Laboratorio de clustering

Marcelo Soria - Mariana Landoni

Setiembre, 2016

Para esta practica de laboratorio vamos a usar un dataset que se puede descargar desde [aqui] (http://archive.ics.uci.edu/ml/datasets/wiki4HE). Son datos de una encuesta entre profesores de dos universidades españolas sobre uso de Wikipedia como recurso educativo.

Primero cargamos algunos paquetes que vamos a necesitar.

```
library(cluster)
library(MASS)
library(fpc)
```

Descargamos los datos de su url y los asignamos a un objeto R, y guardamos un backup.

```
encuesta <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/00334/wiki4HE.csv",
                       header=T, sep=";", stringsAsFactors = F)
encuesta.bak <- encuesta
```

Y hacemos algunos chequeos.

```
dim(encuesta)
head(encuesta)
str(encuesta)
```

- ¿Qué información nos dan los comandos anteriores?

A continuación vamos a revisar la distribución de los datos de unas cuantas variables.

```
sapply(encuesta[,c(3, 5, 7:53)], table)
```

- ¿Cómo están indicados los datos faltantes?

Limpieza y preparación de los datos

El sexo lo vamos a indicar como “F” y “M”.

```
encuesta$GENDER <- as.factor(ifelse(encuesta$GENDER == 1, "F", "M"))
```

En el caso de la variable DOMAIN vemos que hay datos faltantes, y además hace falta convertirla a una variable de tipo factor.

```
encuesta$DOMAIN[ encuesta$DOMAIN=="?" ] <- NA
encuesta$DOMAIN <- as.factor(encuesta$DOMAIN)
```

Este procedimiento lo tenemos que repetir para varias variables. Es mejor crear una función para hacer esto, y después hacer la conversión.

```
chr.a.fac <- function(x){
  x[ x == "?" ]  <- NA
  x <- as.factor(x)
}

encuesta$UOC_POSITION <- chr.a.fac( encuesta$UOC_POSITION )
encuesta$OTHER_POSITION <- chr.a.fac( encuesta$OTHER_POSITION )
encuesta$OTHERSTATUS <- chr.a.fac( encuesta$OTHERSTATUS )
```

La variable YEARSEXP es numérica

```
encuesta$YEARSEXP <- as.numeric(encuesta$YEARSEXP)
```

```
## Warning: NAs introduced by coercion
```

Hay dos variables que deberían estar codificadas como de tipo lógico. Luego, la variable UNIVERSITY tiene que estar codificada como de tipo factor. Las respuestas a las encuestas están codificadas en una escala 1:5, al convertirlas a numéricas, los signos "?" se recodifican como NAs. Estas respuestas van desde la columna 11 a la 53.

```
encuesta$PhD <- as.logical(encuesta$PhD)
encuesta$USERWIKI <- as.logical(as.numeric(encuesta$USERWIKI))
encuesta$UNIVERSITY <- as.factor(encuesta$UNIVERSITY)
encuesta[, 11:53] <- sapply(encuesta[, 11:53], as.numeric)
```

Para el trabajo que sigue solo vamos a trabajar con los datos de la universidad que más respuestas tiene. Además vamos a eliminar tres variables, una es justamente UNIVERSITY, que después del filtrado es redundante. Las otras otras dos son más relevante para los participantes de la otra universidad, o no tienen información completa.

Hay datos faltantes y para simplificar el análisis, solo nos vamos quedar con los casos completos.

En último lugar vamos a separar los datos profesionales y demográficos de los encuestados de sus respuestas.

```
# encuesta.uoc <- subset(encuesta, UNIVERSITY==1, select= -c(6,8,9))
encuesta.uoc <- subset(encuesta, UNIVERSITY==1, select= -c(UNIVERSITY,OTHER_POSITION,OTHERSTATUS))

table(complete.cases(encuesta.uoc))
```

```
##
## FALSE  TRUE
##   258   542
```

```
encuesta.uoc.c <- encuesta.uoc[complete.cases(encuesta.uoc), ]
uoc.personal <- encuesta.uoc.c[,1:7]
uoc.preguntas <- encuesta.uoc.c[, 8:50]
```

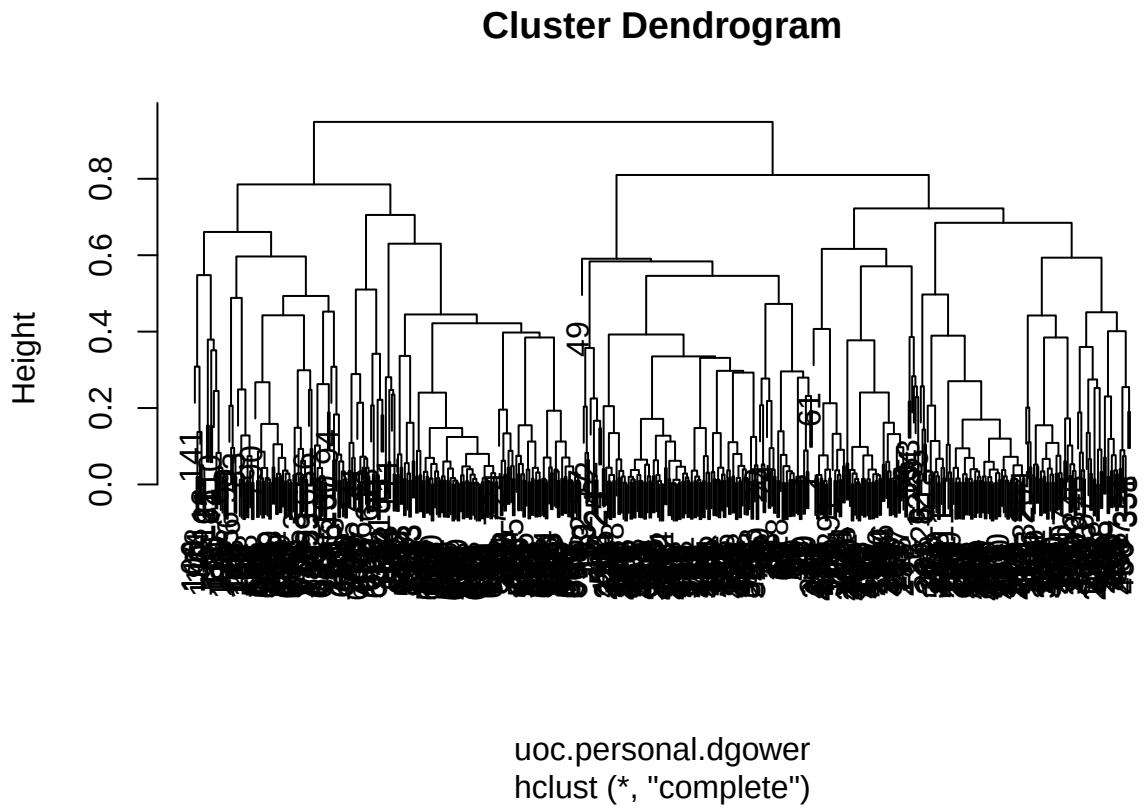
Análisis

Vamos a construir una matriz de distancias de Gower para los datos personales, y realizamos un cluster jerárquico para tener una imprema impresión sobre cómo se agrupan los datos.

```
uoc.personal.dgower <- daisy(uoc.personal, metric="gower", stand=T)
```

```
## Warning in daisy(uoc.personal, metric = "gower", stand = T): setting
## 'logical' variables 4, 7 to type 'asymm'
```

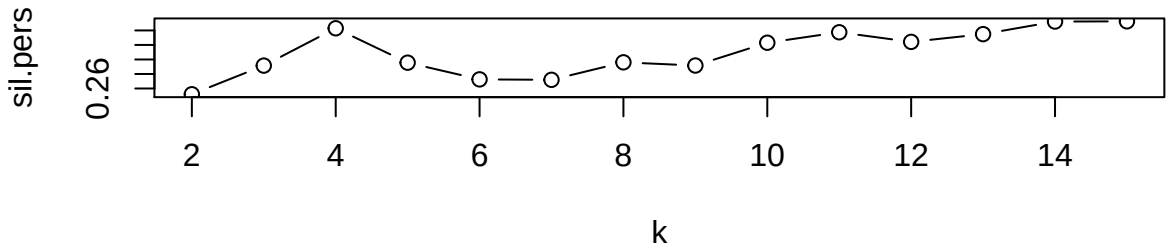
```
plot(hclust(uoc.personal.dgower))
```

