Laboratorio de clustering

Marcelo Soria - Juan Kamienkowski

DM CyT, 2018

Para esta practica de laboratorio vamos a usar un dataset que se puede descargar desde aqui. Son datos de una encuesta entre profesores de dos universidades españolas sobre uso de Wikipedia como recurso educativo.

Se pude leer información adicional de este trabajo aquí

Primero cargamos algunos paquetes que vamos a necesitar.

```
library(cluster)
library(MASS)
# install.packages("fpc")
library(fpc)
# install.packages("dplyr")
library(dplyr)
# install.packages("ggplot2")
library(ggplot2)
# install.packages("stringr")
library(stringr)
```

Descargamos los datos desde su url y los asignamos a un objeto R.

```
encuesta <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/00334/wiki4HE.csv",hea
```

Luego hacemos algunos chequeos.

```
dim(encuesta)
str(encuesta[1:10])
```

- ¿Qué información nos dan los comandos anteriores?
- ¿Cómo están indicados los datos faltantes?

Limpieza y preparación de los datos

Tratamiento de datos faltantes.

El resultado de la función *str()* en el paso anterior nos mostró que todas las variables que tienen datos faltantes aparecen como variables de tipo carácter, pero deberían ser numéricas, o en algunos casos de tipo factor. Entonces, para el tratamiento de variables con datos faltantes hay que reemplazar el carácter "?" por *NA* y luego convertir el tipo de la variable a numérica. Pero antes de hacer esto, calculamos los recuentos de datos faltantes por variable para mantener un registro de sus frecuencias.

```
contar_na <- function(x) length(str_which(x, "\\?"))
recuento_na <- sapply(encuesta, contar_na)
sort(recuento_na, decreasing = T)
```

```
##    OTHERSTATUS OTHER_POSITION          Vis2   UOC_POSITION          PEU3
##            540            261           117            113            97
##           Vis1            Im3           JR2            BI2          Inc4
##             72             57            53             43            42
##           Inc3           Inc1          Inc2            BI1           Qu5
##             37             35            35             32            29
```

```
##            JR1       YEARSEXP           Use4            Qu4            Im1
##             27             23             23             22             22
##            Im2           ENJ2           Use2            Qu3           Use5
##             20             17             17             15             15
##           PEU2           Use1            Pf3           Exp4           Exp1
##             14             14             14             14             13
##           Exp3           Exp5            SA2            PU2            SA1
##             13             13             12             11             11
##            SA3            Pf1           Exp2            Qu2           Use3
##             11             11             11             10              9
##           Vis3            PU1           ENJ1            Qu1            Pf2
##              8              7              7              7              6
##            PU3      USERWIKI           PEU1         DOMAIN            AGE
##              5              4              4              2              0
##         GENDER            PhD     UNIVERSITY
##              0              0              0
```

Ahora hacemos la sustitución y conversión:

```
convertir_na <- function(x, na_symbol = "?"){
  if(typeof(x) == "character"){
    x[ x == na_symbol ] <- NA
  }
  return(as.numeric(x))
}

encuesta_2 <- as.data.frame( sapply(encuesta, convertir_na) )
```

Conversiones de tipos

Las siguientes variables deberían ser de tipo factor: GENDER, DOMAIN, UOC_POSITION, UNIVERSITY, OTHER_POSITION y OTHERSTATUS. Estas conversiones son importantes para evitar más adelante calcular, por ejemplo, distancias Euclídeas entre sexos o entre dominios de trabajo. En algunos casos, y para aumentar la claridad vamos a especificar los niveles de estas variables.

```
encuesta_2$GENDER <- factor( ifelse(encuesta_2$GENDER == 1, "F", "M") )

domain_labels <- c("Arts_Humanities", "Sciences",  "Health_Sciences","Engineering_Architecture", "Law",
encuesta_2$DOMAIN <- factor(encuesta_2$DOMAIN, labels = domain_labels  )

pos_labels =  c("Professor", "Associate", "Assistant", "Lecturer", "Instructor", "Adjunct")
encuesta_2$UOC_POSITION <- factor(encuesta_2$UOC_POSITION, labels = pos_labels)

encuesta_2$OTHERSTATUS <- factor(encuesta_2$OTHERSTATUS)
# Queda codificado como númenro porque los nombres de categorías
# no coinciden con el número de categorías

encuesta_2$UNIVERSITY <- factor( ifelse(encuesta_2$UNIVERSITY == 1, "UOC", "UPF"))
```

