Segmentación de mercados: proceso

Jordi López Sintas

9 de enero de 2015

Modelos de segmentación: proceso

1. El procedimiento estándar

Fase exploratoria: la clasificación jerárquica

La clasificación jerárquica con el procedimiento de Ward minimiza la suma de las distancias euclidianas al cuadrado entre el individuo i y el centro del segmento al que se asigna,$\sum_k \sum_{i \in k} \sum_j = (x(i,j) - x(j',j))^2$. El primer sumatorio calcula el error de la observación i en el segmento k para todas las variables, el segundo realiza el cálculo para todos los individuos en el segmento k y, finalmente, el último sumatorio calcula la suma del error para todos los segmentos (Wishart 1998).

Funciona de la siguiente manera

1. Computa la matriz de las distancias euclidianas al cuadrado para todo par de observaciones, *i*, *i'*, $d^2(i,i')$=$\sum_k (x(i,j) - x(j',j))^2$.
2. Agrupa las dos observaciones o grupos i y i' más cercanos, es decir, cuya agrupación minimiza el incremento en el error, *E*. Inicialmente serán aquellos casos cuya distancia, $d^2(i,i')$, sea mínima.
3. Transforma la matriz inicial de distancias euclidianas al cuadrado D^2 en E^2, la cual contendrá el error al cuadrado de la unión del nuevo segmento $i \cup i$.
4. Repite los pasos 2 y 3 y cada vez forma un nuevo grupo con las observaciones o grupos cuya unión resulte en un incremento mínimo en el error E^2.
5. Finaliza cuando todos los casos se hayan agrupado en un solo segmento

El procedimiento minimiza el cuadrado de la suma de las desviaciones entre los individuos y el centro del grupo al que se le ha asignado. Realiza un proceso de aglomeración con tantas fases o etapas como individuos ha de clasificar, n, de manera que una vez clasificados en su totalidad se minimiza una medida de la heterogeneidad, $min \sum_{SQD-I}$. En la etapa inicial existen tantos grupos como individuos debe agrupar. A partir de ese momento en cada etapa se formará un nuevo segmento agrupando a dos de los segmentos ya formados en etapas anteriores o un grupo y un individuo aun no clasificado, aquéllos más parecidos entre sí, de forma que se minimice en cada agrupación el incremento en la suma de las diferencias entre el individuo agrupado y la media del grupo al que se asigna, $min \left\{ \sum_i x^2(i,j) - \frac{\sum_j x^2(i,j)}{n} \right\}$. En la etapa inicial la suma del cuadrado de las distancias es cero y se va incrementando a medida que se realizan las agrupaciones.

Suponogamos que tenemos cinco individuos medidos en una única variable $\{A = 2, B = 5, C = 9, D = 10, E = 15\}$.

d(A, B)=22+52-(1/2)(2+5)2=29-24,5=4,5

	A	B	C	D	E
A	0	4,5	24,5	32	98
B		0	8	12,5	60,5
C			0	**0,5** 24	,5
D				0	18
E					0

La agrupación C y D es la que minimiza el SQD.

	A	B	CD	E
A	0	**4,5** 38	98	
B		0	14	60,5
CD			0	28,66
E				0

La agrupación A y B es la que minimiza la SQD

	AB	CD	E
AB	0	41	108,66
CD		0	**28,66**
E			0

La agrupación *CDE* es la que minimiza la *SQD*. Finalmente $d(AB,\ CDE)$= 113,2.

La agrupación se muestra visualmente con un gráfico denominado dendrograma, donde las letras identifican a los individuos.

```
ejemplo<-c(2,5,9,10,15)
ejemplo
```

```
## [1]  2  5  9 10 15
```

```
ejemplo.dist<-dist(ejemplo)
ejemplo.dist
```

```
##    1  2  3  4
## 2  3
## 3  7  4
## 4  8  5  1
## 5 13 10  6  5
```

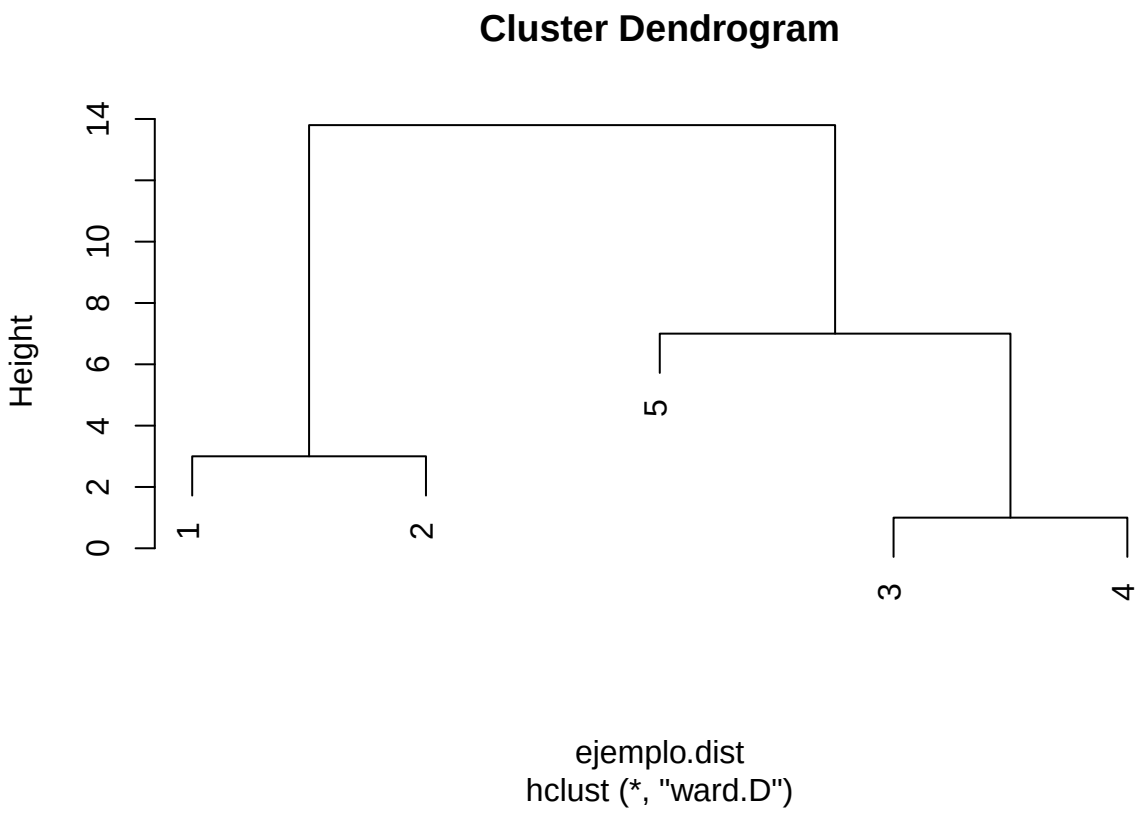
```
ejemplo.hclust<-hclust(ejemplo.dist, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

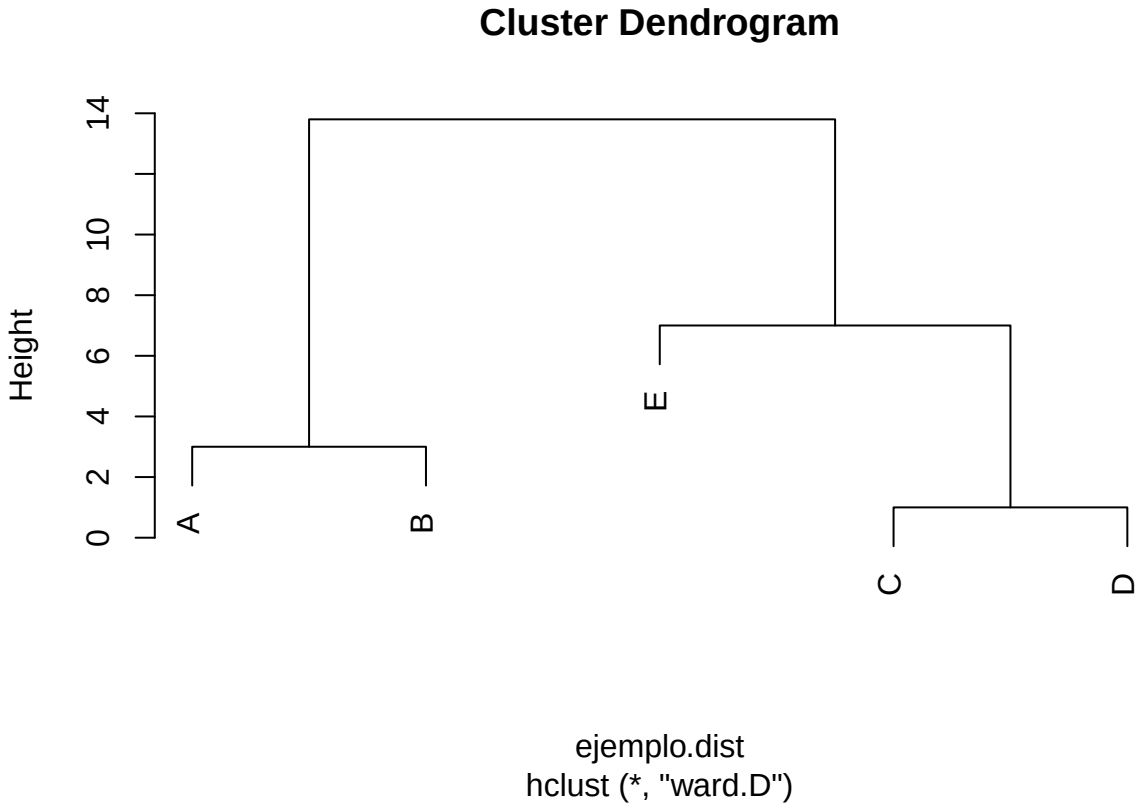
```
ejemplo.hclust
```

```
##
## Call:
## hclust(d = ejemplo.dist, method = "ward")
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 5
```