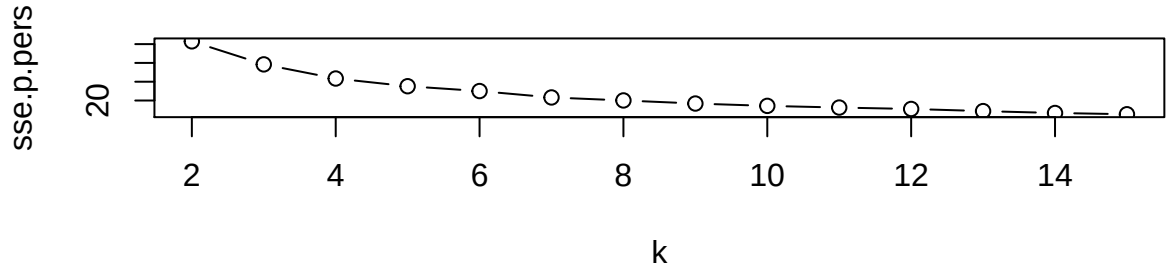


- ¿Que se puede decir de la presencia de grupos en el dataset?

Para agrupar los datos datos vamos a usar el método PAM, y como desconocemos el mejor valor de K a utilizar, vamos a probar varios y después usar los gráficos de SSE vs. k y Silhouette vs. k.

```
sse.p.pers <- array()
sil.pers <- array()
kit <- 14
for(i in 1:kit){
  personal.pam <- pam(uoc.personal.dgower, i+1, diss = T)
  pers.meds <- personal.pam$medoids[personal.pam$clustering]
  sse.p.pers[i] <- sum(as.matrix(uoc.personal.dgower)[cbind(row.names(uoc.personal), pers.meds)]^2)
  sil.pers[i] <- personal.pam$silinfo$avg.width
}

par(mfrow=c(2,1))
plot(2:(kit+1), sil.pers, type="b", xlab="k")
plot(2:(kit+1), sse.p.pers, type="b", xlab="k")
```



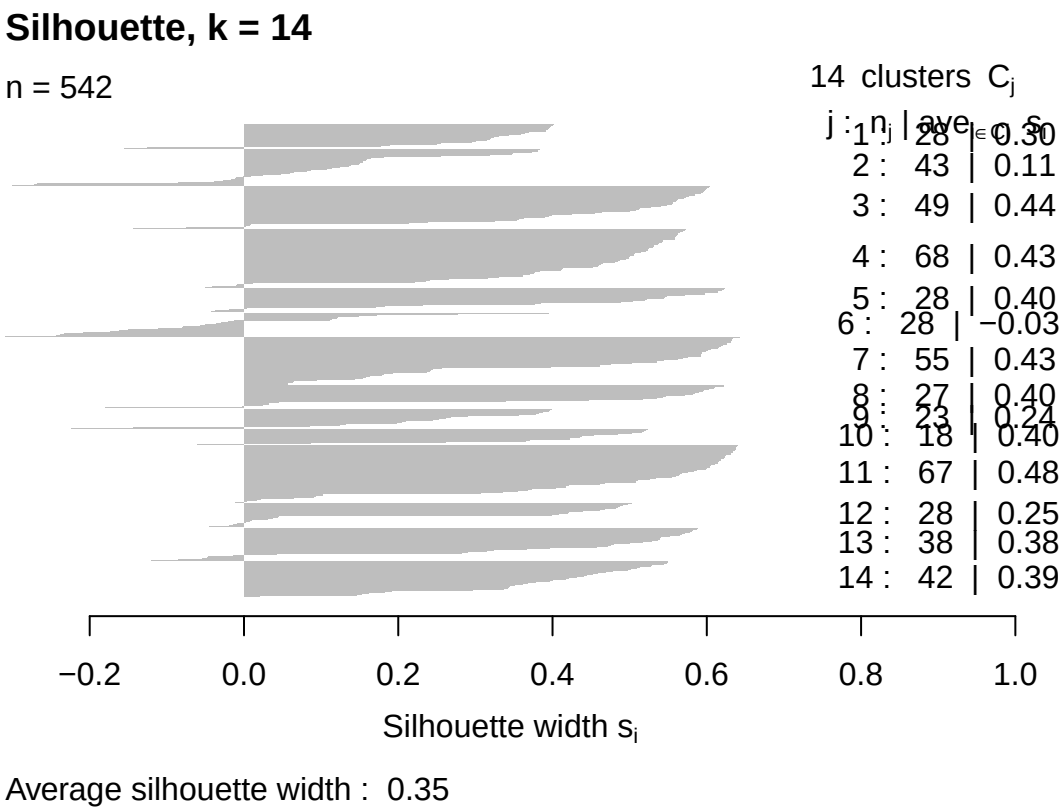



```
par(mfrow=c(1,1))
```

- ¿Cómo se interpretan estos gráficos?
- ¿Por qué el valor de Silhouette sube, baja a partir de k $>$ 4 y después vuelve a subir gradualmente? Ayuda: mirar el cluster jerárquico que hicimos antes.

Probamos primero con k=14.

```
personal.pam <- pam(uoc.personal.dgower, 14, diss = T)
plot(silhouette(personal.pam), main="Silhouette, k = 14")
```



Valores que toman los prototipos

```
data.frame(uoc.personal[personal.pam$medoids,], tamaño=personal.pam$clusinfo[,1])
```

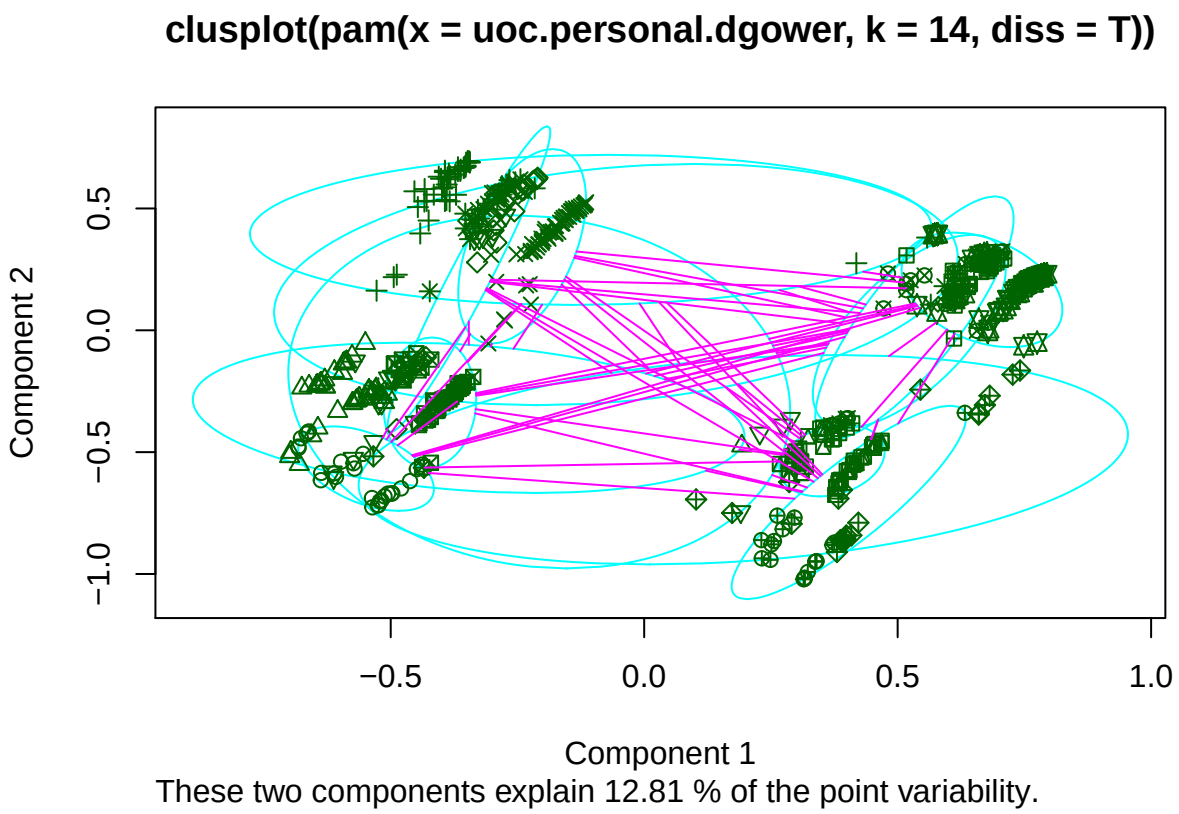
```
##     AGE GENDER DOMAIN   PhD YEARSEXP UOC_POSITION USERWIKI tamaño
## 13   38      M      6  TRUE       15            2    FALSE     28
## 398  45      M      4  TRUE       18            6    FALSE     43
## 154  39      M      4 FALSE        5            6    FALSE     49
## 149  43      M      6 FALSE        7            6    FALSE     68
## 244  38      M      1 FALSE        7            6    FALSE     28
## 345  53      M      1  TRUE       10            6     TRUE     28
## 179  41      M      6  TRUE       10            6    FALSE     55
## 303  44      M      5 FALSE       12            6    FALSE     27
## 86   40      F      6  TRUE       12            3    FALSE     23
## 128  43      F      6  TRUE       20            2    FALSE     18
## 734  35      F      6 FALSE        4            6    FALSE     67
## 685  38      F      3 FALSE        7            6    FALSE     28
## 698  46      F      1 FALSE        5            6    FALSE     38
## 699  41      F      6  TRUE       14            6    FALSE     42
```

```
personal.pam$isolation
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## no no no no no no no no no no no no no no
## Levels: no L L*
```