Hay tres variables que deberían estar codificadas como de tipo lógico, PhD, USERWIKI y OTHER_POSITION. La variable OTHER_POSITION tiene una particularidad, el valor 1 indica que la persona es docente en UOC y docente part-time en otra universidad, y el valor 2 que no tiene otra posición part-time. Esta variable tiene datos faltantes, por lo que no se puede usar la función *ifelse* de R base como hicimos con GENDER o PhD. En este caso usamos la versión más estricta *if_else()* del paquete *dplyr* (revisar la documentación)

```
encuesta_2$PhD <- as.logical(encuesta_2$PhD)
encuesta_2$USERWIKI <- as.logical(as.numeric(encuesta_2$USERWIKI))
encuesta_2$OTHER_POSITION <- if_else(encuesta_2$OTHER_POSITION == 1, TRUE, FALSE, NA)
```

Para el trabajo que sigue solo vamos a trabajar con los datos de UOC, que es la universidad que más respuestas tiene. Después de este paso, la variable UNIIVERSITY no la precisamos más.

```
encuesta_uoc <- encuesta_2 %>% filter(UNIVERSITY == "UOC") %>% select(-UNIVERSITY)
```

Hay datos faltantes en más de la mitad de los registros:

```
table(complete.cases(encuesta_uoc))
```

```
##
## FALSE  TRUE
##   624   176
```

Casi todas las variables tienen datos faltantes:

```
table(sapply(encuesta_uoc, anyNA))
```

```
##
## FALSE  TRUE
##     4    48
```

Sin embargo, la mayoría de los datos faltantes se concentran en unas pocas variables. Esto lo habíamos visto anteriomente, al crear el vector *recuento_na*. Para simplificar el análisis no vamos a aplicar técnicas de imputación de datos faltantes, pero haremos algunos cambios en el dataset.

La variable OTHER_POSITION sólo tiene sentido para los docentes que tienen otra posición además de la que tienen en UOC. Las preguntas Vis2 y Peu3 solo deberían ser respondidos por quienes editan artículos en Wikipedia, o conocen a alguien que lo haga. Podemos eliminar estas variables.

```
encuesta_uoc$OTHER_POSITION <- NULL
encuesta_uoc$Vis2 <- NULL
encuesta_uoc$PEU3 <- NULL

table(complete.cases(encuesta_uoc))
```

```
##
## FALSE  TRUE
##   612   188
```

En una situación de trabajo real, habría que continuar el análisis de los datos faltantes, y considerar aplicar alguna ténica de imputación de datos faltantes. Nosotros nos vamos a quedar con los registros completos, y luego vamos a separar los datos profesionales y demográficos de los encuestados.

```
encuesta_uoc_c <- encuesta_uoc[complete.cases(encuesta_uoc), ]
uoc_personal <- encuesta_uoc_c[,1:7]
uoc_preguntas <- encuesta_uoc_c[, 8:49]
```