```r
plot(ejemplo.hclust)
```




```r
labels<-c("A", "B", "C", "D", "E")
ejemplo.hclust$labels<-labels
plot(ejemplo.hclust)
```



Veamos cómo lo haríamos en el entorno R de análisis de datos.

Partición de la muestra en k segmentos: el procedimiento k-means

Los algoritmos de clasificación del tipo k-means inician el proceso de selección especificando el número de clases que desea el analista, k, así como los centros iniciales para cada uno de los k clusters a formar. Después asigna cada individuo al grupo cuyo centro tenga más próximo. La solución inicial se mejora de manera iterativa hasta que se alcanza alguna forma de estabilidad, por ejemplo, hasta que las reasignaciones no reducen la suma del cuadrado de las distancias de los consumidores a los centros de los segmentos a los que han sido asignados (Helsen y Green 1991). Concretamente:

1. Empieza el proceso con un conjunto de centros iniciales o coordenadas de los k segmentos. Estos centros pueden obtenerse de manera aleatoria o por algún procedimiento como el propuesto por Hartigan y Wong (1979).
2. Después asignamos al consumidor i al cluster cuyo centro está más próximo. Los centros permanecen inalterados durante todo el proceso de asignación.
3. Seguidamente calculamos un nuevo conjunto de centros de los segmentos, como la media de los consumidores asignados a cada uno. Estos nuevos centros serán la base de un nuevo ciclo de reasignaciones.
4. Repetimos los procesos 2 y 3 hasta que ningún individuo cambie de grupo en el proceso de asignación especificado en el paso 2.

En este algoritmo el centro de cada segmento se define como la media de todos los individuos asignados al segmento en cada una de las variables en las que se han medido sus propiedades. Por ello necesita una matriz de datos nxp en lugar de una matriz de distancias entre los individuos (a diferencia de lo que ocurre en los

algoritmos de clasificación jerárquica). El propósito del algoritmo es minimizar la suma de las distancias euclidianas al cuadrado de la partición realizada en k segmentos. Implícitamente asume que los grupos muestran una distribución normal esférica. Vemos un ejemplo.

n= número de individuos a clasificar

p= número de variables en las que se han medido las propiedades de los individuos

$x(i,j)$= valor que muestra el individuo i en la variable j

$x(l,j)$= valor medio de la variable j en el segmento l.

$n(l)$=número de individuos en el grupo l.

$d(i,l)$= distancia entre el individuo i y el centro del segmento l.

$d(i,l)$=raíz cuadra de la suma para toda p de las diferencias al cuadrado

$e(p(n,\ k))$=error de la partición

$p(n,\ k)$= resultado de partir la muestra en k segmentos y asignar a los n individuos a cada uno de los k segmentos.

$min\ e(p(n,\ k))=\sum_{i=1,\ldots,n} d^2(i,l(i))=min\ SQD_I$

Tipo de pescado	Energía	Calorías	Calcio	Sum(i)
Caballa	5	9	20	34
Perca	6	11	2	19
Salmón	4	5	20	29
Sardina	6	9	46	61
Atún	5	7	1	13
Camarones	3	1	12	16

Supongamos que queremos formar tres segmentos. El procedimiento nos daría como resultado los siguientes segmentos:

Segmento 1: perca, atún y caballa

Segmento 2: caballa y salmón

Segmento 3: sardina

Seguidamente calcularemos la media de las propiedades de los objetos clasificados en cada uno de los segmentos.

Segmento	Energía	Calorías	Calcio
1	14/3	19/3	5
2	9/2	7	20
3	6	9	46

Y calculamos la distancia euclidiana entre los individuos y las medias de cada grupo:

E(p(n=6,k=3))=SQDI=d^2ˆ(1,1)+d^2(2,1)+...+d^2(6,1)+...+d^2(1,3)+...+d^2(6,3)=137,805

Seguidamente probamos si cualquier cambio en la asignación de individuos a los segmentos reduce el error de la clasificación o suma del cuadrado de las distancias entre individuos y centros. Siendo $n(l)$ el número de individuos asignados al segmento l, y $l(i)$ el segmento que contiene al individuo i, primero calculamos las distancias al cuadrado entre el primer individuos y los centros de cada uno de los grupos:

$d^2(1,1)=(5-14/3)^2+(9-19/3)^2+(20-5)^2=232{,}22$

$d^2(1,2)=4,25$

$d^2(1,2)=677$

A la hora de decidir donde clasificar al individuo i, calcularemos la variación en la SQD_I. En este caso, cambiar la caballa de segmento incrementaría el error de la partición realizada.

Realizamos el proceso para todos los objetos y encontramos, en este caso, que el objeto 6, el camarón, puede ser clasificado en el segmento 2, en lugar del *1* inicial, y reducir el error de la clasificación. Quedando así la clasificación siguiente:

Segmento 1: perca y atún

Segmento 2: caballa, salmón y camarón

Segmento 3: sardinas.

Código (modificar)

Energia Calorias Calcio

Caballa 5 9 20

Perca 6 11 2

Salmon 4 5 20

Sardina 6 9 46

Atun 5 7 1

Camarones 3 1 12

```
peces<-read.csv("peces.csv", row.names=1, header=T)
peces
```

```
##           Energia Calorias Calcio
## Caballa         5        9     20
## Perca           6       11      2
## Salmon          4        5     20
## Sardina         6        9     46
## Atun            5        7      1
## Camarones       3        1     12
```

```
peces.dist<-dist(peces)

peces.dist
```

```
##             Caballa     Perca    Salmon   Sardina      Atun
## Perca     18.138357
## Salmon     4.123106 19.078784
## Sardina   26.019224 44.045431 26.381812
## Atun      19.104973  4.242641 19.131126 45.055521
## Camarones 11.489125 14.456832  9.000000 35.057096 12.688578
```

Caballa Perca Salmon Sardina Atun

Perca 18.138357

Salmon 4.123106 19.078784

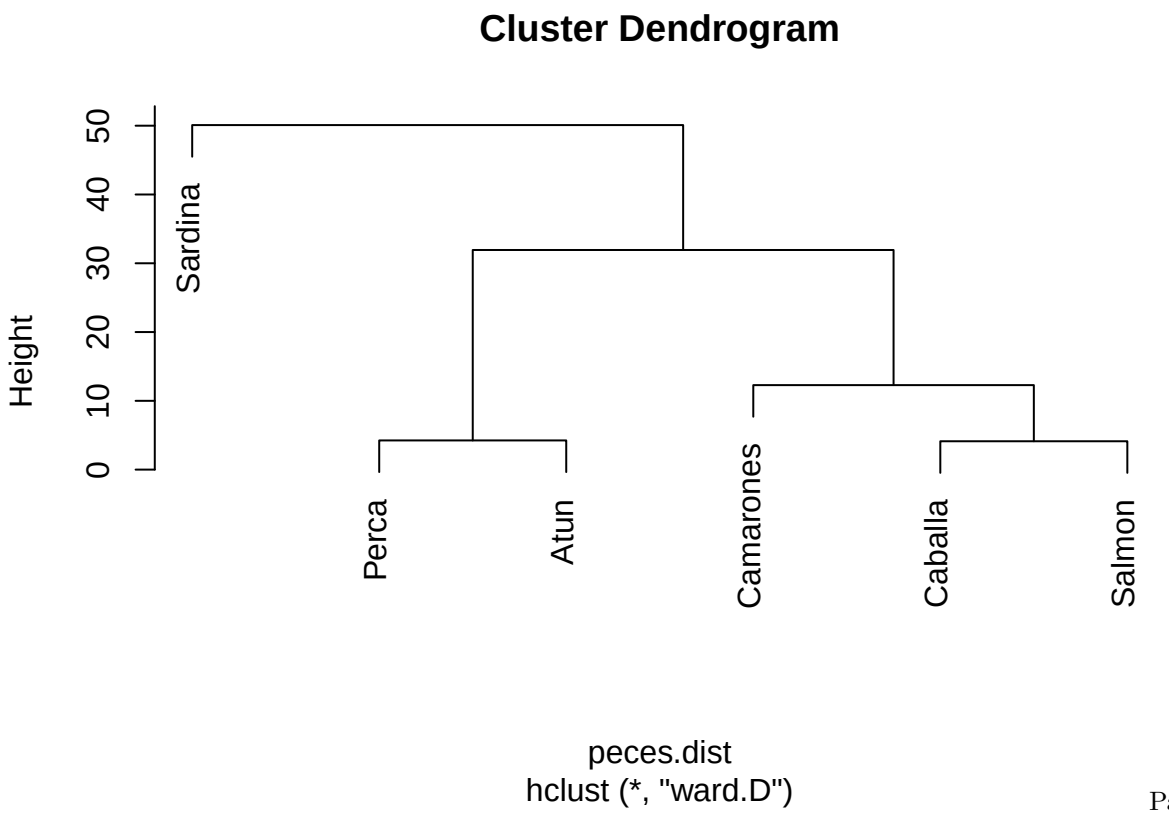
Sardina 26.019224 44.045431 26.381812

Atun 19.104973 4.242641 19.131126 45.055521

Camarones 11.489125 14.456832 9.000000 35.057096 12.688578

You can also embed plots, for example:

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```



Partimos la muestra en tres segmentos

```
## KMNS(*, k=3): iter=  1, indx=0
##  QTRAN(): istep=6, icoun=1
## KMNS(*, k=3): iter=  2, indx=3
## KMNS(*, k=3): iter=  3, indx=6
```

Si queremos saber las opciones de lal función k-means, podemos utilizar la función ? seguida del nombre de la función

```
?kmeans
```

Para saber los objetos incluidos en el resultado de clasificar la muestra, podemos uitlizar la función names()

```
names(peces.kmeans)
```

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

Los centros de los segmentos se encuentran en el objeto `centers`. La classificación se encuentra en el objeto `cluster`, y en el objeto `iter`el número de iteraciones realizadas.