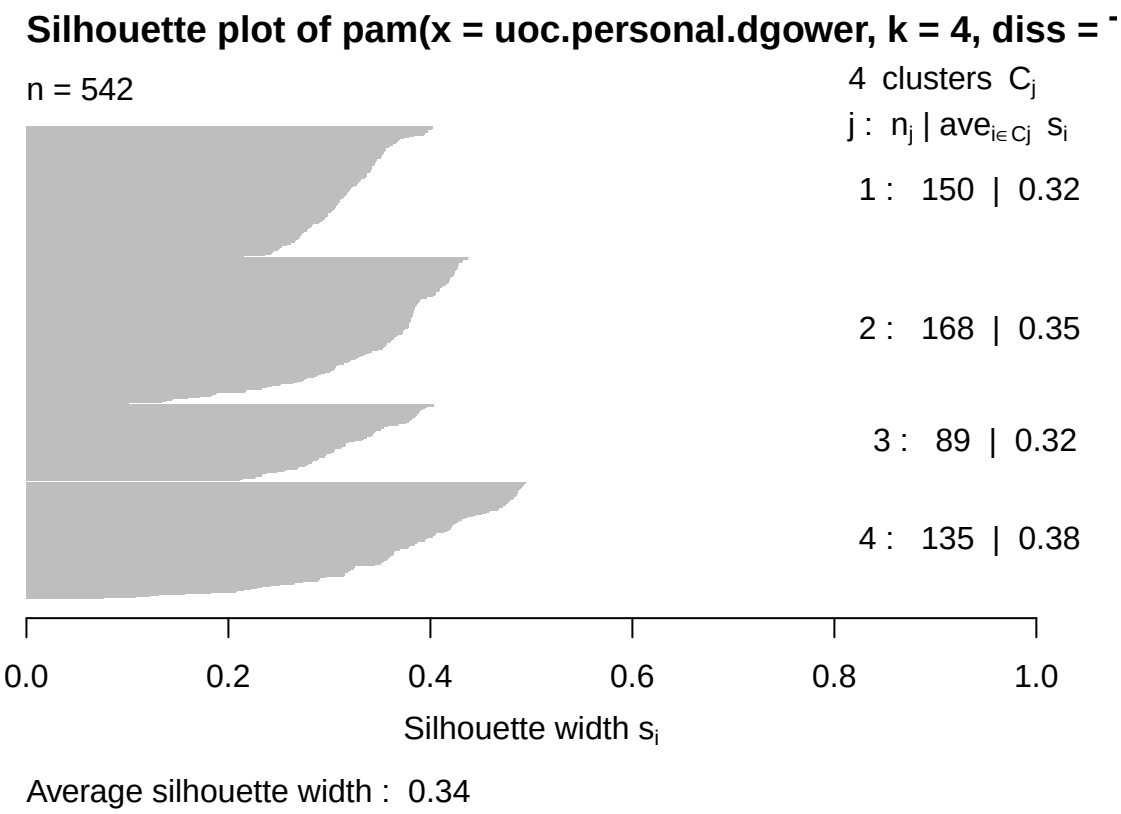
Y un gráfico para ver la relación entre grupos e individuos.

```
clusplot(personal.pam)
```



Y nos quedamos con k=4,

```
# Probamos k=4
personal.pam <- pam(uoc.personal.dgower, 4, diss = T)
plot(silhouette(personal.pam))
```



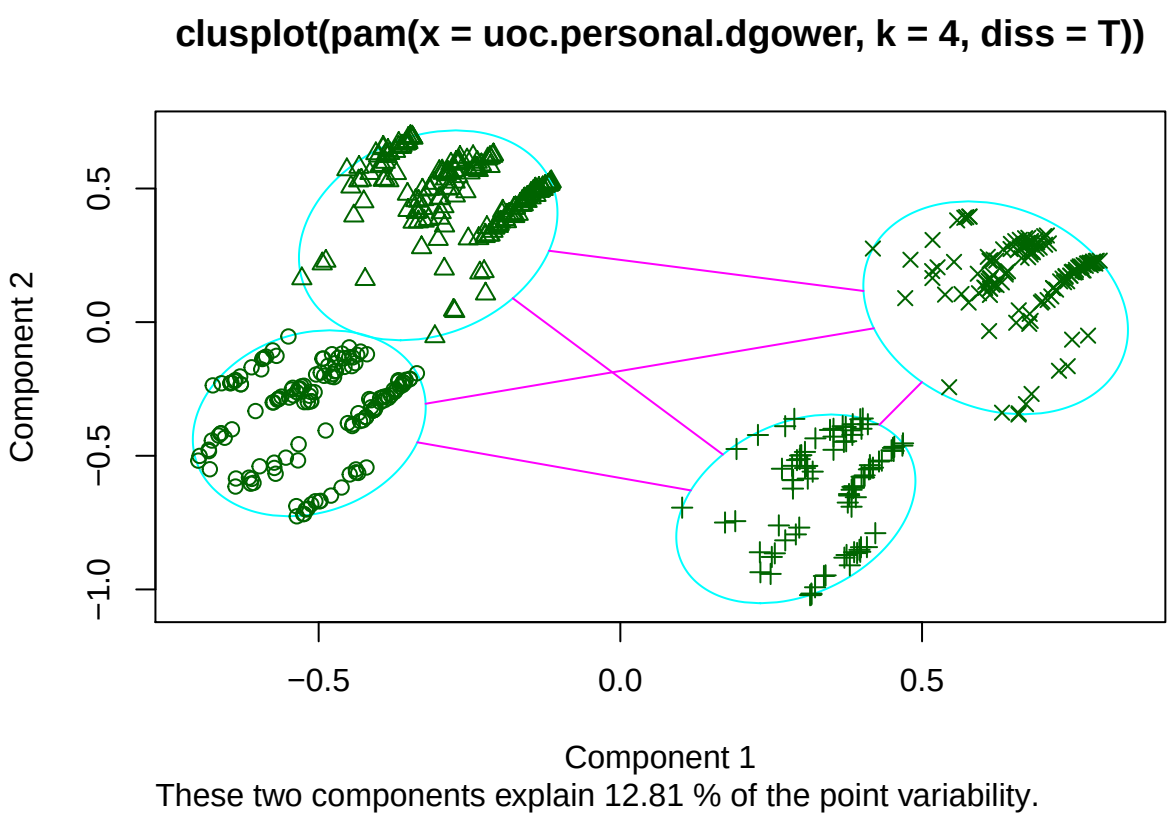

```
data.frame(uoc.personal[personal.pam$medoids,], tamaño=personal.pam$clusinfo[,1])
```

```
##     AGE GENDER DOMAIN   PhD YEARSEXP UOC_POSITION USERWIKI tamaño
## 336  42      M      6  TRUE       14            6    FALSE    150
## 149  43      M      6 FALSE        7            6    FALSE    168
## 699  41      F      6  TRUE       14            6    FALSE     89
## 530  38      F      6 FALSE        5            6    FALSE    135
```

```
personal.pam$isolation
```

```
##  1  2  3  4
## no no no no
## Levels: no L L*
```