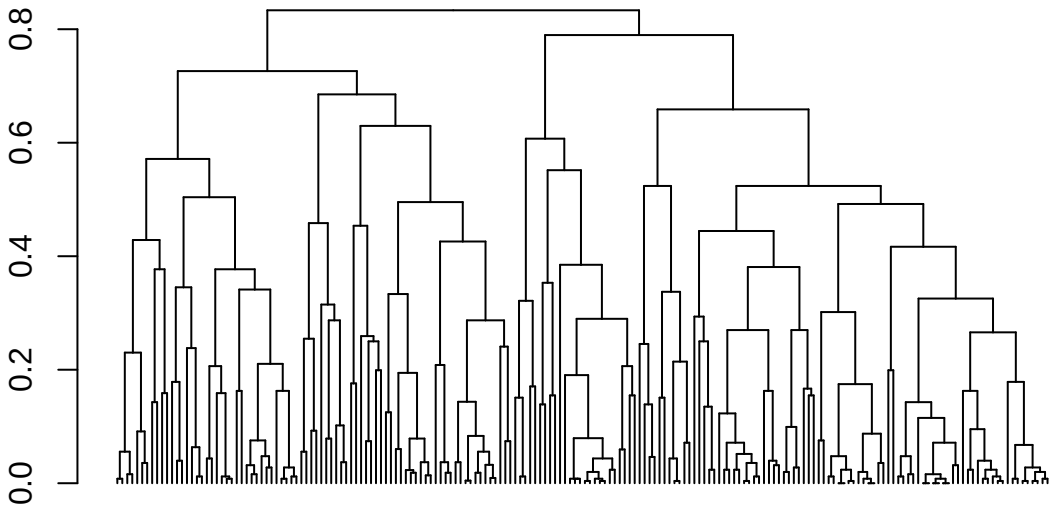
Análisis

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

```
uoc_personal_dgower <- daisy(uoc_personal, metric="gower")
```

```
## Warning in daisy(uoc_personal, metric = "gower"): setting 'logical'
## variable 4 to type 'asymm'
```

```
plot(as.dendrogram(hclust(uoc_personal_dgower)), leaflab="none")
```