```
peces.kmeans$centers
```

```
##   Energia Calorias   Calcio
## 1     5.5        9  1.50000
## 2     6.0        9 46.00000
## 3     4.0        5 17.33333
```

```
peces.kmeans$cluster
```

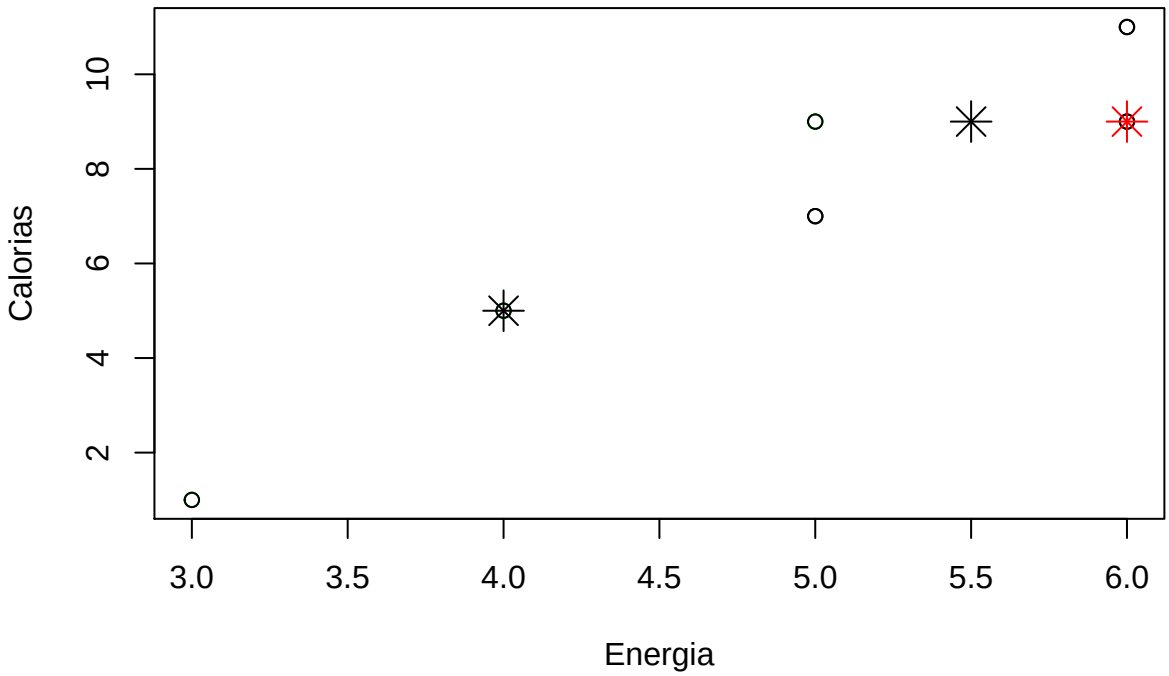
```
##   Caballa     Perca    Salmon   Sardina      Atun Camarones
##         3         1         3         2         1         3
```

```
peces.kmeans$iter
```

```
## [1] 3
```

El resultado lo podemos visualizar en el espacio de las bases de segmentación

```
plot(peces[1:2], col=peces.kmeans$cluster)
points(peces[1:2])
points(peces.kmeans$centers, col=1:2, pch=8,cex=2)
```



Procedimientos más robustos

PAM

El algoritmo pam tiene por objetivo encontrar los k consumidores representativos de la heterogeneidad del mercado alrededor de los cuales se clasifican los n consumidores medidos en p variables (una matriz de datos de tamaño nxp). Estos k consumidores representativos formarán los k segmentos en los que se dividirá una muestra de un mercado. Los k consumidores representativos se obtienen de minimizar la disimilitud entre todos los consumidores y su consumidor representativo más cercano. Es decir, el procedimiento busca encontrar conjunto de k segmentos de individuos pertenecientes a la muestra que minimiza la siguiente función objetivo: peces

$$\sum_{i=1}^{n} \min_{\forall k} d\left(i, m_k\right)$$

De tal manera que cada consumidor se asigna al segmento al que pertenece el consumidor representativo más cercano. Es decir, el consumidor i se asigna al segmento k cuando el consumidor representativo *m(k)* está más cerca de i que de cualquier otro consumidor representativo, $d\left(i, m_k\right) \leq d\left(i, m_w\right)$, para todo consumidor representativo w diferente de k.

Dado que el algoritmo pam sólo depende de las disimilitudes o distancias entre consumidores, la función sólo precisa como argumento una matriz de disimilitudes. Si el argumento que damos a la función es una matriz de datos nxp, entonces la función pam la convertirá en una matriz de disimilitudes.

El algoritmo pam es parecido al conocido método *k-means* (MacQueen 1967) ampliamente conocido en marketing (REF), implementado en el entorno R por medio de la función `kmeans` que utiliza el algoritmo de Hartigan y Wong (1979). El algoritmo k-means tiene por objetivo minimizar la suma del cuadrado de las distancias euclidianas e implícitamente asume que los grupos se distribuyen de forma esférica. La función `pam`, según Kaufman y Rouseeuw (1990), es más robusta que *k-means* porque el algoritmo implementado en la función pam minimiza la suma de las distancias sin elevarlas al cuadrado. Incluso los centros provisinales de los segmentos no son necesarios para iniciar el proceso de división de la muestra.

Adicionalmente la función pam del paquete `cluster` nos ofrece una gráfica denominada silueta (Rouseeuw 1987) y el correspondiente índice de calidad del proceso de segmentación con el objeto de ayudarnos a seleccionar el número de grupos que formaremos. La construcción del gráfico procede de la siguiente manera. Para cada consumidor i simbolizaremos con A el grupo de pertenencia y computaremos la disimilitud media del consumidor i frente al resto de consumidores del segmento A.

$$a\left(i\right) = \frac{1}{|A| - 1} \sum_{j \in A} d\left(i, j\right)$$

Después consideraremos cualquier grupo C diferente del anterior, A, y calcularemos la similitud media de i frente a todos los elementos de C, $d\left(i, C\right) = \frac{1}{|C|} \sum_{j \in c} d\left(i, j\right)$. Después realizaremos el mismo cálculo para el resto de los segmentos formados y tomaremos el valor menor,$b\left(i\right) = d\left(i, C\right)$. Llamaremos B al segmento cuya disimilitud media sea menor frente al consumidor i, d(i,B)=b(i), que llamaremos el segmento vecino del consumidor i. Esta es la segunda mejor la clasificación del consumidor i. La longitud de la silueta se calculará de la siguiente manera:

$s\left(i\right) = \frac{b(i) - a(i)}{\max\{b(i), a(i)\}}$. El valor de $s(i)$ está acotado entre -1 y 1. Si el valor está cerca de 1, nos indica que el consumidor i está bien clasificado; si está cerca de 0, está mal clasificado (está más cerca de B que de A).

Para finalizar calcula un índice de calidad global que es la media de todos los índices de calidad de la clasificación de todos los consumidores, $s(i)$, correspondientes a los n consumidores de la muestra de datos.

Índice de calidad	Interpretación
0,71-1,00	Hemos encontrado una estructura fuerte
0,51-0,70	Hemos encontrado una estructura razonable
0,26-0,50	Hemos encontrado una estructura débil que podría ser artificial
$\leq$ 0,25	No hemos encontrado una estructura en los datos

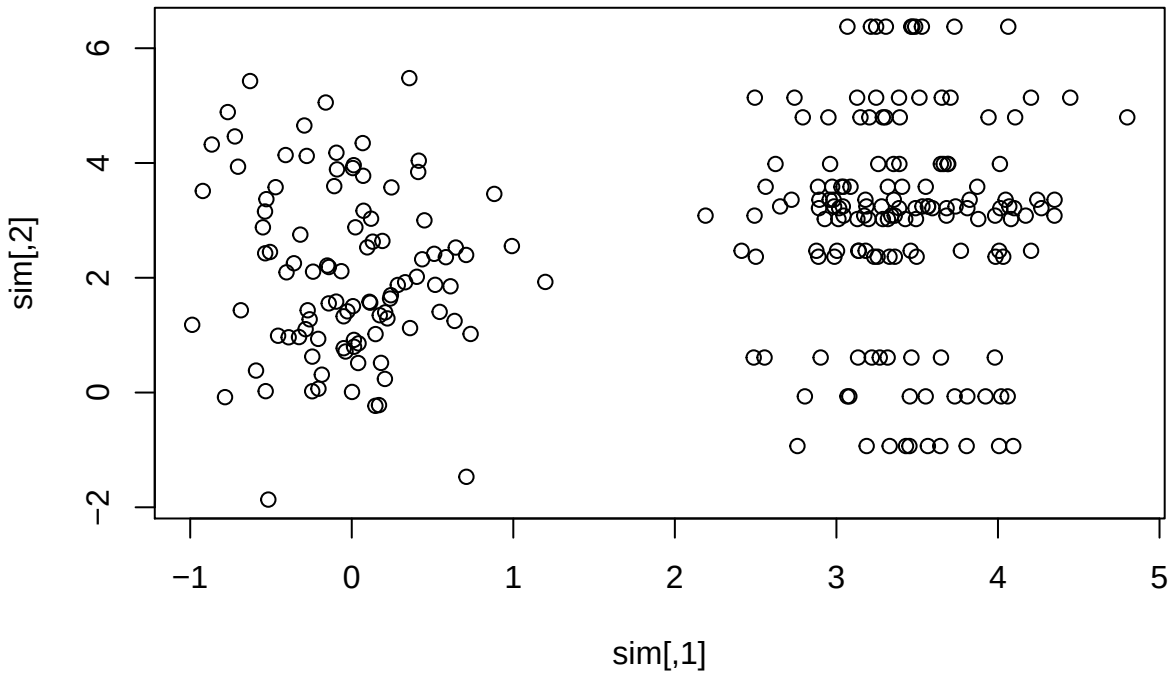
CLARA

La función `pam` almacena en la memoria del ordenador toda la matriz de disimilitudes cuyo tamaño es $O(n^2)$. Este procedimiento es ineficiente con matrices datos que contienen más de 250 consumidores, pues tanto la memoria ocupada como el tiempo de cálculo crecen de manera cuadrática en función del tamaño de la muestra, n^2). Para resolver este problema Kaufman y Rouseeuw (1986) han propuesto la función clara. Esta función sólo guarda en la memoria del ordenador un subconjunto de la matriz de disimilitudes. El procedimiento de la función clara toma muestras de la matriz original de datos, para cada una de ellas realiza una segmentación y obtiene k segmentos que minimizan la función objetivo de `pam`. Después compara la suma de las distancias y se queda con la partición que la minimiza. Finalmente asigna los n consumidores a la partición anterior. De esta manera el almacenamiento en memoria y el tiempo de computación crece sólo de manera lineal con el tamaño de la muestra, n.

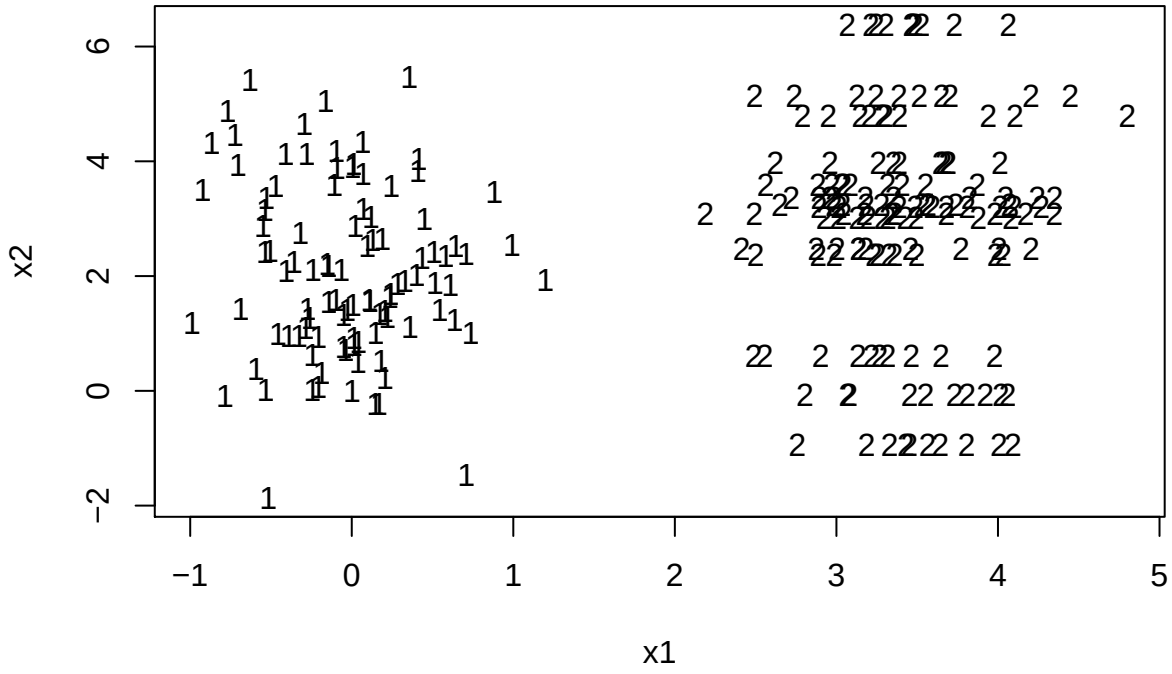
```
sim<-rbind(cbind(rnorm(100,0,0.5),
rnorm(100,2,1.5)),cbind(rnorm(150,3.5,0.5), rnorm(15,4,2.5)))
head(sim)
```