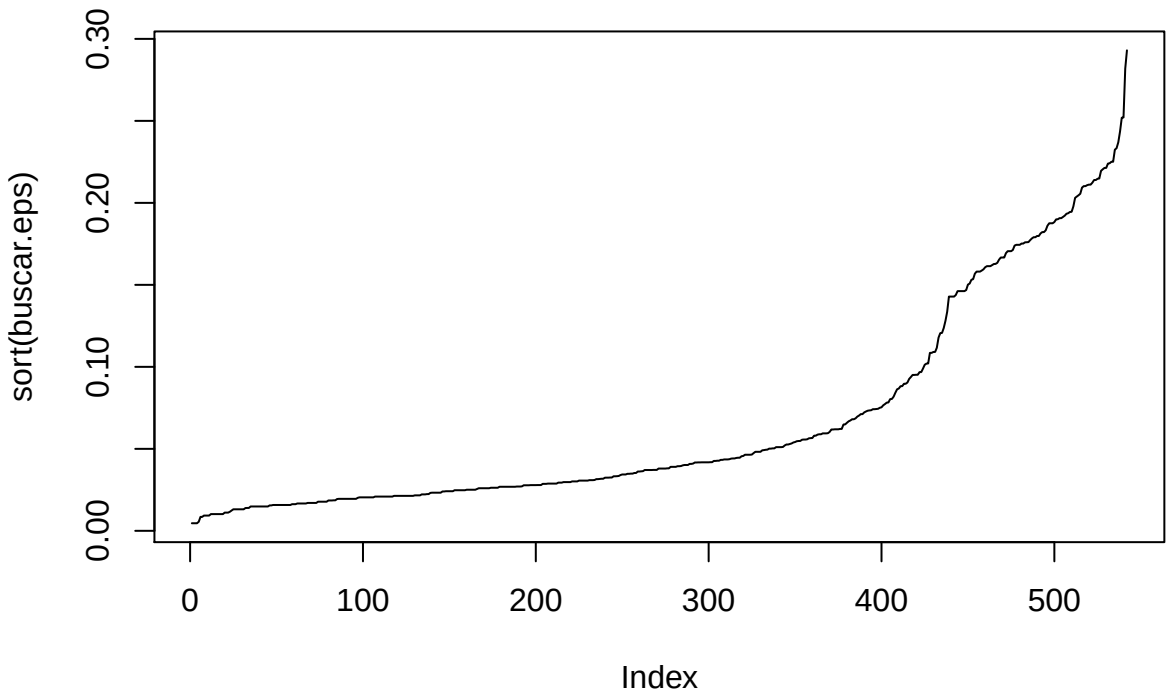
```
clusplot(personal.pam)
```



Clustering por densidad

Antes de realizar el paso de clustering tenemos que buscar el valor adecuado de *eps*. Para *minPts* usaremos el default de cinco.

```
buscar.eps <- apply(as.matrix(uoc.personal.dgower),1, function(x) sort(x)[5])
plot(sort(buscar.eps), type="l")
```



Y hacemos dos pruebas:

```
personal.dbs.1 <- dbscan(uoc.personal.dgower, eps=0.09)
personal.dbs.1
```

```
## dbscan Pts=542 MinPts=5 eps=0.09
##
##   0
## 542
```

```
personal.dbs.2 <- dbscan(uoc.personal.dgower, eps=0.15)
personal.dbs.2
```

```
## dbscan Pts=542 MinPts=5 eps=0.15
##          0 1 2 3 4
## border 519 0 4 2 4
## seed     0 6 1 5 1
## total  519 6 5 7 5
```

¿Qué pasó?

Clustering difuso

```r
personal.fuzz.1 <- fanny(uoc.personal.dgower, 4, diss = T, memb.exp = 1.35)
personal.fuzz.1$coeff
```

```
## dunn_coeff normalized
##  0.6141479  0.4855305
```

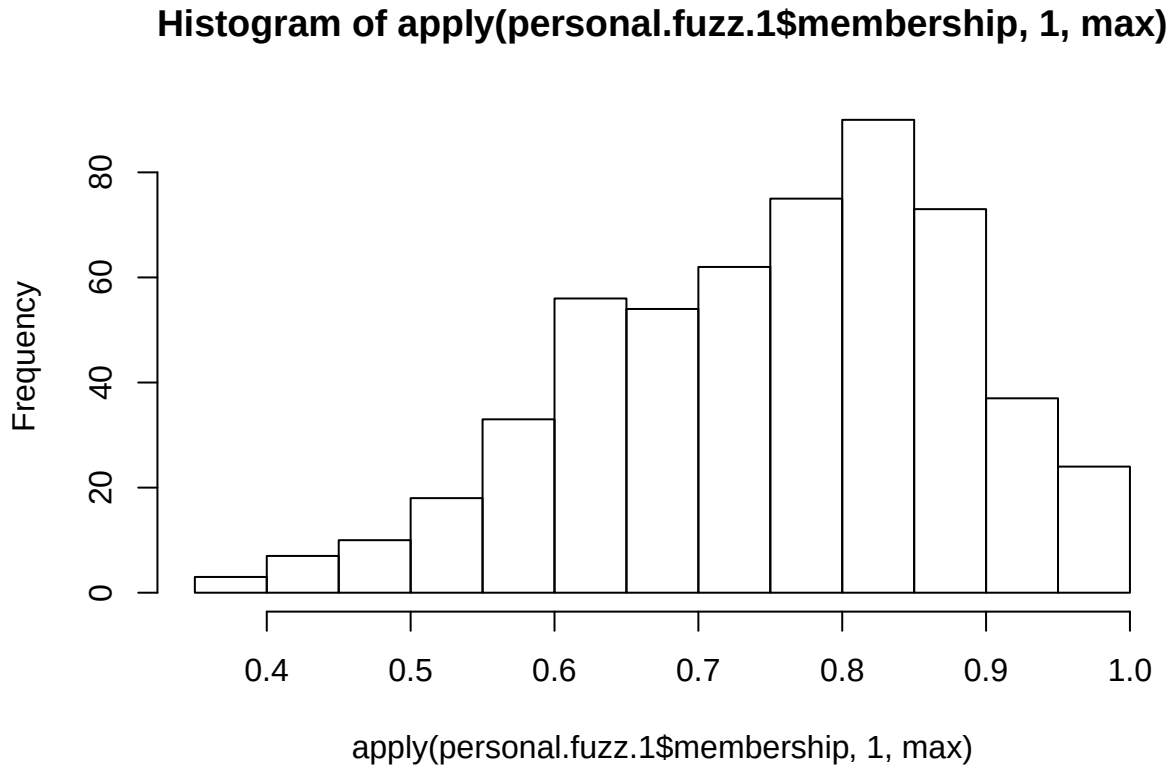
```r
head(personal.fuzz.1$membership)
```

```
##        [,1]      [,2]      [,3]       [,4]
## 1 0.6440614 0.1155567 0.1970013 0.04338063
## 2 0.6242916 0.1152986 0.2178497 0.04256019
## 3 0.6180924 0.1498581 0.1870518 0.04499762
## 4 0.2681075 0.5115003 0.1244104 0.09598186
## 5 0.2782778 0.3999537 0.1785020 0.14326655
## 6 0.2885315 0.4788763 0.1357635 0.09682872
```

```r
head(personal.fuzz.1$clustering, 10)
```

```
##  1  2  3  4  5  6  8  9 11 12
##  1  1  1  2  2  2  2  1  1  2
```

```r
hist(apply(personal.fuzz.1$membership,1, max))
```



```
fuzz.pers <- apply(personal.fuzz.1$membership,1, max) < 0.6
table(fuzz.pers)
```

```
## fuzz.pers
## FALSE  TRUE
##   471    71
```

```
fuzz.pers.col <- personal.fuzz.1$clustering
fuzz.pers.col[fuzz.pers] <- 0
```

Combinamos lo que acabamos de hacer con un ordenamiento hecho por NMDS. Primero con las asignaciones originales y la otra marcando en negro los "encuestados difusos".