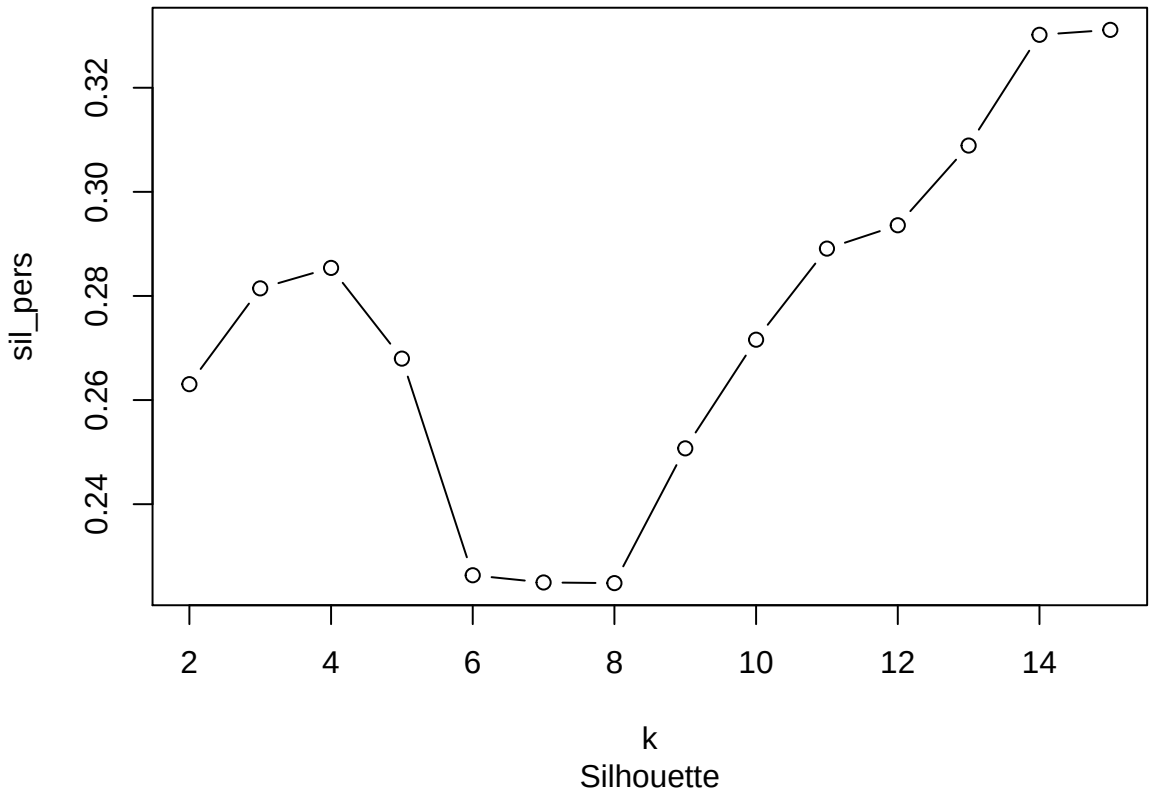


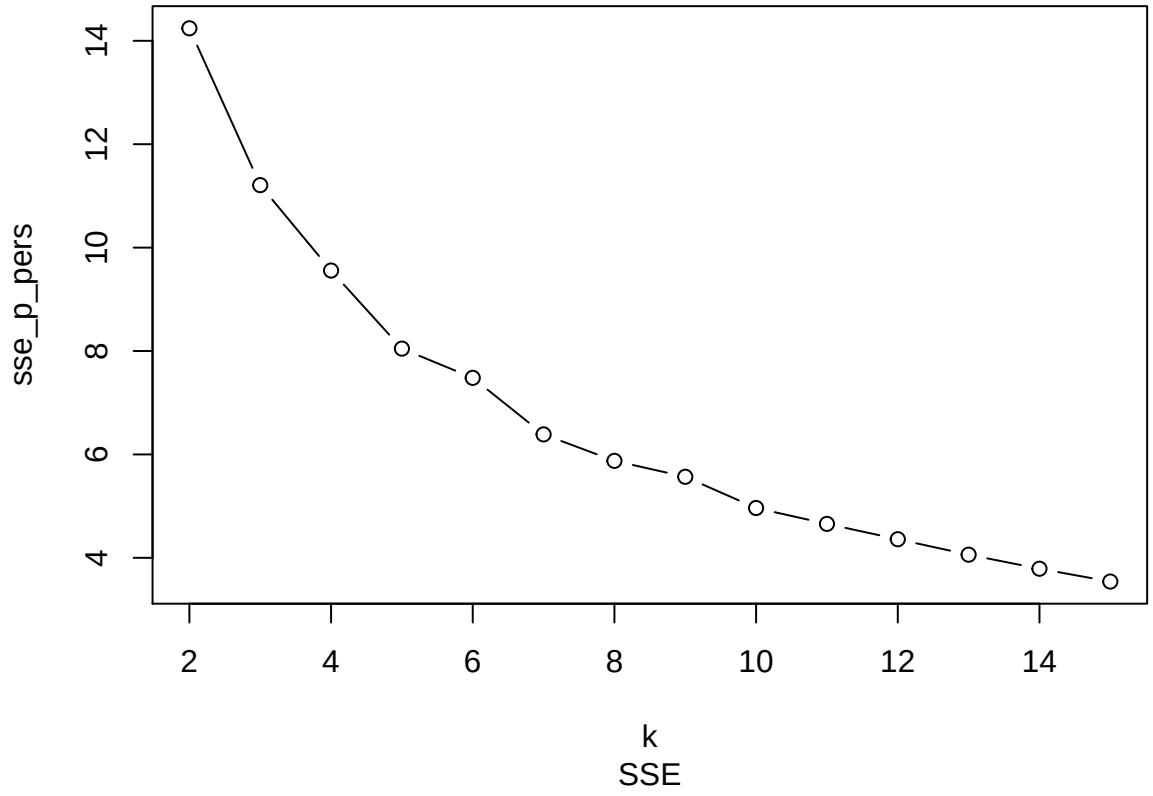
- ¿Qué se puede decir de la presencia de grupos en el dataset?

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.

En el loop de más abajo se recorren los valores de k desde 2 hasta el máximo número de k (cantidad de clusters) que se van a probar. En cada iteración se calcula un nuevo PAM y con sus medoides se calcula el SSE y Silhouette.

```
sse_p_pers <- array()
sil_pers <- array()
kit <- 14
for(i in 1:kit){
  # Cálculo de PAM:
  personal_pam <- pam(uoc_personal_dgower, i+1, diss = T)
  # Determinar el ID del medoide que le corresponde a cada registro:
  pers_meds <- personal_pam$medoids[personal_pam$clustering]
  # Cálculo de SSEs: construir un vector que registre las distancias entre
  # cada objeto y su correspondiente medoide elevadas al cuadrado, y luego
  # calcular su suma. Almacenar cada SSE en un vector.
  sse_p_pers[i] <- sum(as.matrix(uoc_personal_dgower)[cbind(row.names(uoc_personal), pers_meds)]^2)
  # Almacenar cada valor de silhouette global
  sil_pers[i] <- personal_pam$silinfo$avg.width
}

par(mfrow=c(2,1))
plot(2:(kit+1), sil_pers, type="b", xlab="k", sub="Silhouette")
plot(2:(kit+1), sse_p_pers, type="b", xlab="k", sub = "SSE")
```