```
##              [,1]      [,2]
## [1,] -0.096588947 1.5844287
## [2,] -0.536965239 3.1548497
## [3,] -0.285569224 1.1039914
## [4,]  0.205183337 0.2366073
## [5,]  0.006690843 3.9117365
## [6,] -0.260604902 1.2759361
```

```
plot(sim)
```




```
library(MASS)
sim.kmeans<-kmeans(sim, 2)
#eqscplot(sim, type="n", xlab="x1", ylab="x2")
plot(sim, type="n", xlab="x1", ylab="x2")
text(sim, labels=sim.kmeans$cluster)
```



repetimos el proceso para 3 y 4 segmentos

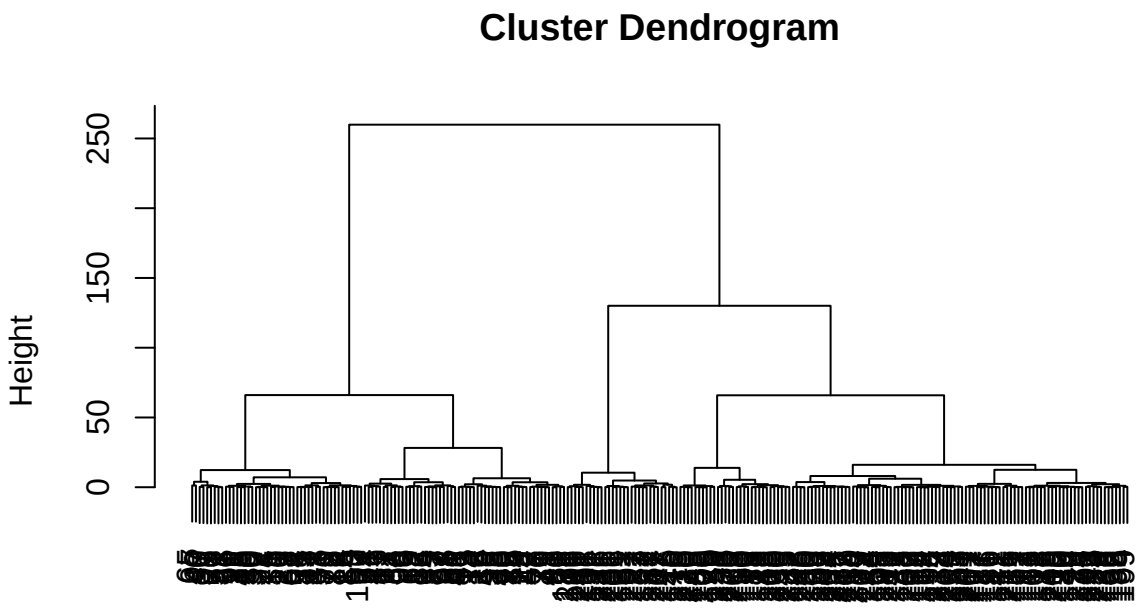
```
sim.kmeans3<-kmeans(sim, 3)
sim.kmeans4<-kmeans(sim, 4)
```

Ahora calcularemos los centros iniciales para ver si mejoramos la clasificación

```
sim.hclust<-hclust(dist(sim), method = "ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
plot(sim.hclust)
```




```
centros.h<-tapply(sim, list(rep(cutree(sim.hclust, 2),ncol(sim)), col(sim)),mean)
centros.h
```

```
##             1        2
## 1 -0.02418191 2.089737
## 2  3.40692926 2.949795
```

```
sim.kmeans.centros<-kmeans(sim, centros.h)
```

Repetimos la partición pero para 3 y 4 segmentos

```
centros.h3<-tapply(sim, list(rep(cutree(sim.hclust, 3), ncol(sim)), col(sim)), mean)
centros.h3
```

```
##             1         2
## 1 -0.02418191  2.089737
## 2  3.42522992 -0.129836
## 3  3.40235409  3.719702
```

```
sim.kmeans.centros3<-kmeans(sim, centros.h3)
centros.h4<-tapply(sim, list(rep(cutree(sim.hclust, 4), ncol(sim)), col(sim)), mean)
centros.h4
```

```
##              1          2
## 1 -0.050585844  0.8021672
## 2 -0.001689665  3.1865553
## 3  3.425229915 -0.1298360
## 4  3.402354090  3.7197022
```

```
sim.kmeans.centros4<-kmeans(sim, centros.h4)
table(sim.kmeans$cluster, sim.kmeans.centros$cluster)
```

```
##
##       1   2
##   1 100   0
##   2   0 150
```

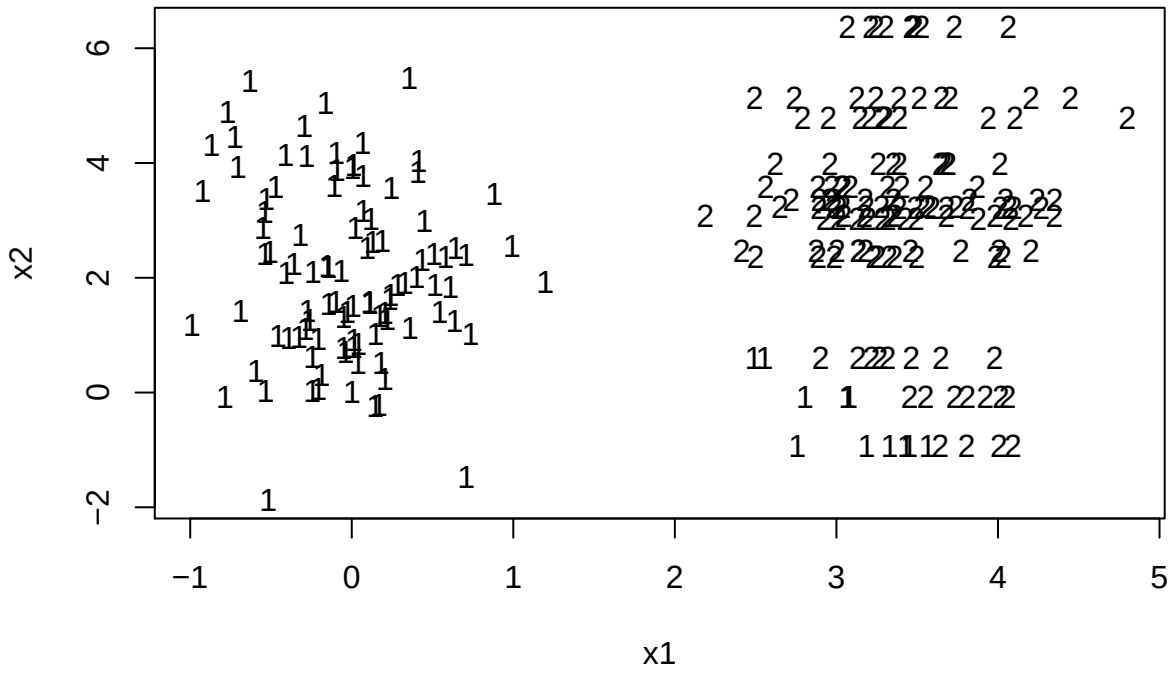
```
table(sim.kmeans3$cluster, sim.kmeans.centros3$cluster)
```