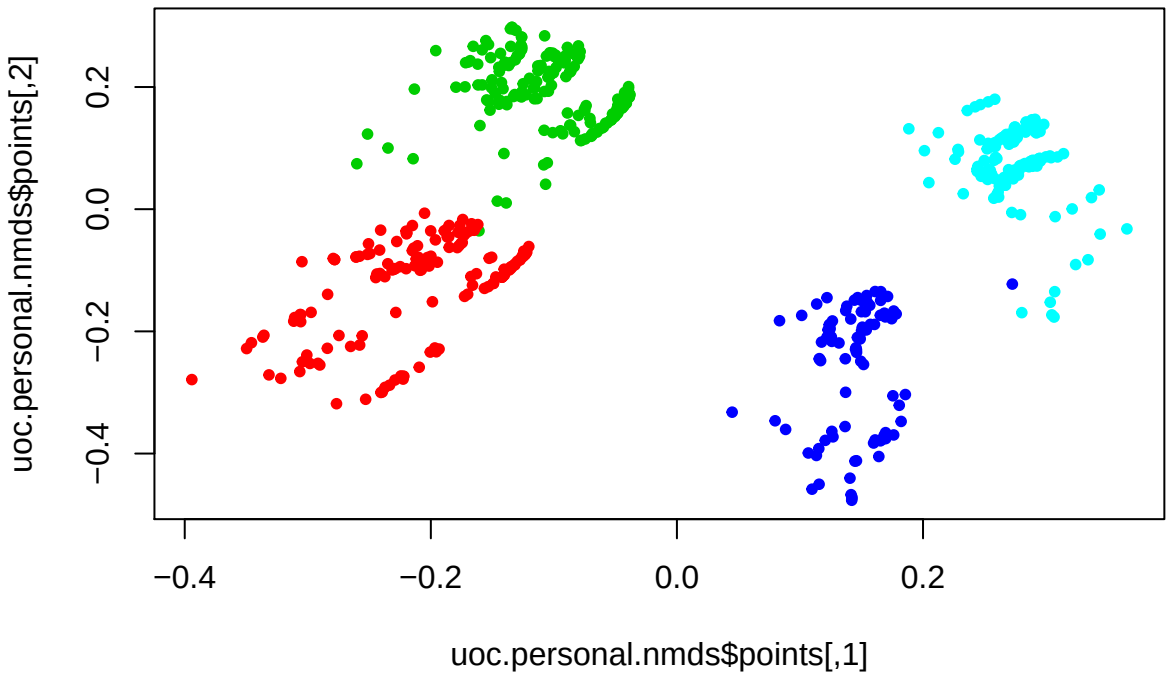
```
uoc.personal.nmds <- isoMDS(uoc.personal.dgower + 0.0001)
```

```
## initial  value 28.240080
## final  value 28.240080
## converged
```

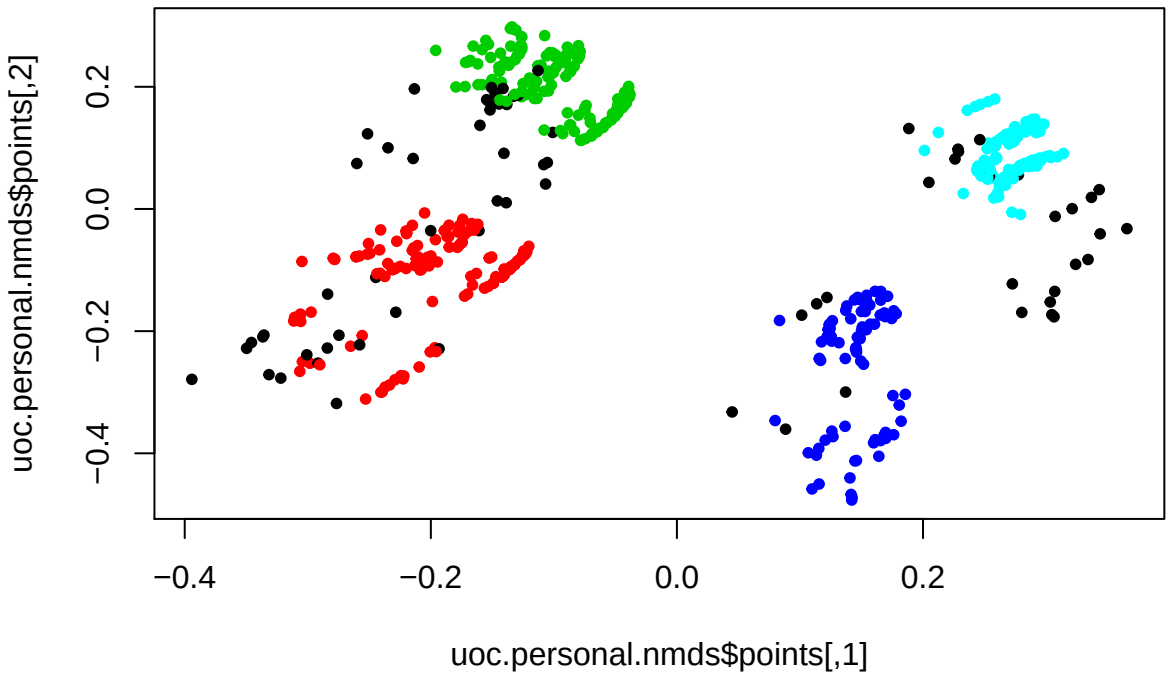
```
uoc.personal.nmds$stress
```

```
## [1] 28.24008
```

```
plot(uoc.personal.nmds$points, col=personal.fuzz.1$clustering+1, pch=20)
```



```
plot(uoc.personal.nmds$points, col=fuzz.pers.col+1, pch=20)
```



Análisis de las respuestas a las encuestas

Primero realicemos un cluster con el metodo PAM usando las distancias euclideas entre respuestas. Veamos, además, algunas características adicionales de la salida de la función *pam()*.

```
preguntas.pam <- pam(uoc.preguntas, 5, metric = "euclidean")
# Quienes son los medoides
preguntas.pam$id.med
```

```
## [1] 252 479 486 260 114
```

```
# como se agrupan los encuestados
head( preguntas.pam$clustering, 15)
```

```
##  1  2  3  4  5  6  8  9 11 12 13 15 17 18 19
##  1  1  2  3  4  1  4  5  5  2  4  4  4  1  4
```

```
# cual es el clustering que le corresponde a cada encuestado
head(preguntas.pam$medoids)
```

```
##     PU1 PU2 PU3 PEU1 PEU2 PEU3 ENJ1 ENJ2 Qu1 Qu2 Qu3 Qu4 Qu5 Vis1 Vis2
## 348   3   3   3    5    4    3    3    4   3   4   3   4   3    3    3
## 686   2   2   3    4    4    3    3    3   2   3   2   3   2    2    3
## 699   3   3   3    3    3    3    3    3   3   3   3   3   3    3    3
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 156   3   4   4    4    4    3    5    4   4   5   4   3   3    3    3
##     Vis3 Im1 Im2 Im3 SA1 SA2 SA3 Use1 Use2 Use3 Use4 Use5 Pf1 Pf2 Pf3 JR1
## 348    2   3   4   3   4   4   4    2    2    2    2    3   3   3   2   4
## 686    2   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3
## 699    3   3   3   3   3   3   3    3    3    3    3    3   3   3   3   3
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 156    3   3   4   4   4   4   5    3    2    4    4    4   2   2   2   4
##     JR2 BI1 BI2 Inc1 Inc2 Inc3 Inc4 Exp1 Exp2 Exp3 Exp4 Exp5
## 348   4   3   3    4    3    4    4    2    3    4    2    2
## 686   2   2   2    3    3    3    3    2    3    2    1    1
## 699   3   3   3    3    3    3    3    3    3    3    3    3
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 156   3   4   4    4    4    4    3    4    4    4    1    3
```