sil_pers
0.24
0.26
0.28
0.30
0.32
2
4
6
8
10
12
14
k
Silhouette

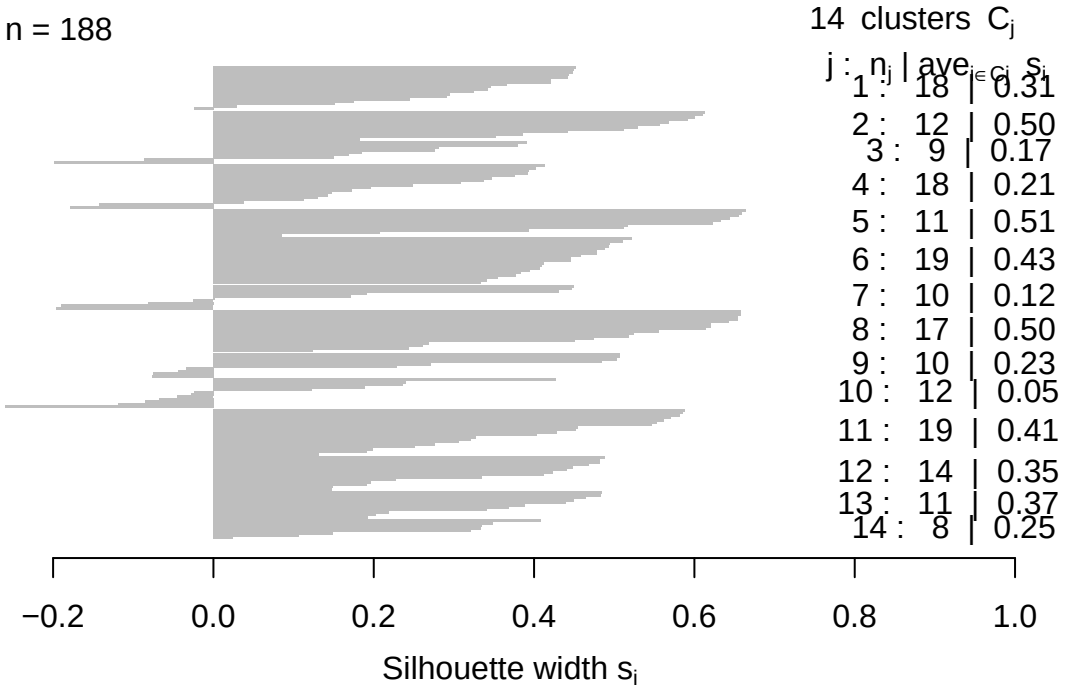

sse_p_pers
4
6
8
10
12
14
2
4
6
8
10
12
14
k
SSE

```
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
## 439  42      M       Political_Sciences FALSE        8      Adjunct
## 309  44      M                      Law  TRUE       16      Adjunct
## 242  40      M          Arts_Humanities  TRUE       10      Adjunct
## 352  37      M       Political_Sciences  TRUE        7      Adjunct
## 214  48      M                Sciences  TRUE       20      Adjunct
## 412  44      M Engineering_Architecture  TRUE       22      Adjunct
## 303  44      M                      Law FALSE       12      Adjunct
## 336  42      M       Political_Sciences  TRUE       14      Adjunct
## 587  36      F       Political_Sciences  TRUE       10      Adjunct
## 569  47      F          Arts_Humanities FALSE        5      Adjunct
## 557  44      F       Political_Sciences  TRUE       20      Adjunct
## 654  38      F       Political_Sciences FALSE        7      Adjunct
## 733  33      F       Political_Sciences  TRUE        7      Adjunct
## 629  42      F                      Law  TRUE       17      Adjunct
```