```
##
##       1   2   3
##   1   0   0 120
##   2   0  31   0
##   3  99   0   0
```

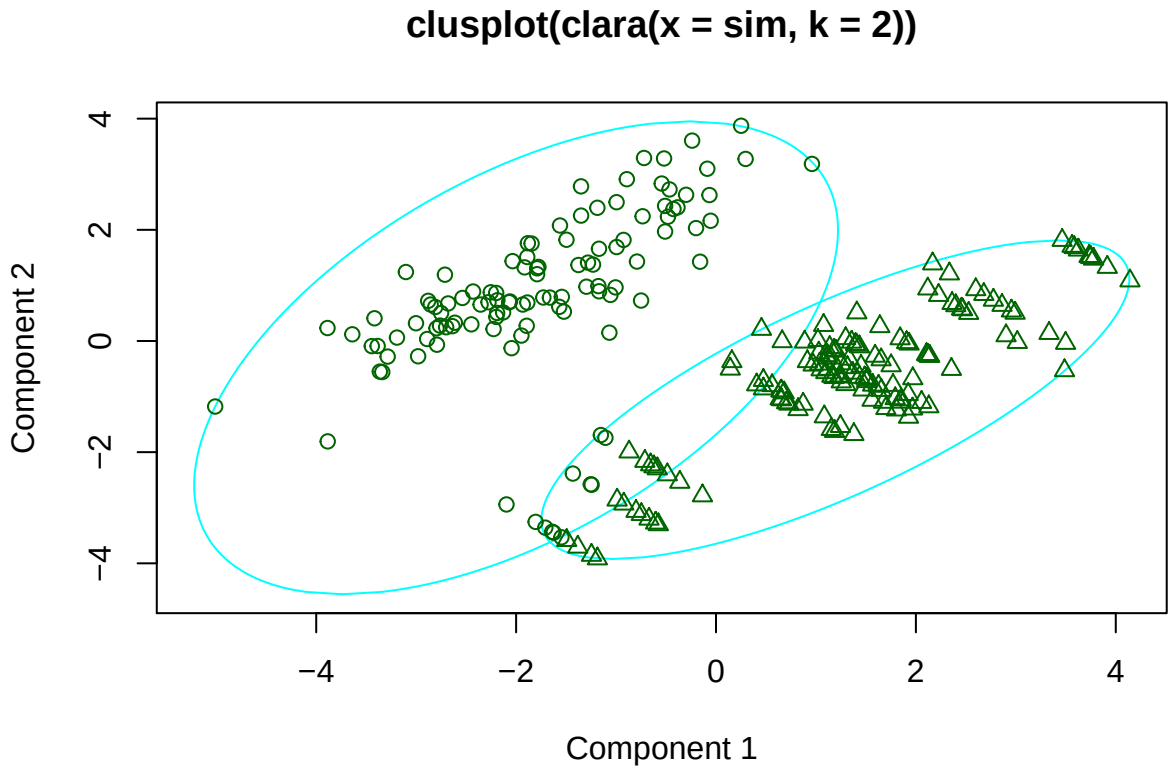
```
table(sim.kmeans4$cluster, sim.kmeans.centros4$cluster)
```

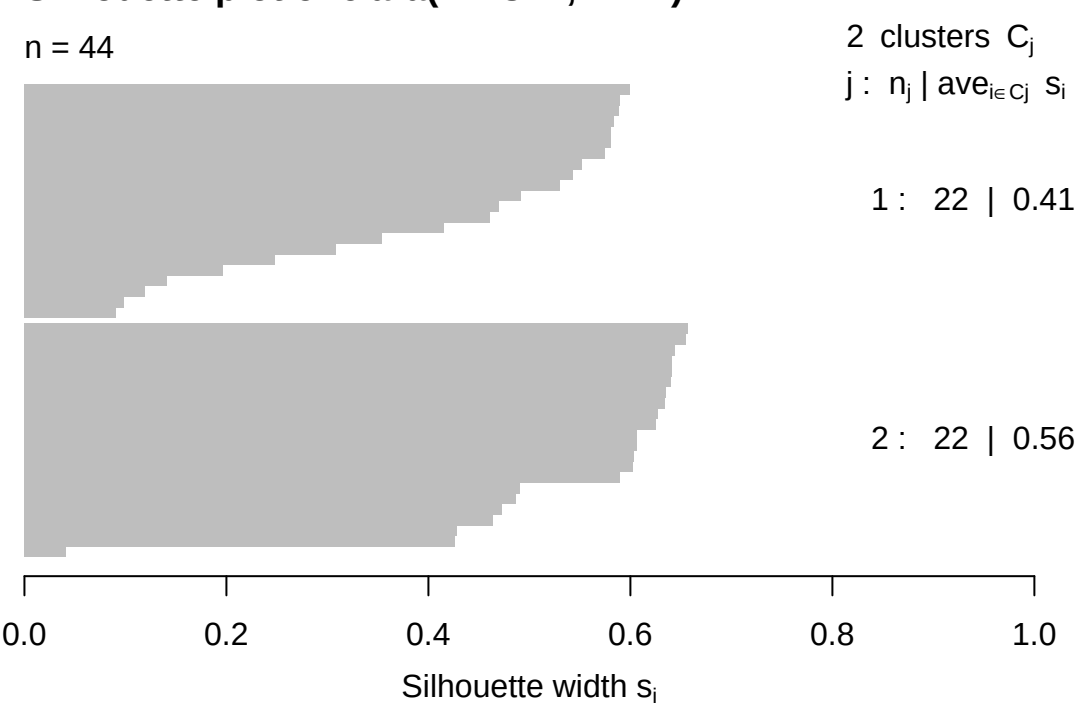
```
##
##      1  2  3  4
##   1 57 41  0  0
##   2  0  0  0 90
##   3  0  1  0 30
##   4  1  0 30  0
```

```
library(cluster)
sim.clara<-clara(sim, 2)
plot(sim, type="n", xlab="x1", ylab="x2")
text(sim, labels=sim.clara$clustering)
```



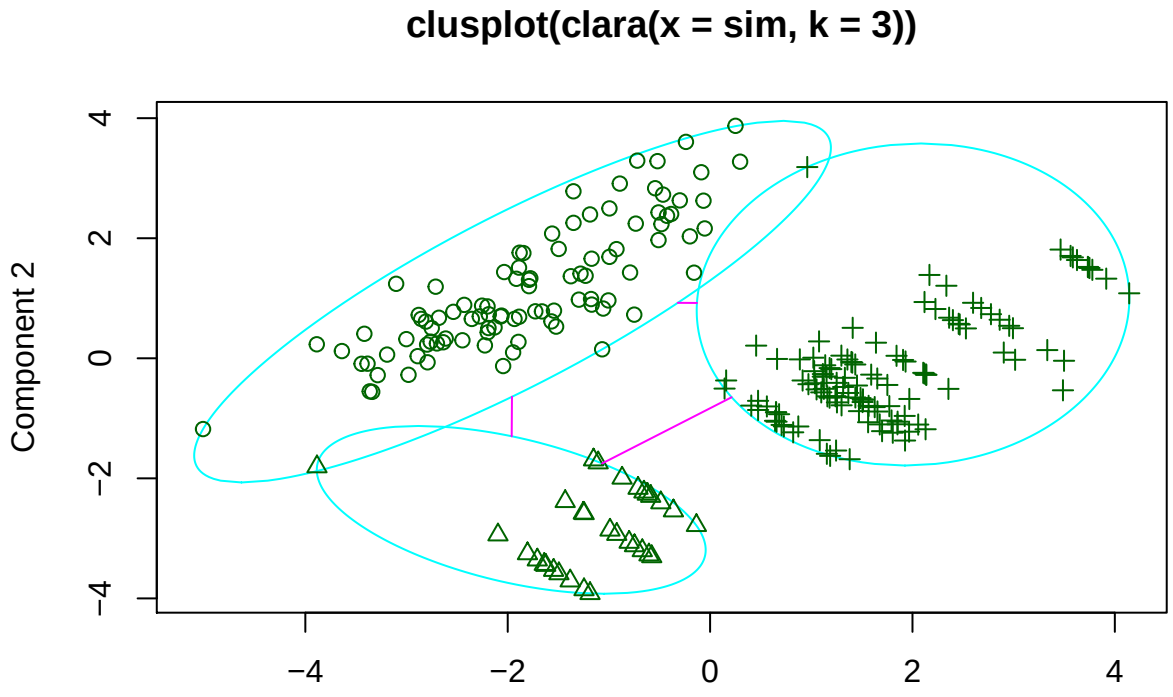

```
plot(sim.clara)
```





Repetimos el proceso para 3 clusters

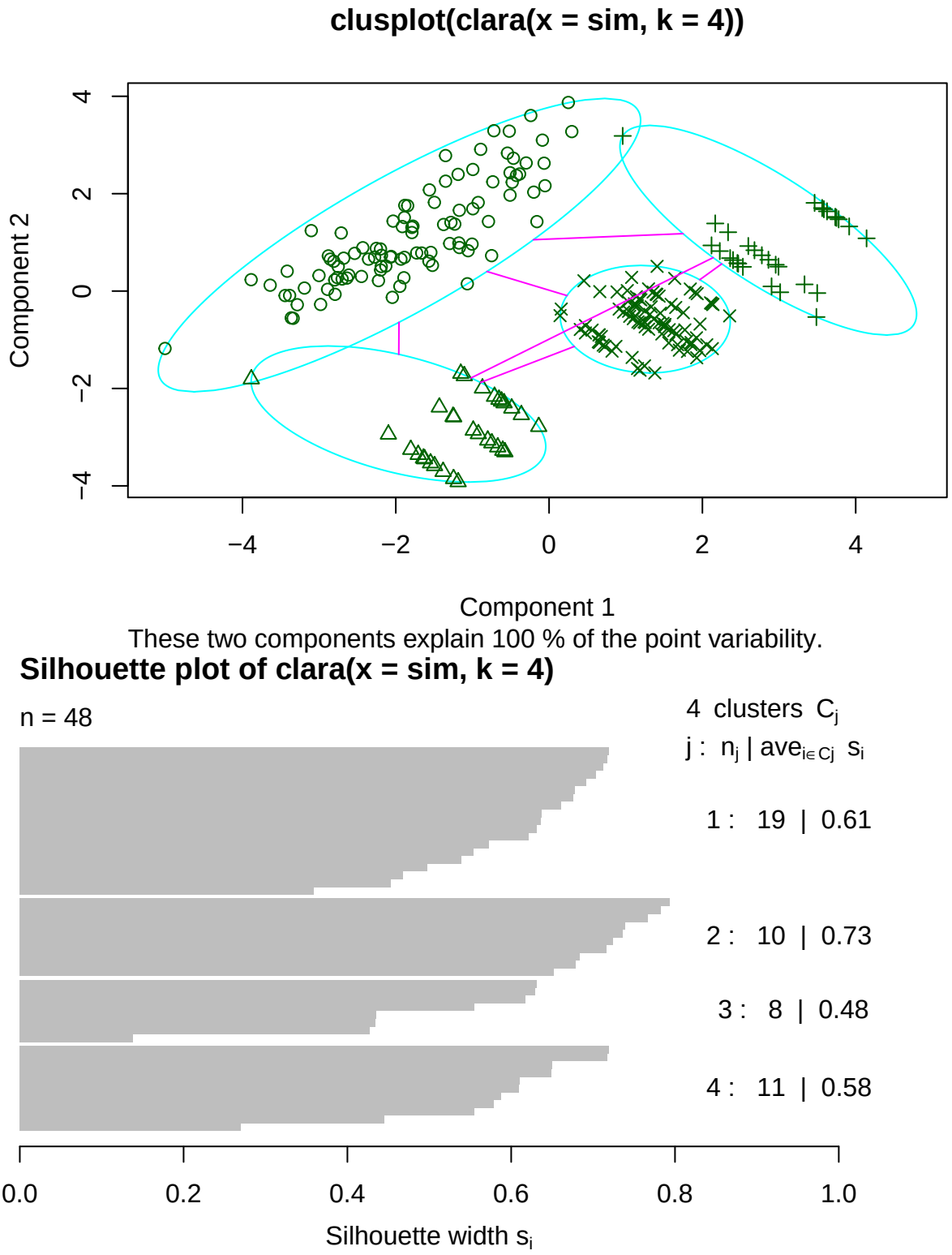
```
sim.clara3<-clara(sim, 3)
plot(sim.clara3)
```



Repetimos

el proceso para 4 clusters. Ahora comparamos los resultados con `kmeans`.