Extracción de datos

Las dos líneas de código que siguen son para ver y almacenar en un objeto de R los valores que toman los medoides de cada objeto clusterizado.

```
head( preguntas.pam$medoids[preguntas.pam$clustering,], 15)
```

```
##     PU1 PU2 PU3 PEU1 PEU2 PEU3 ENJ1 ENJ2 Qu1 Qu2 Qu3 Qu4 Qu5 Vis1 Vis2
## 348   3   3   3    5    4    3    3    4   3   4   3   4   3    3    3
## 348   3   3   3    5    4    3    3    4   3   4   3   4   3    3    3
## 686   2   2   3    4    4    3    3    3   2   3   2   3   2    2    3
## 699   3   3   3    3    3    3    3    3   3   3   3   3   3    3    3
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 348   3   3   3    5    4    3    3    4   3   4   3   4   3    3    3
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 156   3   4   4    4    4    3    5    4   4   5   4   3   3    3    3
## 156   3   4   4    4    4    3    5    4   4   5   4   3   3    3    3
## 686   2   2   3    4    4    3    3    3   2   3   2   3   2    2    3
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
## 348   3   3   3    5    4    3    3    4   3   4   3   4   3    3    3
## 358   4   4   5    4    4    4    4    4   3   4   3   2   4    4    4
##     Vis3 Im1 Im2 Im3 SA1 SA2 SA3 Use1 Use2 Use3 Use4 Use5 Pf1 Pf2 Pf3 JR1
## 348    2   3   4   3   4   4   4    2    2    2    2    3   3   3   2   4
## 348    2   3   4   3   4   4   4    2    2    2    2    3   3   3   2   4
## 686    2   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3
## 699    3   3   3   3   3   3   3    3    3    3    3    3   3   3   3   3
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 348    2   3   4   3   4   4   4    2    2    2    2    3   3   3   2   4
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 156    3   3   4   4   4   4   5    3    2    4    4    4   2   2   2   4
## 156    3   3   4   4   4   4   5    3    2    4    4    4   2   2   2   4
## 686    2   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3
```

```
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
## 348    2   3   4   3   4   4   4    2    2    2    2    3   3   3   2   4
## 358    3   3   4   4   5   5   5    4    3    4    4    4   4   4   5   4
##     JR2 BI1 BI2 Inc1 Inc2 Inc3 Inc4 Exp1 Exp2 Exp3 Exp4 Exp5
## 348   4   3   3    4    3    4    4    2    3    4    2    2
## 348   4   3   3    4    3    4    4    2    3    4    2    2
## 686   2   2   2    3    3    3    3    2    3    2    1    1
## 699   3   3   3    3    3    3    3    3    3    3    3    3
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 348   4   3   3    4    3    4    4    2    3    4    2    2
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 156   3   4   4    4    4    4    3    4    4    4    1    3
## 156   3   4   4    4    4    4    3    4    4    4    1    3
## 686   2   2   2    3    3    3    3    2    3    2    1    1
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 358   3   4   4    4    4    3    4    5    5    5    3    5
## 348   4   3   3    4    3    4    4    2    3    4    2    2
## 358   3   4   4    4    4    3    4    5    5    5    3    5
```

```
vec.meds <- preguntas.pam$medoids[preguntas.pam$clustering,]
```

A continuación buscaremos el mejor valor de K para agrupar. El loop en esencia es similar al que usamos antes para los datos personales, con algunas diferencias. Como argumento de la función *pam()* le estamos pasando una matriz de datos, no una matriz de distancia como hicimos antes, y especificamos que la distancia a calcular es euclidea y que mantenga la matriz de distancia resultante en la salida ($keep.dis = T$).

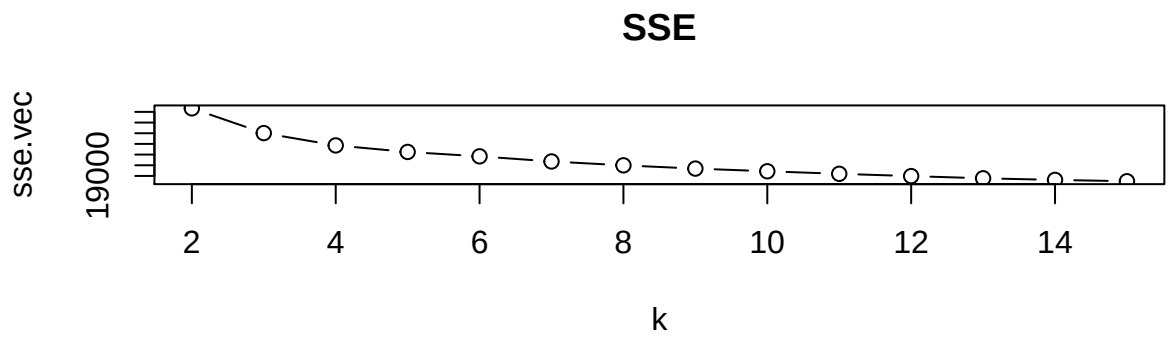
Como en este caso *pam()* "ve" los datos, en *medoids* no solo guarda el ID del prototipo, sino sus datos completos. Por lo tanto, para recuperar la distancia entre un objeto y su medoide necesitamos especificar que necesitamos el ID del medoide, por eso *vec.meds* tiene una asignación diferente de la que habíamos hecho antes para *pers.meds*:

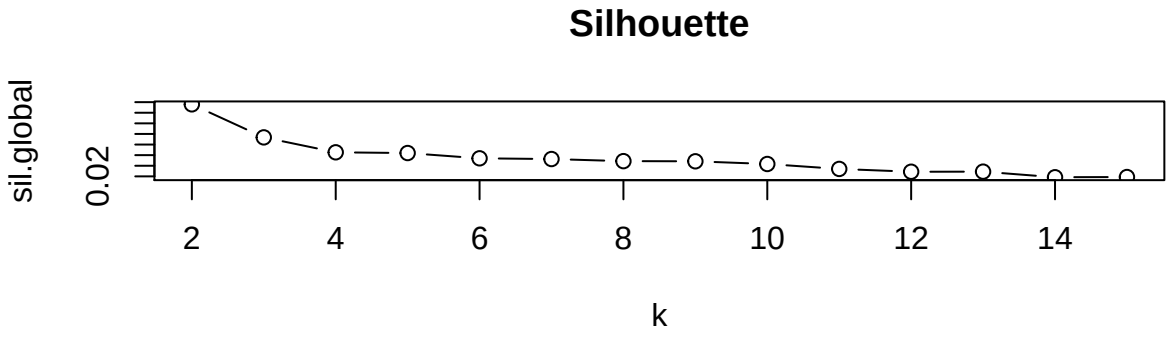
- vec.meds <- row.names(preguntas.pam$medoids)[preguntas.pam$clustering] *

Luego, la matriz de distancia es simplemente una propiedad del objeto que devuelve *pam()*:

- preguntas.pam$diss *

```
sse.vec <- array()
sil.global <- array()
for(i in 1:kit){
  preguntas.pam <- pam(uoc.preguntas, i+1, metric = "euclidean", keep.diss = T)
  vec.meds <- row.names(preguntas.pam$medoids)[preguntas.pam$clustering]
  sse.vec[i] <- sum(as.matrix(preguntas.pam$diss)[cbind(row.names(uoc.personal),vec.meds)]^2)
  sil.global[i] <- preguntas.pam$silinfo$avg.width
}


par(mfrow=c(2,1))
plot(2:(kit+1), sse.vec, xlab="k", type="b", main="SSE")
plot(2:(kit+1), sil.global, xlab="k", type="b", main="Silhouette")
```






```
par(mfrow=c(1,1))
```

El k óptimo es dos según Silhouette, según SSe no es tan claro, pero también se ubicaría entre 2 o 3.

Una función de distancia para variables categóricas ordenadas.

