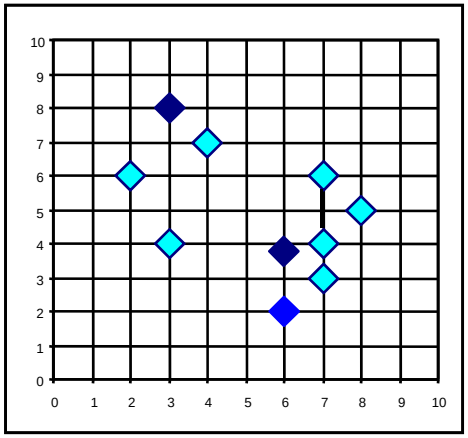


Asignar las instancias restantes a su mas cercano medoide.

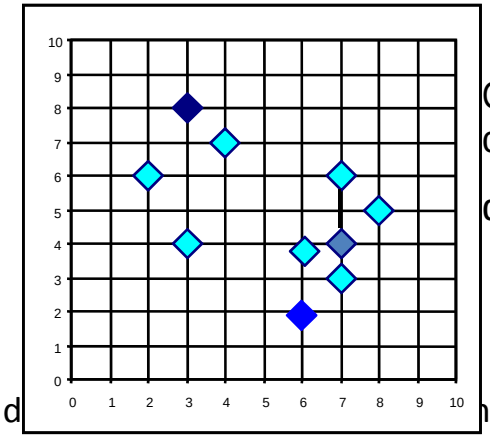
Total Cost = 20



Seleccionar al azar una instancia nonmedoide ,O_{ramdom}



Calcular el costo total de swapping



Swapping O y O_{ramdom}

Si se mejora la calidad

Hacer el loop hasta que no haya cambios

R y la función PAM

La función pam de la librería cluster encuentra los conglomerados usando el particionamiento alrededor de medoides.

Aplicaremos PAM al conjunto de datos bupa.

```
>pambupa=pam(bupa[,1:6],2,diss=F)
>pambupa$clustering
> table(pambupa$clustering)
  1   2
294  51
> table(pambupa$clustering,bupa[,7])
     1   2
  1 128 166
  2  17  34
```

R y la función PAM

```
>pambupa=pam(bupa[,1:6],2,diss=F,stand=T)
>table(pambupa$clustering)
1   2
292  53
>table(pambupa$clustering,bupa[,7])
     1   2
  1 123 169
  2  22  31
>a=pam(daisy(bupa[,-7], metric = "euclidean"), 2, diss = T)
> table(a$clustering)
 1   2
294  51
```

Generalmente los resultados son un poco mejores que el de k- means, pero es mas lento que kmeans

Mapas auto-organizados (Self-orgamizing Maps, SOM)(Kohonen, 1988)

Son algoritmos de particionamiento que son restringidos al hecho que los clusters pueden ser representados en una estructura geometrica de dimensión baja, tal como un grid (los más usados son los SOM en una y dos dimensiones). El algoritmo SOM es bastante similar a k-means

Los clusters que son cercanos entre sí aparecen en celdas adyacentes del grid. O sea, SOM mapea el espacio de entrada de las muestras en un espacio de menor dimension en el cual la medida de similaridad entre las muestras es medida por la relación de cercanía de los vecinos.

Cada uno de los K clusters es representado por un objeto prototipo Mi , i = 1,…K .

Mapas auto-organizados
(Self-orgamizing Maps, SOM)

En este método la posición de una observación en el espacio inicial de entradas influye de alguna manera en su asignación a un conglomerado.

Se construye un plot en donde las observaciones similares son ploteadas cercanas entre sí.

Un SOM es entrenado como una red neural.
El SOM puede ser considerado también como una técnica de reducción de dimensionalidad y guarda relación con la técnica conocida como escalamiento multidimensional (proceso de encontrar un conjunto de puntos en un espacio multidimensional que corresponda a medidas de similaridad entre objetos).

El algoritmo SOM considerando un grid rectangular de dos dimensiones

Paso1-Seleccionar los numeros de filas (q_1) y columnas (q_2) en el grid. Luego habrá $K=q_1q_2$ clusters

Paso2. Inicializar el tamaño del parámetro de actualizacion (la tasa de aprendizaje en terminos de redes neurales) α (α=1) y el radio del grid (r=2)

Paso3. Inicializar los vectores prototipos M_j, j ε (1,…,q1) x (1,….,q2) mediante la eleccion aleatoria de K instancias.