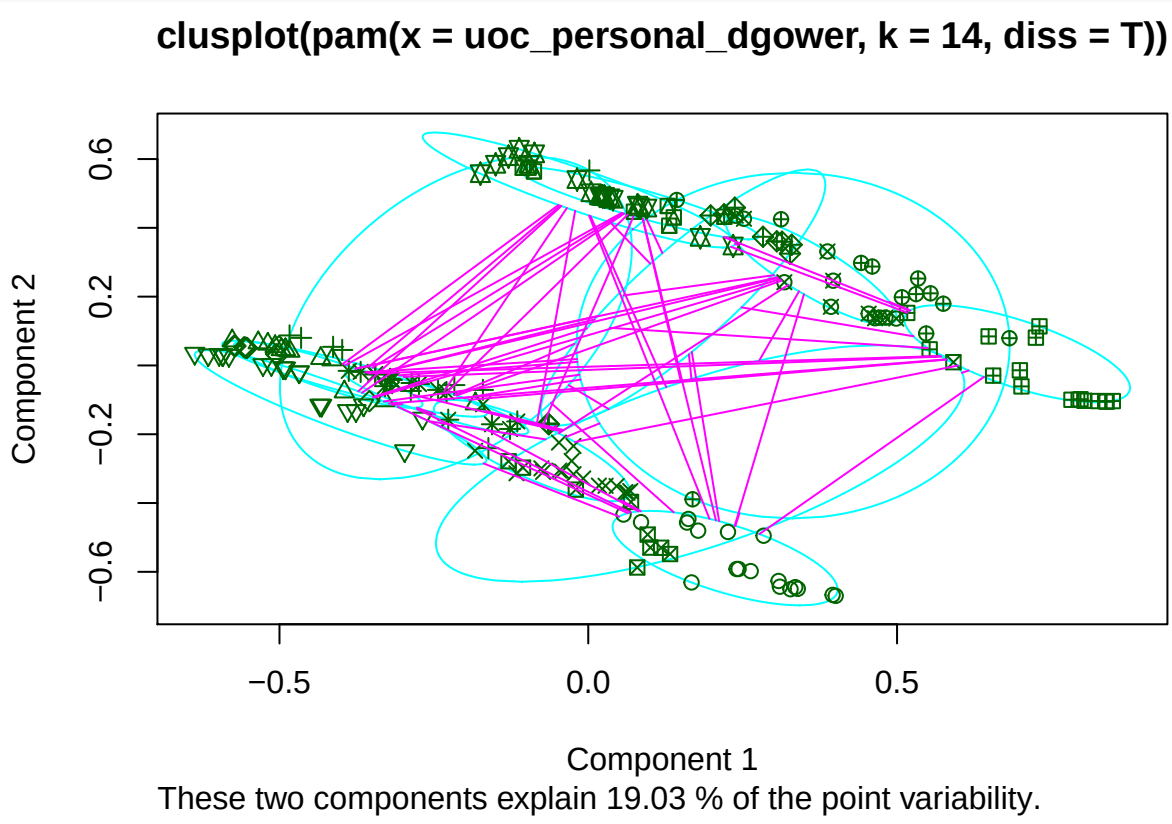
```
##     OTHERSTATUS tamaño
## 439           7     18
## 309           2     12
## 242           2      9
## 352           7     18
## 214           2     11
## 412           2     19
## 303           7     10
## 336           2     17
## 587           4     10
## 569           6     12
## 557           2     19
## 654           7     14
## 733           7     11
## 629           2      8
```

```
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*
```

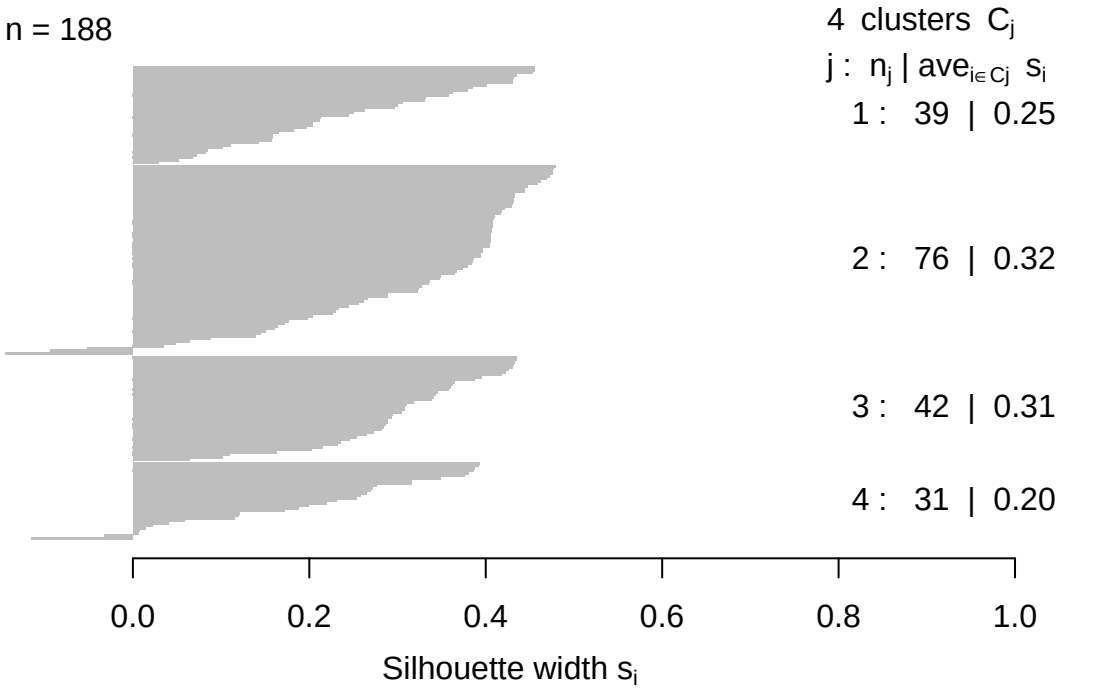
Y un gráfico para ver la relación entre grupos e individuos.

```
clusplot(personal_pam)
```