Las respuestas están codificadas con cinco valores enteros ordenados. En consecuencia, en lugar de la distancia euclídea, podríamos usar alguna de distancia para variables categóricas ordenadas, como la que vimos en la térocia de medidas de (di)similitud. Para normalizar las distancias entre 0 y 1, vamos a tener en cuenta que el máximo valor de disimilitud es la máxima diferencia entre respuestas, 5-1 = 4. Por ejemplo, para la distancia entre dos encuestados cualquiera:

```
sum(abs(uoc.preguntas[1,] - uoc.preguntas[2,])) / (ncol(uoc.preguntas)*4)
```

```
## [1] 0.2151163
```

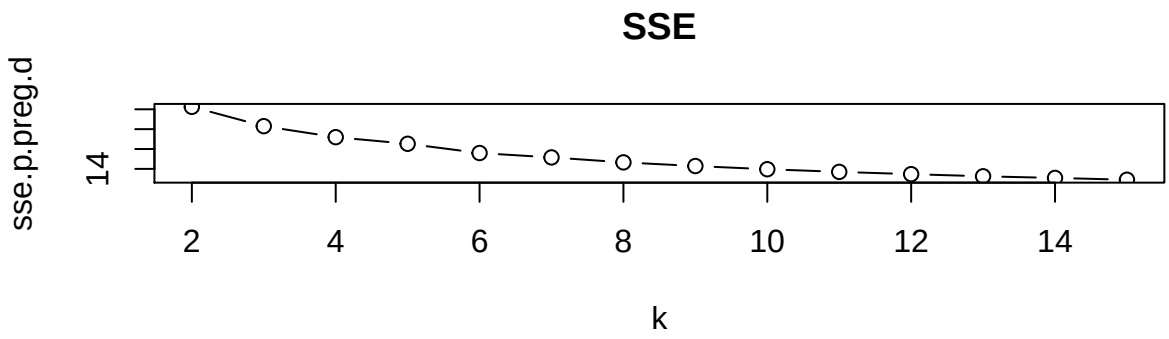
Vamos a repetir esto para todos los encuestados para crear la matriz de distancias.

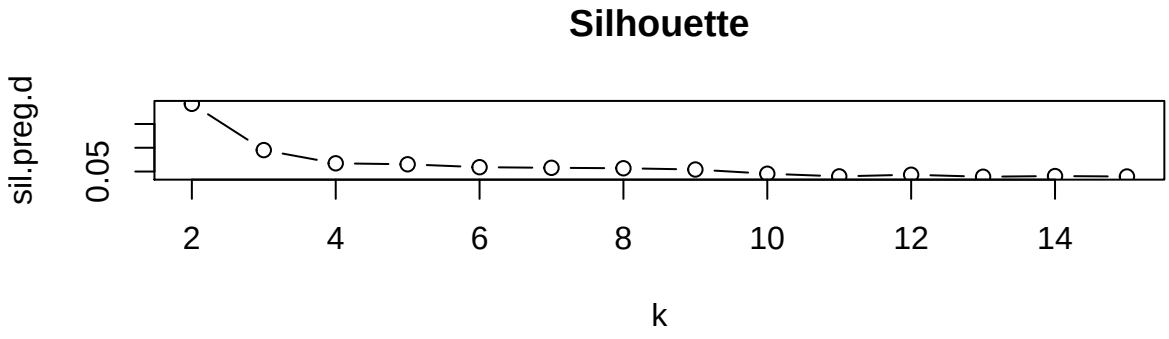
```
dist.enc <- matrix(NA, nrow(uoc.preguntas), nrow(uoc.preguntas))
min.dis <- ncol(uoc.preguntas)*4
mat.dat <- as.matrix(uoc.preguntas)
# El loop que sigue se podría acelerar teniendo en cuenta que el resultado
# es una matriz singular, pero para el tamaño que tiene, no haría falta
```

```
for(i in 1:nrow(mat.dat)){
  for(j in 1:nrow(mat.dat)){
    dist.enc[i, j] <- sum(abs(mat.dat[i,] - mat.dat[j,])) / min.dis
  }
}
row.names(dist.enc) <- row.names(mat.dat)
dist.enc <- as.dist(dist.enc)
```

Y repetimos la misma actividad de antes, buscamos el k óptimo.

```
sse.p.preg.d <- array()
sil.preg.d <- array()
for(i in 1:kit){
  preguntas.d.pam <- pam(dist.enc, i+1, diss = T)
  vec.meds.d <- preguntas.d.pam$medoids[preguntas.d.pam$clustering]
  sse.p.preg.d[i] <- sum(as.matrix(dist.enc)[cbind(row.names(uoc.preguntas), vec.meds.d)]^2)

  sil.preg.d[i] <- preguntas.d.pam$silinfo$avg.width
}

par(mfrow=c(2,1))
plot(2:(kit+1), sse.p.preg.d, xlab="k", type="b", main="SSE")
plot(2:(kit+1), sil.preg.d, xlab="k", type="b", main="Silhouette")
```





```r
par(mfrow=c(1,1))
```

En este caso, el análisis con Silhouette indica más fuertemente usar un k=2, y con SSE, como pasó antes, no se ve un claro ganador. Probemos con k=2.

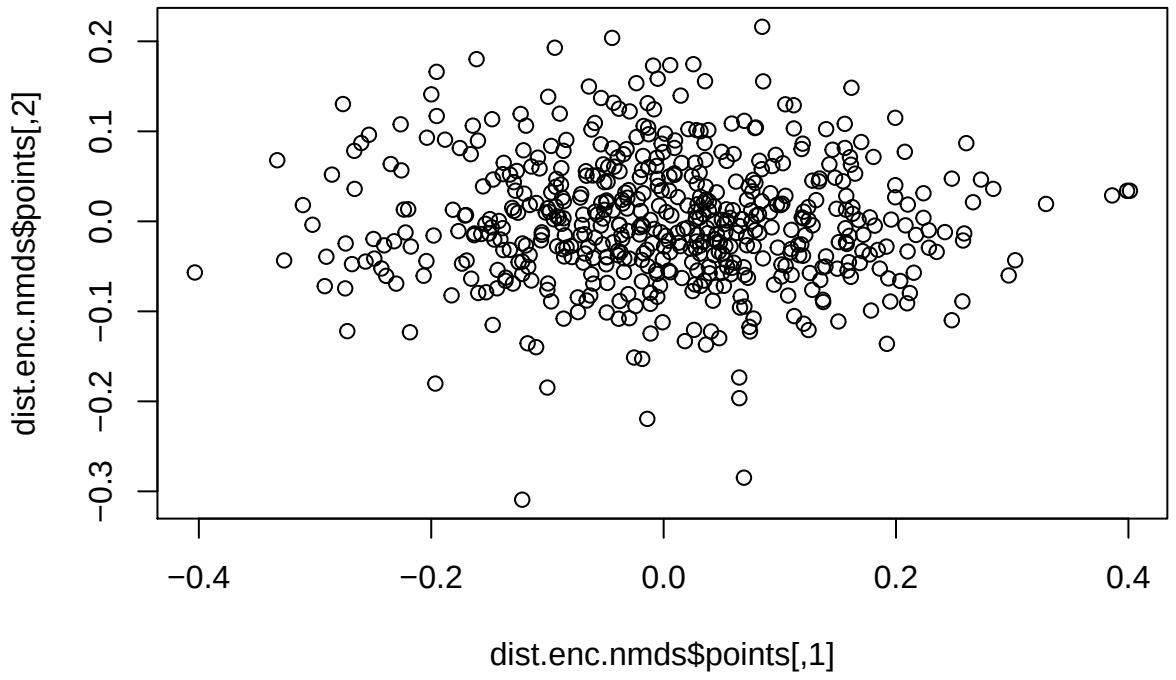
```r
preguntas.d.pam <- pam(dist.enc, 2, diss = T)

dist.enc.nmds <- isoMDS(dist.enc + 0.0001)
```

```
## initial  value 26.070010
## iter   5 value 19.945440
## final  value 19.724695
## converged
```

```r
dist.enc.nmds$stress
```