Paso4. Para cada observacion $x_i$del conjunto de datos hacer los siguiente:

Identificar el vector índice j' del prototipo Mj mas cercano a x_i.

Identificar un conjunto S de prototipos vecinos de $M_{j'}$. Es decir,

$S=\{j: \text{distancia } (j, j')<r\}$

La distancia puede ser Euclideana o cualquiera otra

Actualizar cada elemento de S moviendo el correspondiente prototipo hacia x_i:

$M_j \leftarrow M_j + \alpha(x_i - M_j)$ para todo j ε S

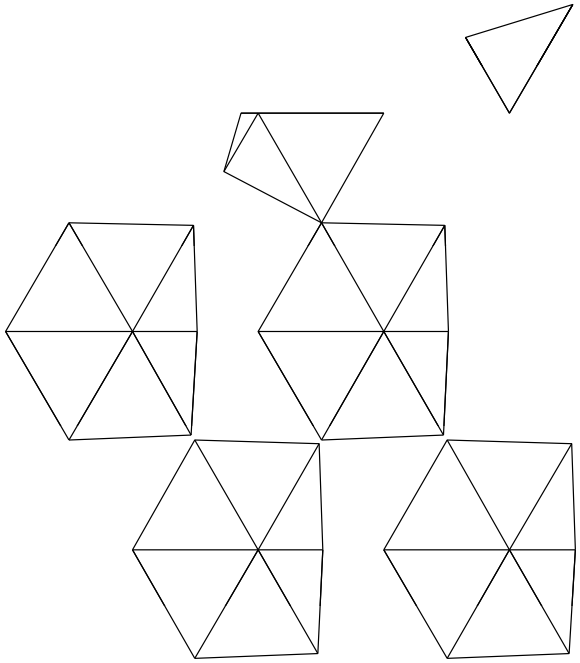
Paso5. Disminuir el tamaño de α y r en una cantidad predeterminada y continuar hasta alcanzar convergencia.

R y la función SOM

La función **SOM** de la librería **class** de B. Ripley construye un SOM, al igual que la libreria **som. Aunque los resultados son solamente visuales. De cada centroide salen lineas que representan cada una de las variables**

Por ejemplo para el conjunto de datos bupa

```
>bupgr=somgrid(2,3,topo="hexa")
> sombupa=SOM(bupa[,1:6],bupgr)
>plot(sombupa)
```



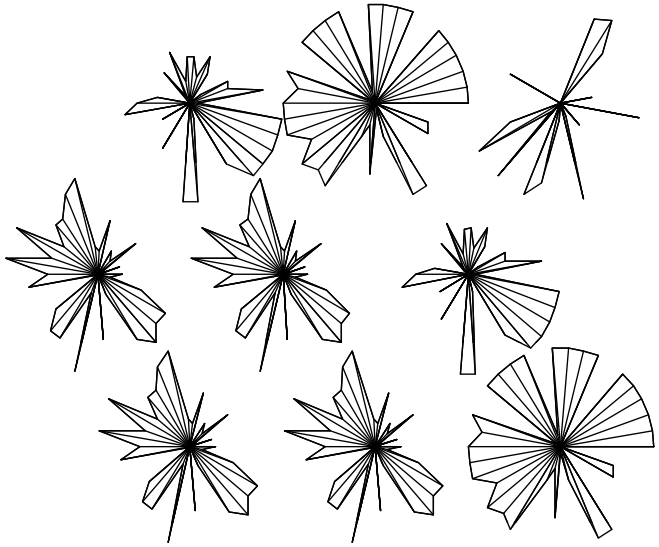
R y la función SOM

Por ejemplo para el conjunto de datos golub

```
golubgr=somgrid(3,3,topo="hexa")
somgolub=SOM(golub,golubgr)
 plot(somgolub)
```

Otros software: Somtoolbox en Matlab, Cluster por Eissen y Genecluster (MIT Center por Genome Research). Este ultimo muestra los elementos de cada cluster.

Lamentablemente, al libreria SOM, ha sido dicontinuada. Aquí usaremos solo som.