Para este ejercicio nos quedamos con k=4, que no es necesariamente el mejor valor.

```
# Probamos k=4
personal_pam <- pam(uoc_personal_dgower, 4, diss = T)
plot(silhouette(personal_pam))
```



Average silhouette width : 0.29

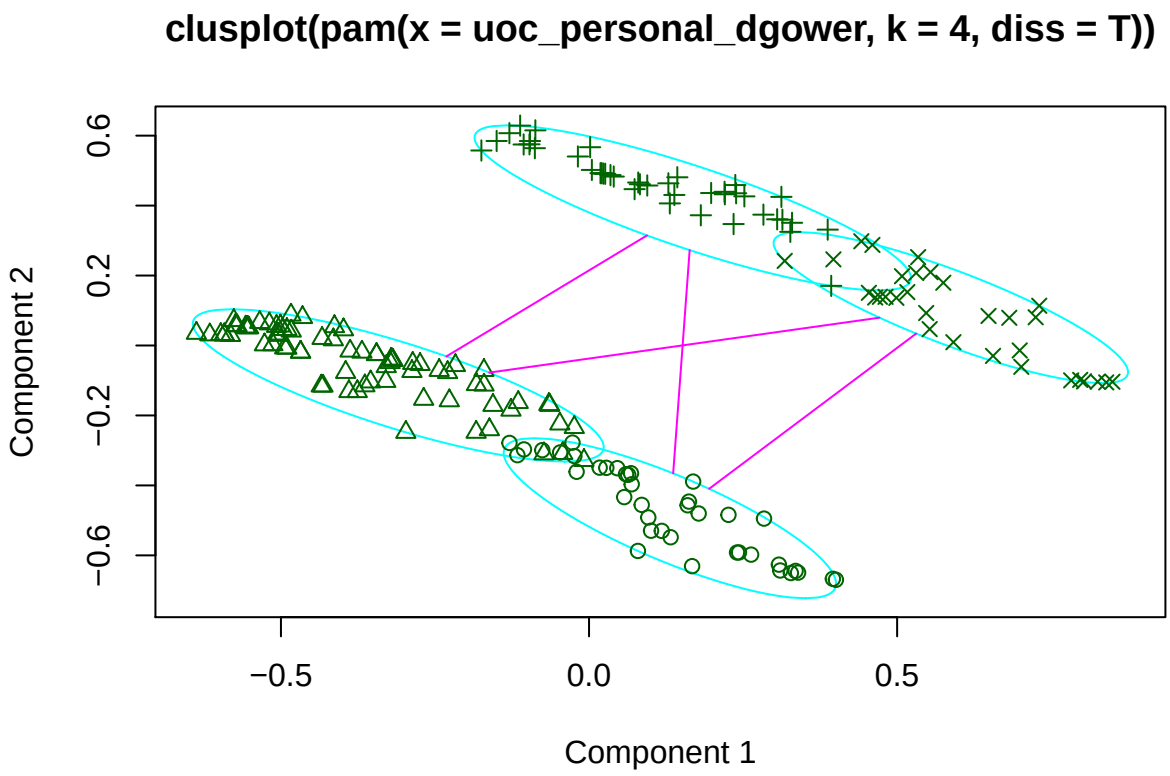
```
data.frame(uoc_personal[personal_pam$medoids,], tamaño=personal_pam$clusinfo[,1])
```

```
##     AGE GENDER            DOMAIN   PhD YEARSEXP UOC_POSITION OTHERSTATUS
## 439  42      M Political_Sciences FALSE        8      Adjunct           7
## 212  42      M Political_Sciences  TRUE       18      Adjunct           2
## 535  42      F Political_Sciences  TRUE       18      Adjunct           2
## 654  38      F Political_Sciences FALSE        7      Adjunct           7
##     tamaño
## 439     39
## 212     76
## 535     42
## 654     31
```

```
personal_pam$isolation
```