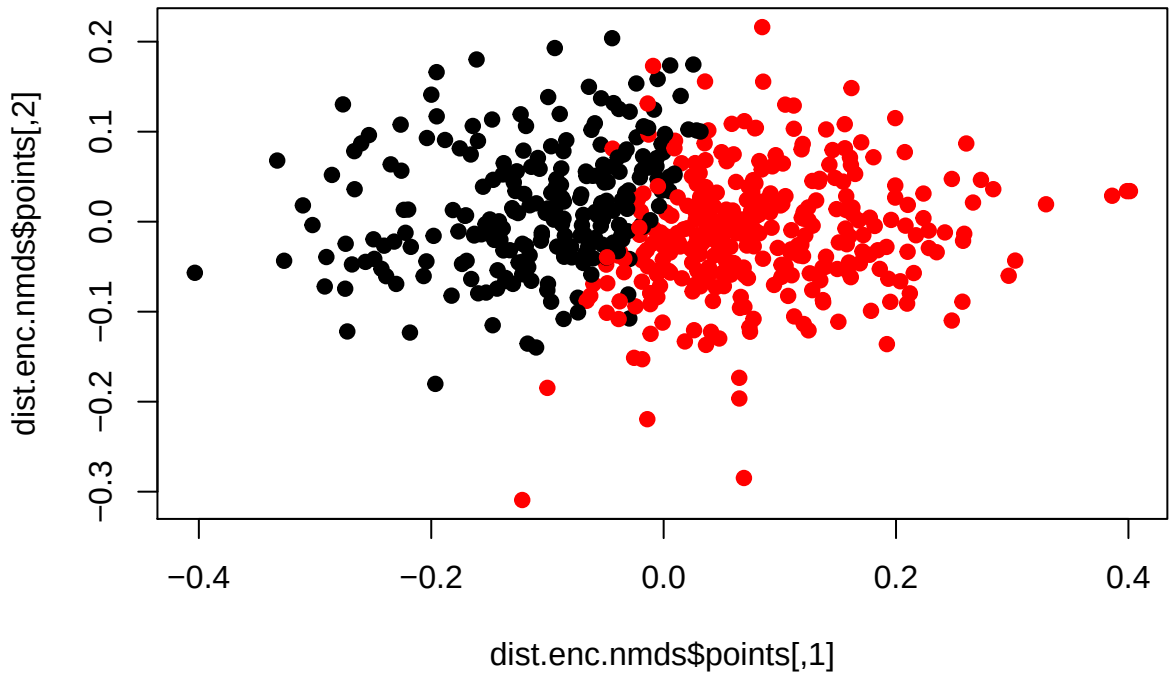
```
## [1] 19.72469
```

```r
plot(dist.enc.nmds$points)
```



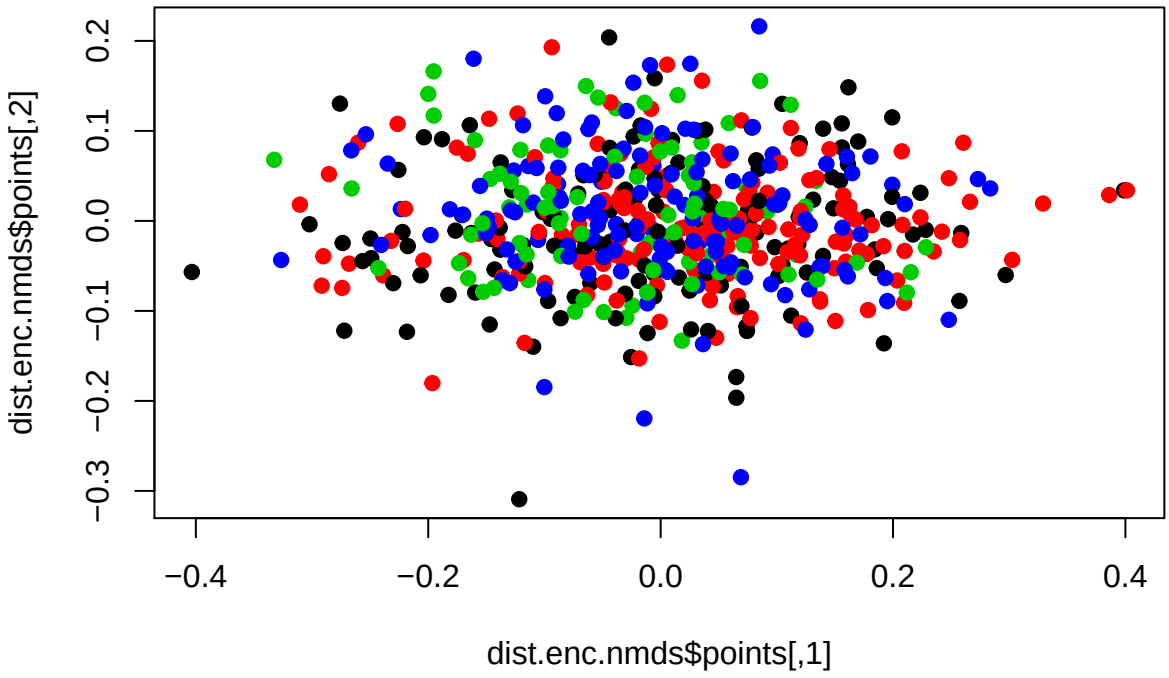
¿Cómo se agrupan los encuestados según sus respuestas?

```
plot(dist.enc.nmds$points, col=preguntas.d.pam$clustering, pch=19)
```



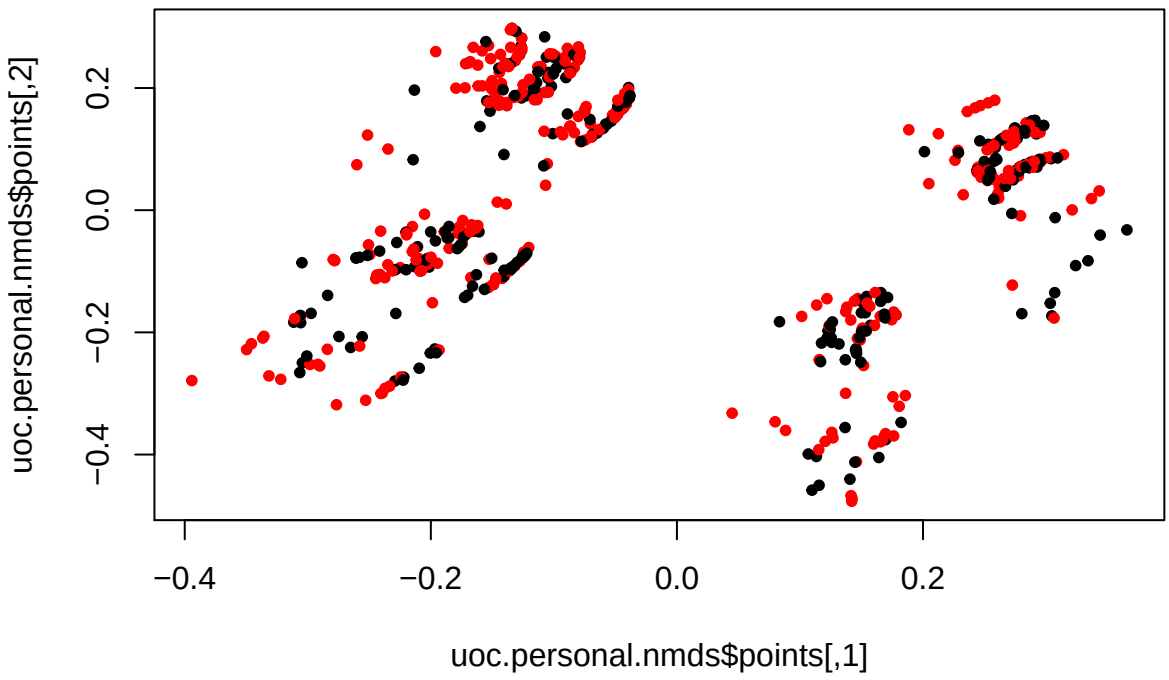
Y según sus características demográficas y profesionales ¿Cómo se distribuyen sobre la nube de respuestas?

```
plot(dist.enc.nmds$points, col=personal.pam$clustering, pch=19)
```



Como una visualización complementaria podemos ver cómo se distribuyen según sus respuestas en el clustering por características deomgráficas y personales.

```
plot(uoc.personal.nmds$points, col=preguntas.d.pam$clustering, pch=20)
```



Finalmente podemos hacer una matriz de confusión entre ambos agrupamientos.

```
table(personal.pam$clustering, preguntas.d.pam$clustering, dnn=c("grupo demográfico", "grupo de respues
```