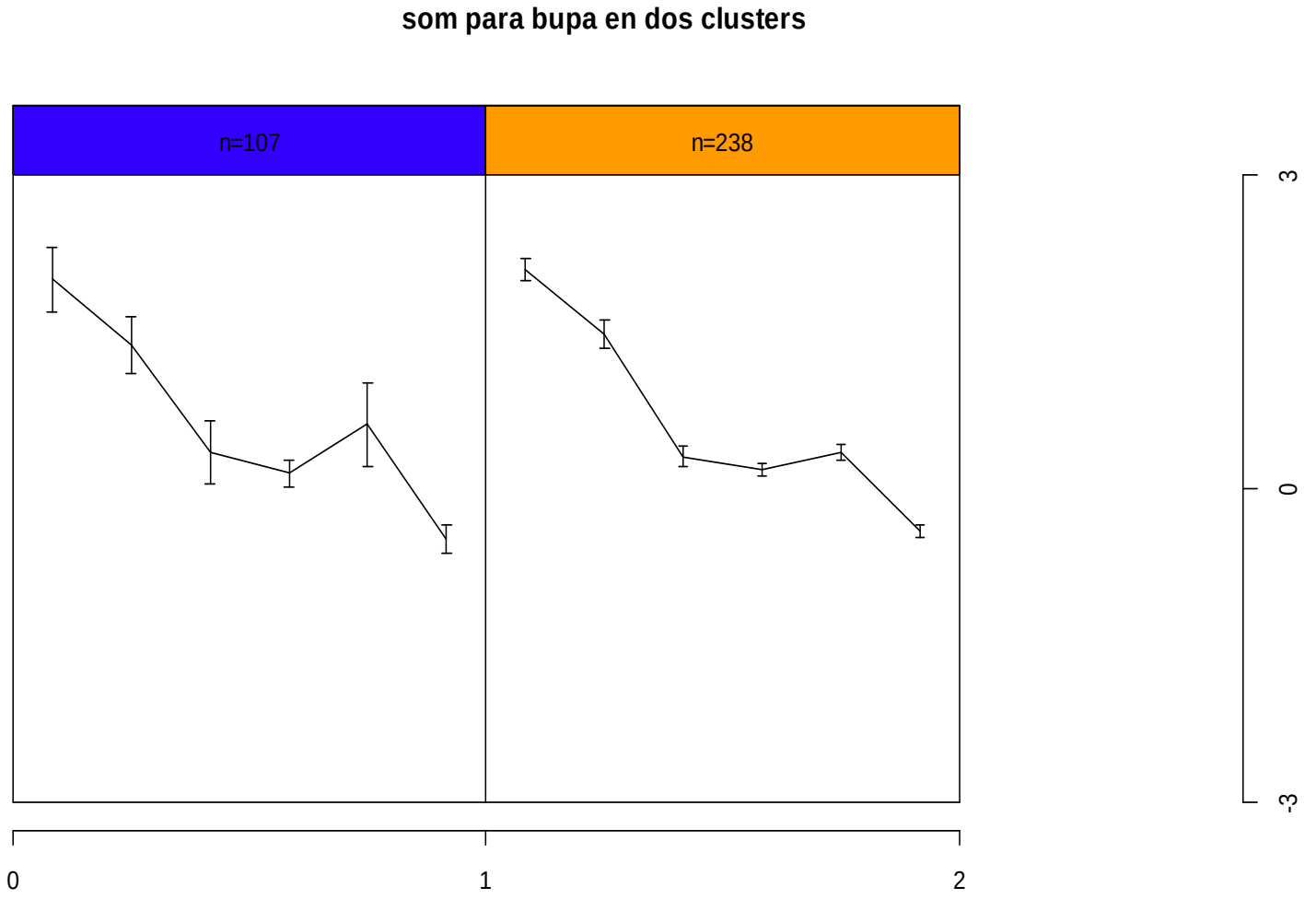
SOM usando la libreria som

```
library(som)
#Haciendo el som para bupa
# normalizando los datos
bupa.n=normalize(bupa[,-7])
# Haciendo solo dos clusters
a=som(bupa.n,xdim=2,ydim=1,topol="hexa",neigh="gaussian")
#ploteando el som
plot(a)
#listando las asignaciones de los objetos a los clusters
a$visual
```

	x	y	qerror
1	1	0	0.7166912
2	1	0	1.0007942
3	0	0	0.6676584
3	.....		

```
#el numero de instances asignadas al cluster (0,0)
dim(a$visual[(a$visual[,1]==0) & (a$visual[,2]==0),])[1]
107
```