```
sim.clara4<-clara(sim, 4)
plot(sim.clara4)
```




```
table(sim.clara$clustering, sim.kmeans.centros$cluster)
```

```
##
```

```
##       1   2
##   1 100  11
##   2   0 139
```

```
table(sim.clara3$clustering, sim.kmeans.centros3$cluster)
```

```
##
##       1   2   3
##   1  98   0   0
##   2   0  31   0
##   3   1   0 120
```

```
table(sim.clara4$clustering, sim.kmeans.centros4$cluster)
```

```
##
##      1  2  3  4
##   1 57 41  0  0
##   2  1  0 30  0
##   3  0  1  0 30
##   4  0  0  0 90
```

Un ejemplo adicional

First read-in data. Data preparation is required to remove variable scaling effects. To see this, consider a simple example. If you measure weight in Kgs and I do so in Grams - all other variables being the same - we'll get two very different clustering solutions from what is otherwise the same dataset. To get rid of this problem, just copy-paste the following code.

```
mydata = USArrests
mydata<- na.omit(mydata) # listwise deletion of missing
mydata.orig = mydata #save original data copy
mydata <- scale(mydata) # standardize variables
```

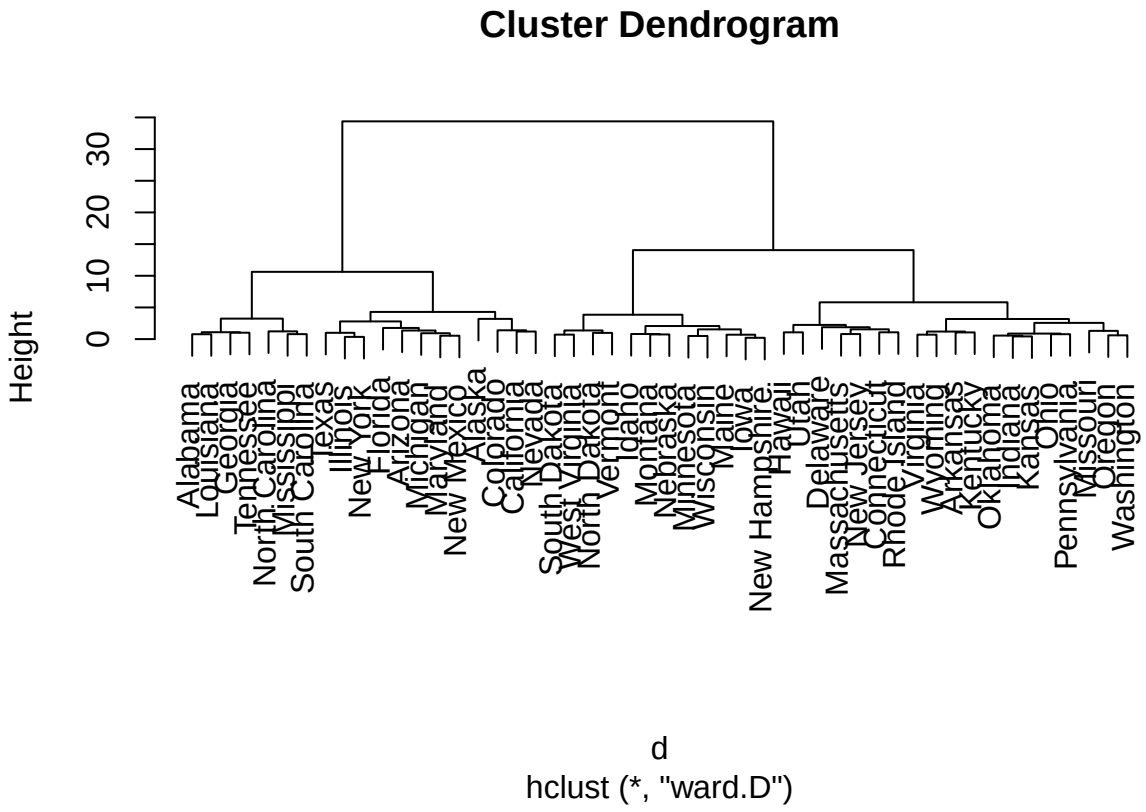
Ward Hierarchical Clustering

```
d <- dist(mydata, method = "euclidean") # distance matrix
fit<- hclust(d, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

Visualizar el gráfico

```
plot(fit) # display dendogram
```



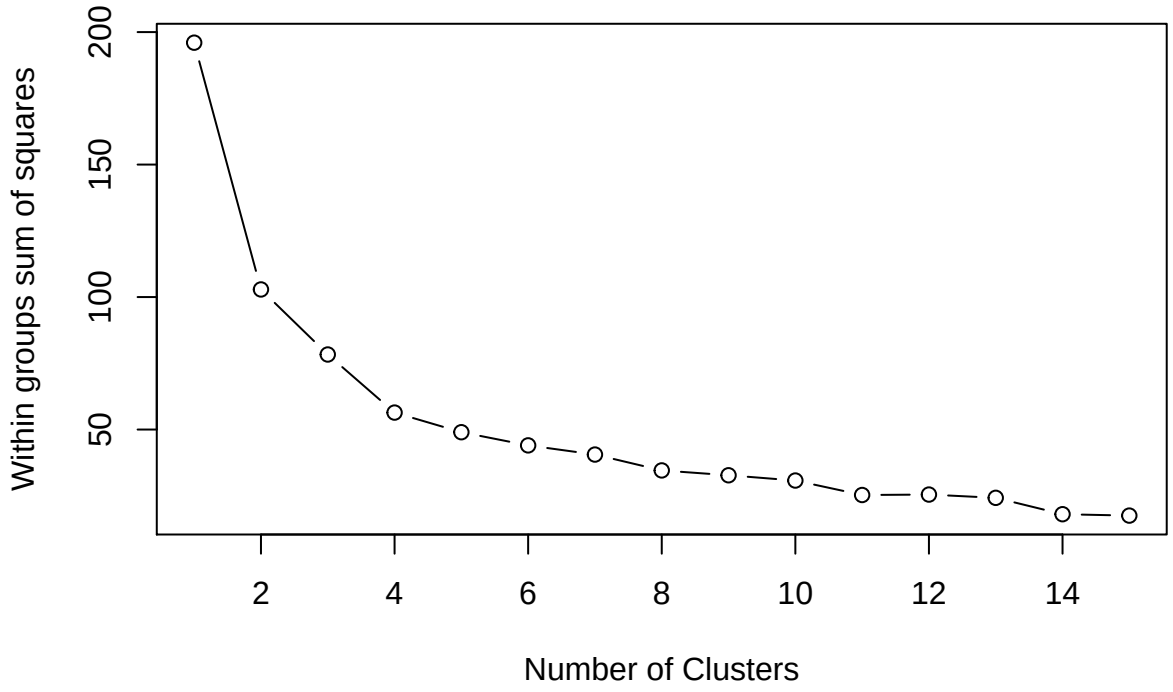

```
k1 = 2
# eyeball the no. of clusters
# cut tree into k1 clusters
groups = cutree(fit, k=k1)
```

Mostrar el dendograma con bordes rojos alrededor de k clusters

```
#rect.hclust(fit, k=2, border="red")
```

De nuevo la pregunta aparece, ¿cómo conocer el número óptimo de segmentos? Observando el dendrograma en ciertas ocasiones puede ser de ayuda para tomar la decisión. Pero en otras ocasiones, ¿qué deberíamos hacer? La mayoría, si no todos, los programas esperan que conozcas el número correcto de segmentos a formar a la hora de utilizar la función `kmaenas`. `R` lo hace mejor y te muestra un one better **scree plot** de la familia de los gráficos que te muestran cómo la varianza dentro de los segmentos (un idicador de la calidad de la solución) varía con el número de segmentos formados. Por ello, con **R** puedes tomar una decisión informada.

```
# Determine number of clusters
wss<- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,centers=i) $withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")
```




```
# Look for an "elbow" in the scree plot
# Use optimal no. of clusters in k-means
k1=2
```

K-Means Cluster Analysis

```
fit <- kmeans(mydata, k1) # k1 cluster solution
fit$centers
```

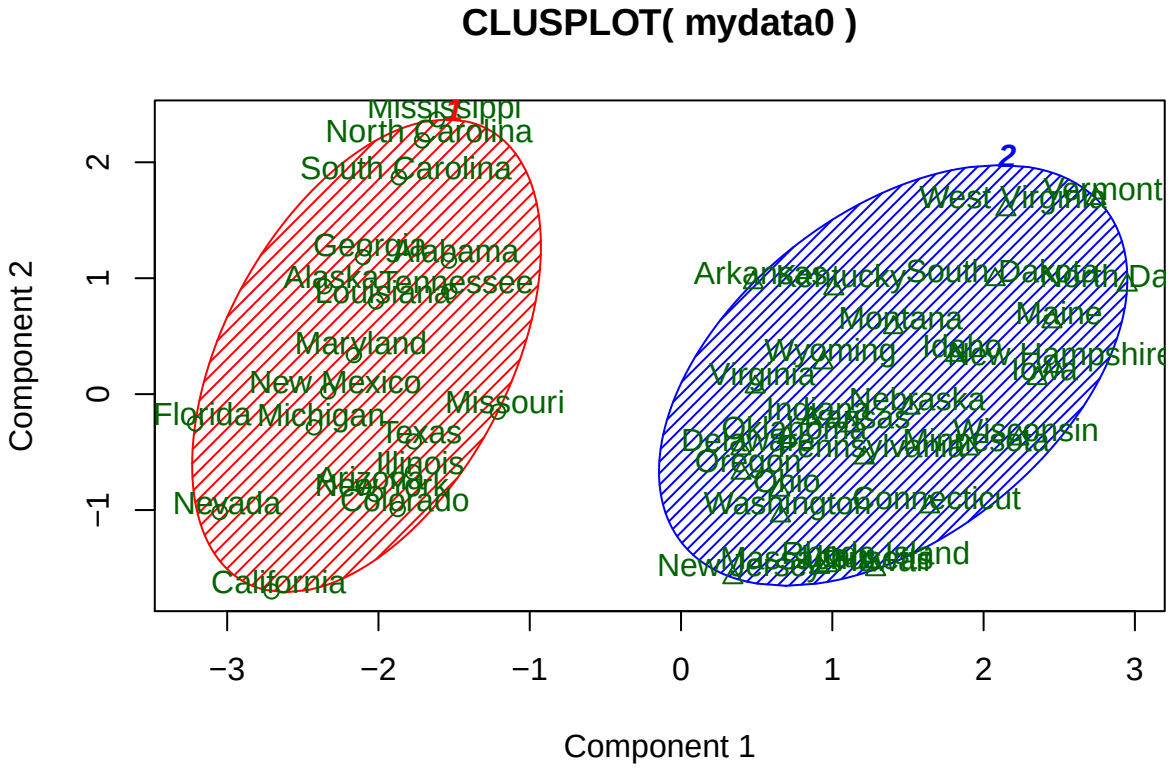
```
##      Murder    Assault   UrbanPop       Rape
## 1  1.004934  1.0138274  0.1975853  0.8469650
## 2 -0.669956 -0.6758849 -0.1317235 -0.5646433
```

```
#Murder Assault UrbanPop Rape
# get cluster means of original data
aggregate(mydata.orig,by=list(fit$cluster),FUN=mean)
```

```
##   Group.1 Murder  Assault UrbanPop     Rape
## 1       1 12.165 255.2500 68.40000 29.16500
## 2       2  4.870 114.4333 63.63333 15.94333
```

```
# Group.1 Murder Assault UrbanPop Rape
# append cluster assignment
mydata0<- data.frame(mydata.orig, fit$cluster)
# Cluster Plot against 1st 2 principal components
# vary parameters for most readable graph
```

```
library(cluster)
clusplot(mydata0, fit$cluster, color=TRUE, shade=TRUE,labels=2,
lines=0)
```