SOM usando la libreria som

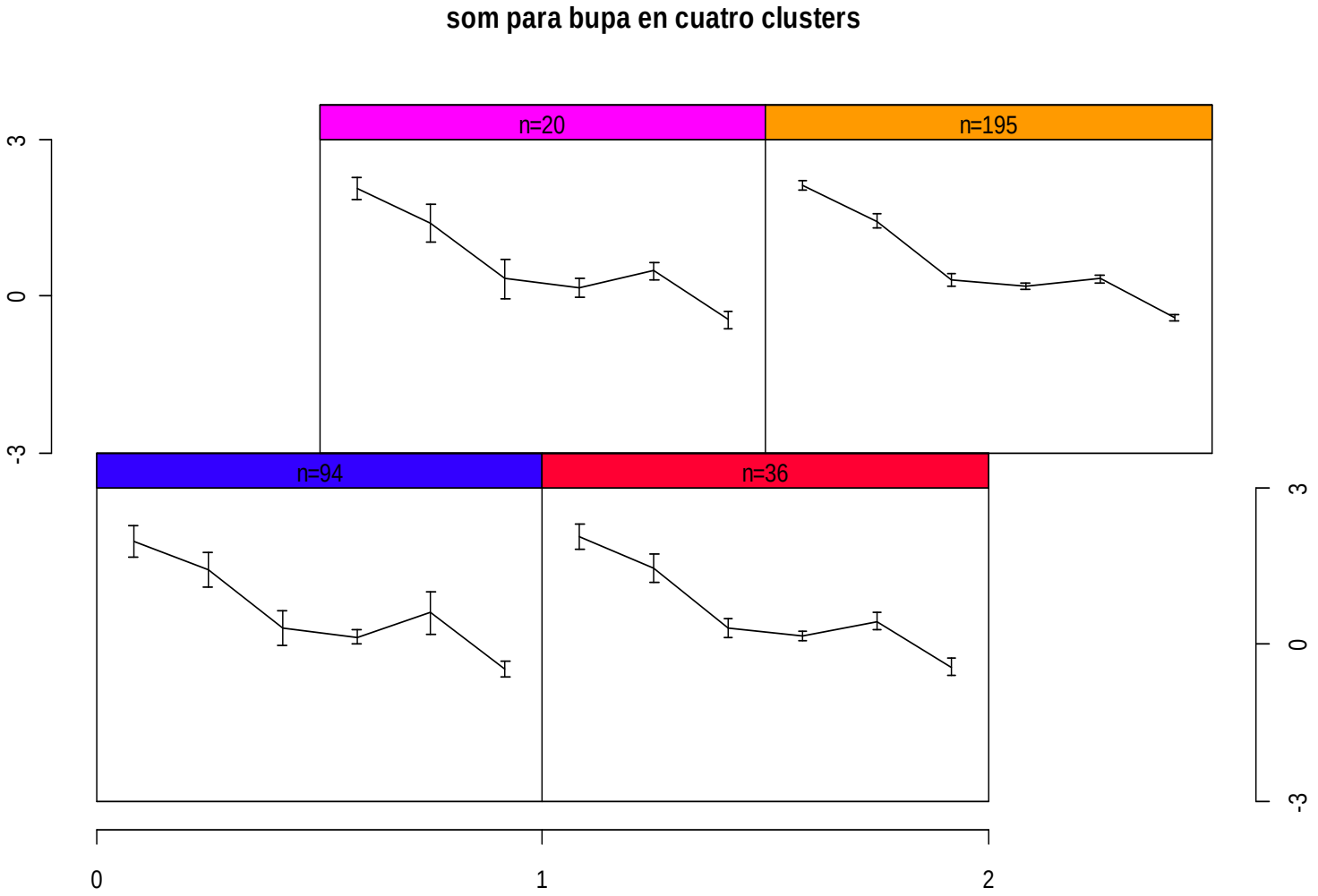


SOM usando la libreria som

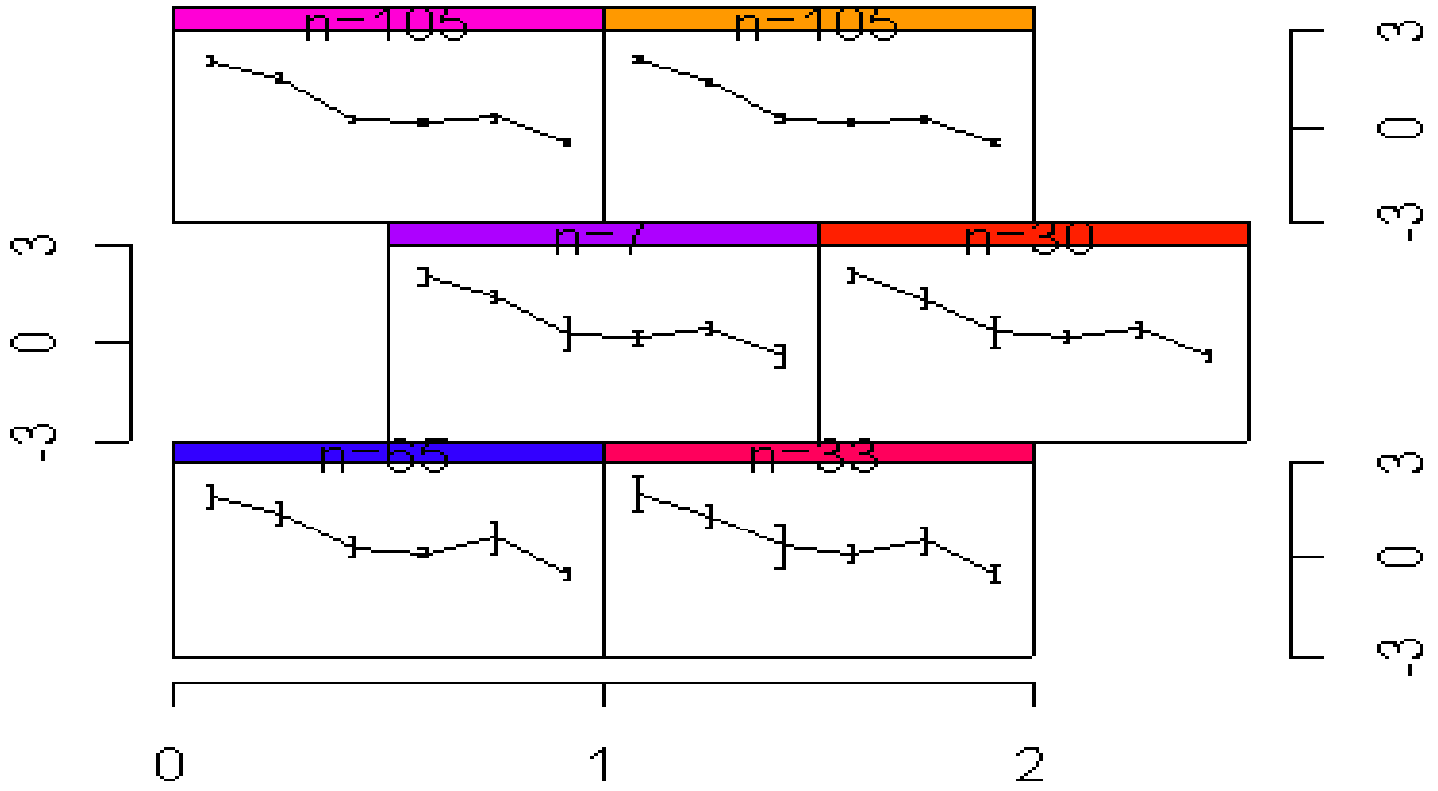
```
>rownames(dim(a$visual[(a$visual[,1]==0) & (a$visual[,2]==0),])
>clusters=a$visual$x+a$visual$y
> clusters
 [1] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1
     ……
> table(clusters,bupa[,7])

clusters   1   2
       0  31  76
       1 114 124
```

El cluster 0 se parea con el grupo 2 y el cluster 1 se parea con el grupo 1, dando en total 155 no coindicencias.

SOM usando la libreria som



SOM usando la libreria SOM



plot(sombupa,sd=1)