Segmentación de muestras mixtas basada en un modelo estadístico

Los modelos de clases latentes suponen que la muestra que estamos analizando es representativa de una población mixta compuesta por k segmentos de tamaño $n(k)$. Por ello la probabilidad de que las medidas obtenidas del individuos i en las p variables o propiedades, y_i, hayan sido generadas por un modelo M con parámetros θ es igual a la suma, para las k clases, de la probabilidad de que la observación del individuo i pertenezca a cada una de las clases $f_k(.)$ ponderada por la proporción en que se han mezclado, π_k. Concretamente,

$$f\left(y_i|\theta\right) = \sum_{k=1}^{k} \pi_k f_k\left(y_i|\theta_k\right)$$

Para estimar los parámetros del modelo M, $\theta = (\theta_k)$, la media y la matrixz de varianzas de cada clase latente, $\theta_k = (\mu_k, \sum_k)$, tenemos que formar la función de verosimilitud y maximizar su logaritmo mediante el procedimiento EM, para un determinado número de clases latentes, k:

$$L\left(M, K\right) = \sum_{i=1}^{n} \log \left[f\left(y_i|\theta\right) = \sum_{k=1}^{k} \pi_k f_k\left(y_i|\theta_k\right) \right] \backslash n$$

Clasificación a posteriori

Los modelos de clases latentes utilizados en los problemas de segmentación no solamente están interesados en estima rlos parámetros del modelo que genera la muestra mixta sino también en la clasificación de los individuos a los k grupos. Para ello utiliza las probabilidades calculadas después de estimar el modelo. Así

una observación $\mathbf{y}_\text{i}$ se asigna al grupo cuya probabilidad de pertenencia es mayor, es decir, la probabilidad de pertenecer al grupo k condicionada a las características medidas del sujeto i, $\mathbf{y}_i$,

$$\pi_{k|y_i} = \frac{\acute{\pi}_k f_k\left(\mathbf{y}_i|\acute{\theta}_k\right)}{\sum_{k=1}^{k} \acute{\pi}_k f_k\left(\mathbf{y}_i|\acute{\theta}_k\right)}$$

Selección del modelo

Los modelos estimados con un número diferente de clases latentes o grupos no son modelos anidados, el modelo con k-1 segmentos o grupos no es un modelo reducido del modelo con k segmentos. Por ello tenemos que basarnos en criterios heurísticos nos aproximan la cantidad de información que incorpora y al mismo tiempo tienen en cuenta el número de parámetros que los definen. Concretamente el Bayesian Information Criteria, BIC, y el Consistent Akaike's Information Criteria, CAIC. Estos criterios favorecen a los modelos que muestran un menor valor del estadístico.

$$BIC(L(M), K) = -2logL(M) + log(N)npar$$

$$CAIC(L(M), K) = -2logL(M) + (log(N) + 1)npar$$

Model Based Clustering

```
library(mclust)
```

```
## Package 'mclust' version 4.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
fit.Mclust<- Mclust(mydata)
fit.Mclust
```

```
## 'Mclust' model object:
##  best model: diagonal, equal shape (VEI) with 3 components
```

```
names(fit.Mclust)
```

```
##  [1] "call"           "data"           "modelName"      "n"
##  [5] "d"              "G"              "BIC"            "bic"
##  [9] "loglik"         "df"             "hypvol"         "parameters"
## [13] "z"              "classification" "uncertainty"
```

```
# view solution summary
```

El mejor modelo tiene varianzas iguales diagonales (VEI) con tres segmentos

```
fit.Mclust$BIC
```

```
##         EII       VII       EEI       VEI       EVI       VVI       EEE
## 1 -583.0950 -583.0950 -594.8311 -594.8311 -594.8311 -594.8311 -524.0454
## 2 -538.8430 -540.8779 -527.0645 -528.6274 -535.4765 -537.2752 -526.5856
## 3 -537.5739 -525.7362 -526.2295 -512.9696 -543.7186 -527.7408 -532.9479
## 4 -521.7731 -526.6936 -517.8151 -519.4226 -543.1230 -543.7168 -534.2090
## 5 -533.2088 -538.0439 -531.1501 -534.8807 -561.6269 -564.2794 -548.5223
## 6 -548.2697 -545.5376 -549.8833 -545.3595 -591.5334 -583.3005 -567.0752
## 7 -554.7524 -564.2459 -555.9070 -562.4990 -596.0957 -601.8503 -567.1581
## 8 -564.6494 -571.8128 -568.0429 -580.0928 -612.3113 -617.3733 -576.3987
## 9 -576.0047 -586.1872 -577.0350 -585.9141 -643.0224 -648.6003 -578.1330
##         EEV       VEV       VVV
## 1 -524.0454 -524.0454 -524.0454
## 2 -525.2101 -527.3191 -537.8664
## 3 -539.6804 -550.5477 -567.0451
## 4 -562.0097 -575.3697 -588.1424
## 5 -576.7656 -594.4986 -628.3149
## 6 -625.1663 -622.9173        NA
## 7 -623.9653 -643.6280        NA
## 8 -642.8520 -656.7630        NA
## 9 -679.3634 -679.5256        NA
## attr(,"G")
## [1] 1 2 3 4 5 6 7 8 9
## attr(,"modelNames")
##  [1] "EII" "VII" "EEI" "VEI" "EVI" "VVI" "EEE" "EEV" "VEV" "VVV"
## attr(,"oneD")
## [1] FALSE
```

```r
# lookup all the options attempted
classif <- fit.Mclust$classification
# classifiation vector
mydata1 <- cbind(mydata.orig, classif)
# append to dataset
# draw dendrogram with red borders around the k1 clusters
mydata1[1:10,]
```

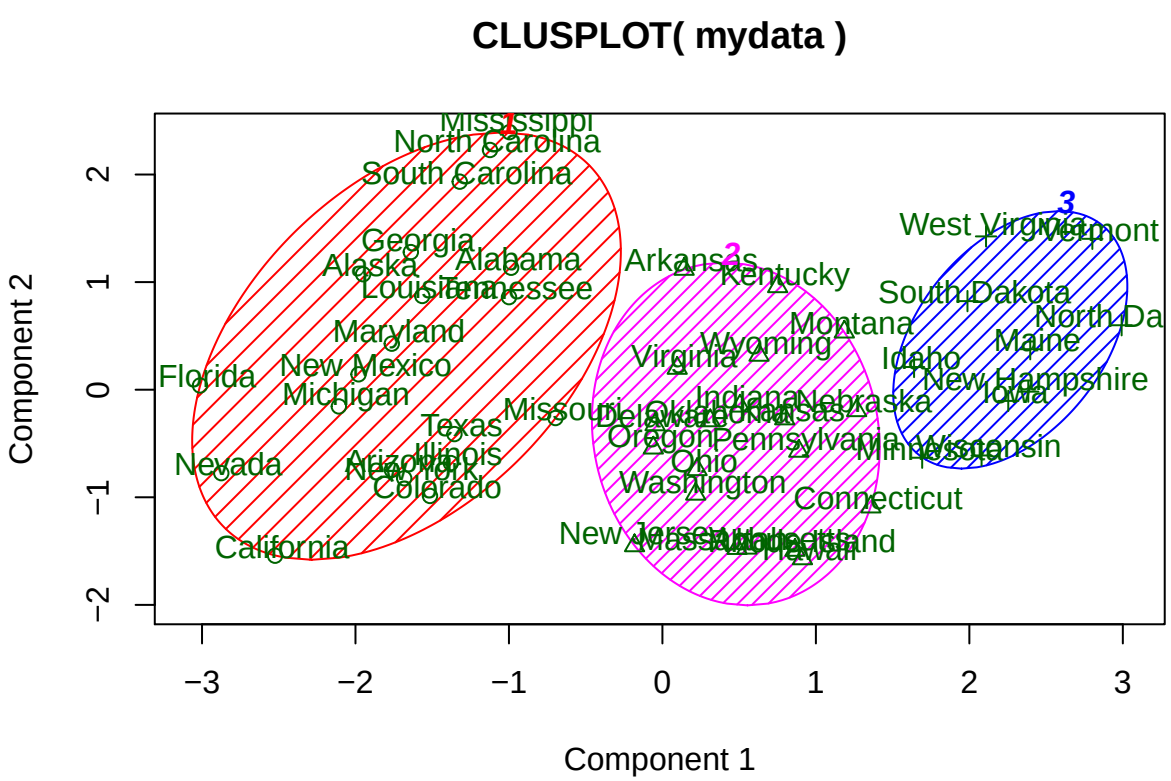
```
##             Murder Assault UrbanPop Rape classif
## Alabama       13.2     236       58 21.2       1
## Alaska        10.0     263       48 44.5       1
## Arizona        8.1     294       80 31.0       1
## Arkansas       8.8     190       50 19.5       2
## California     9.0     276       91 40.6       1
## Colorado       7.9     204       78 38.7       1
## Connecticut    3.3     110       77 11.1       2
## Delaware       5.9     238       72 15.8       2
## Florida       15.4     335       80 31.9       1
## Georgia       17.4     211       60 25.8       1
```

```r
#view top 10 rows
table(fit$cluster, classif)
```

```
##    classif
##      1  2  3
##   1 20  0  0
##   2  0 20 10
```

Si queremos guardar el resultado

```
#write.table(mydata1,file.choose())
#save output
fit1=cbind(classif)
rownames(fit1)=rownames(mydata)
library(cluster)
clusplot(mydata, fit1, color=TRUE, shade=TRUE,labels=2, lines=0)
```



Paara obtener las medias en los segmenotos

```
cmeans=aggregate(mydata.orig,by=list(classif),FUN=mean); cmeans
```

```
##   Group.1 Murder Assault UrbanPop   Rape
## 1       1 12.165  255.25    68.40 29.165
## 2       2  5.965  136.60    69.95 18.460
## 3       3  2.680   70.10    51.00 10.910
```

Preparación de los datos: Análisis componentes principals

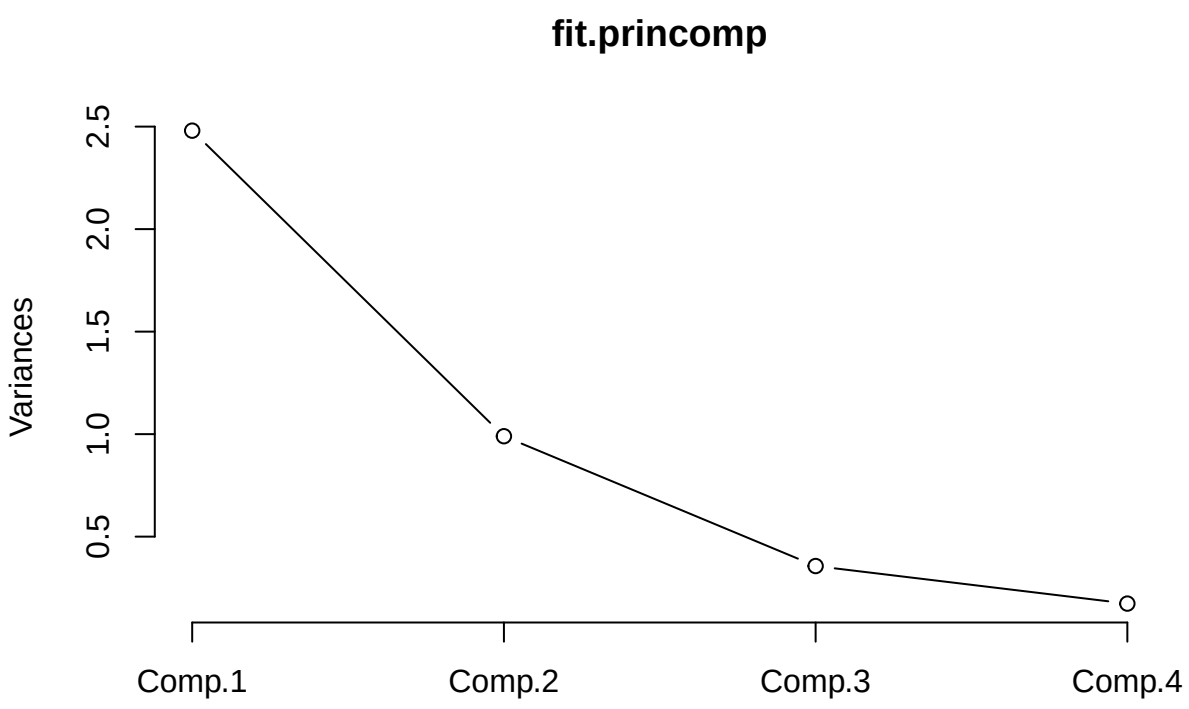
```
# Principal Components Analysis
# entering raw data and extracting PCs
# from the correlation matrix
fit.princomp<- princomp(mydata, cor=TRUE)
summary(fit.princomp)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4
## Standard deviation     1.5748783 0.9948694 0.5971291 0.41644938
## Proportion of Variance 0.6200604 0.2474413 0.0891408 0.04335752
## Cumulative Proportion  0.6200604 0.8675017 0.9566425 1.00000000
```

```
# print variance accounted for
loadings(fit.princomp)
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4
## Murder   -0.536  0.418 -0.341  0.649
## Assault  -0.583  0.188 -0.268 -0.743
## UrbanPop -0.278 -0.873 -0.378  0.134
## Rape     -0.543 -0.167  0.818
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

```
# pc loadings
plot(fit.princomp,type="lines")
```



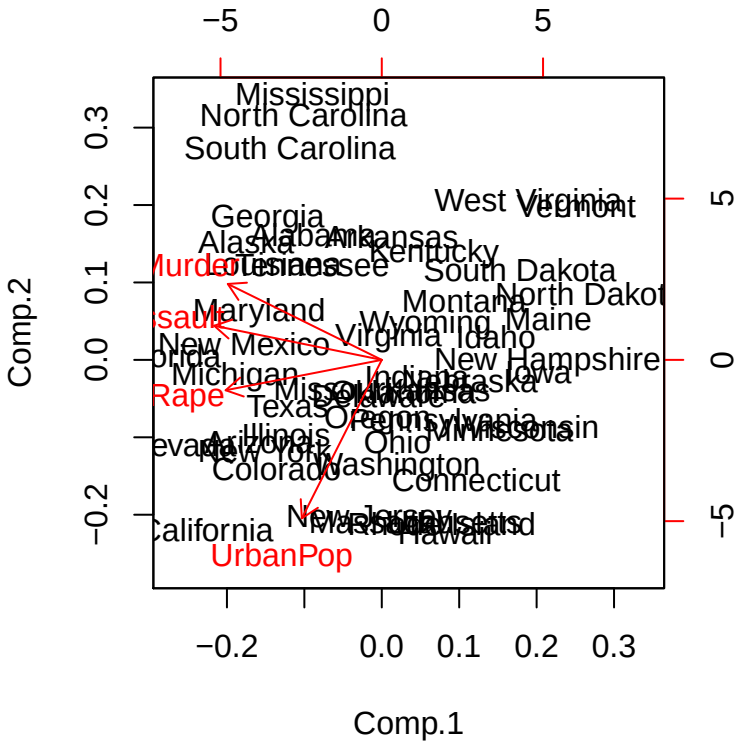

```
# scree plot
```

Puntuación en los compopnentes principales

```
head(fit.princomp$scores)
```

```
##                Comp.1     Comp.2      Comp.3       Comp.4
## Alabama    -0.9855659  1.1333924 -0.44426879  0.156267145
## Alaska     -1.9501378  1.0732133  2.04000333 -0.438583440
## Arizona    -1.7631635 -0.7459568  0.05478082 -0.834652924
## Arkansas    0.1414203  1.1197968  0.11457369 -0.182810896
## California -2.5239801 -1.5429340  0.59855680 -0.341996478
## Colorado   -1.5145629 -0.9875551  1.09500699  0.001464887
```

```
# the principal components
biplot(fit.princomp)
```



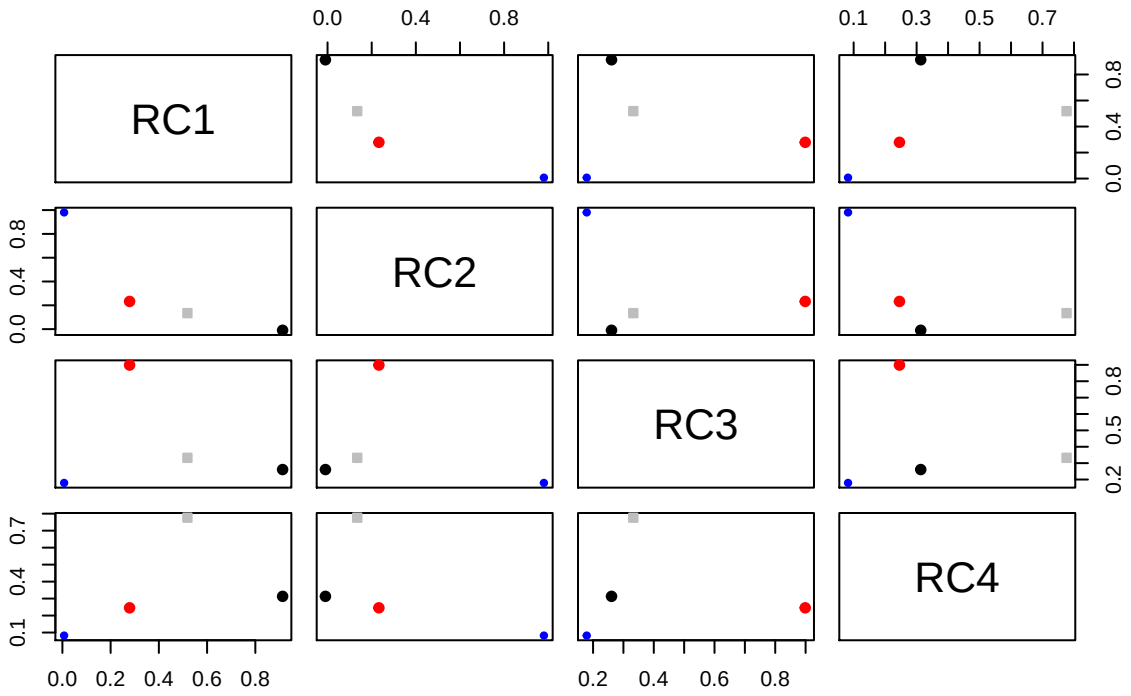

```
#Use cor=FALSE to base the principal components on the covariance matrix.
#Use the covmat= option to enter a correlation or covariance matrixdirectly.
#If entering a covariance matrix, include the option n.obs=.
#The principal( ) function in the psych package can be used to extract and rotate principal components.
# Varimax Rotated Principal Components
# retaining 5 components
library(psych)
```

```
##
## Attaching package: 'psych'
##
## The following object is masked from 'package:mclust':
##
##     sim
```

```r
library(GPArotation)
fit.principal<- principal(mydata, nfactors=4, rotate="varimax")
fit.principal
```

```
## Principal Components Analysis
## Call: principal(r = mydata, nfactors = 4, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##           RC1   RC2  RC3  RC4 h2       u2
## Murder   0.91 -0.01 0.26 0.31  1  3.3e-16
## Assault  0.52  0.13 0.33 0.78  1 -2.2e-16
## UrbanPop 0.01  0.98 0.18 0.08  1  3.0e-15
## Rape     0.28  0.23 0.90 0.25  1  1.2e-15
##
##                        RC1  RC2  RC3  RC4
## SS loadings           1.18 1.03 1.02 0.77
## Proportion Var        0.30 0.26 0.25 0.19
## Cumulative Var        0.30 0.55 0.81 1.00
## Proportion Explained  0.30 0.26 0.25 0.19
## Cumulative Proportion 0.30 0.55 0.81 1.00
##
## Test of the hypothesis that 4 components are sufficient.
##
## The degrees of freedom for the null model are  6  and the objective function was  1.89
## The degrees of freedom for the model are -4  and the objective function was  0
## The total number of observations was  50  with MLE Chi Square =  0  with prob <  NA
##
## Fit based upon off diagonal values = 1
```

```r
# print results
plot(fit.principal)
```




```
?principal
```

mydata can be a raw data matrix or a covariance matrix. Pairwise deletion of missing data is used. rotate can “none”, “varimax”, “quatimax”, “promax”, “oblimin”, “simplimax”, or “cluster” .

Identificación de los individuos que forman los segmentos