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

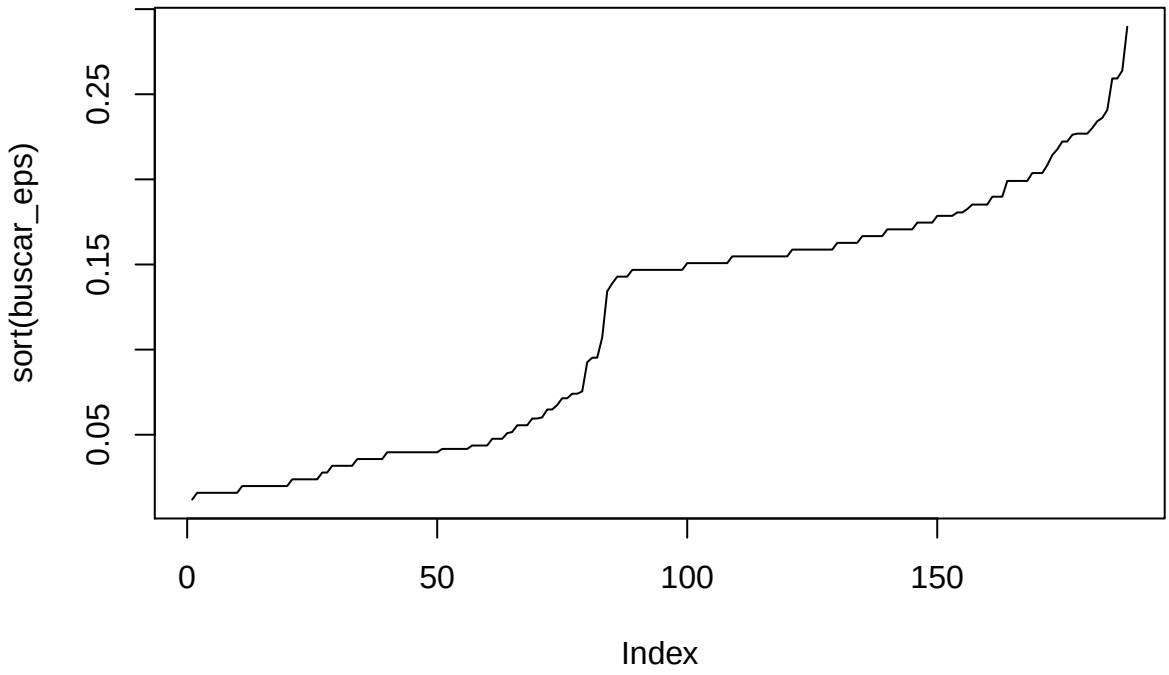
```
clusplot(personal_pam)
```



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. Como ya tenemos una matriz de distancia calculada, simplemente para cada registro recuperamos la quinta distancia a los otros.

```
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=188 MinPts=5 eps=0.09
##
##   0
## 188
```

```
personal_dbs_2 <- dbscan(uoc_personal_dgower, eps=0.15)
personal_dbs_2
```

```
## dbscan Pts=188 MinPts=5 eps=0.15
##          0 1
## border 183 4
## seed     0 1
## total  183 5
```

¿Qué pasó?

Clustering difuso

Vamos a realizar un clustering difuso, con la misma matriz de distancia de Gower que venimos trabajando y cuatro grupos.

```
personal_fuzz_1 <- fanny(uoc_personal_dgower, 4, diss = T, memb.exp = 1.35)
#Coeficiente de Dunnet
personal_fuzz_1$coeff
```

```
## dunn_coeff normalized
##  0.5843586  0.4458115
```

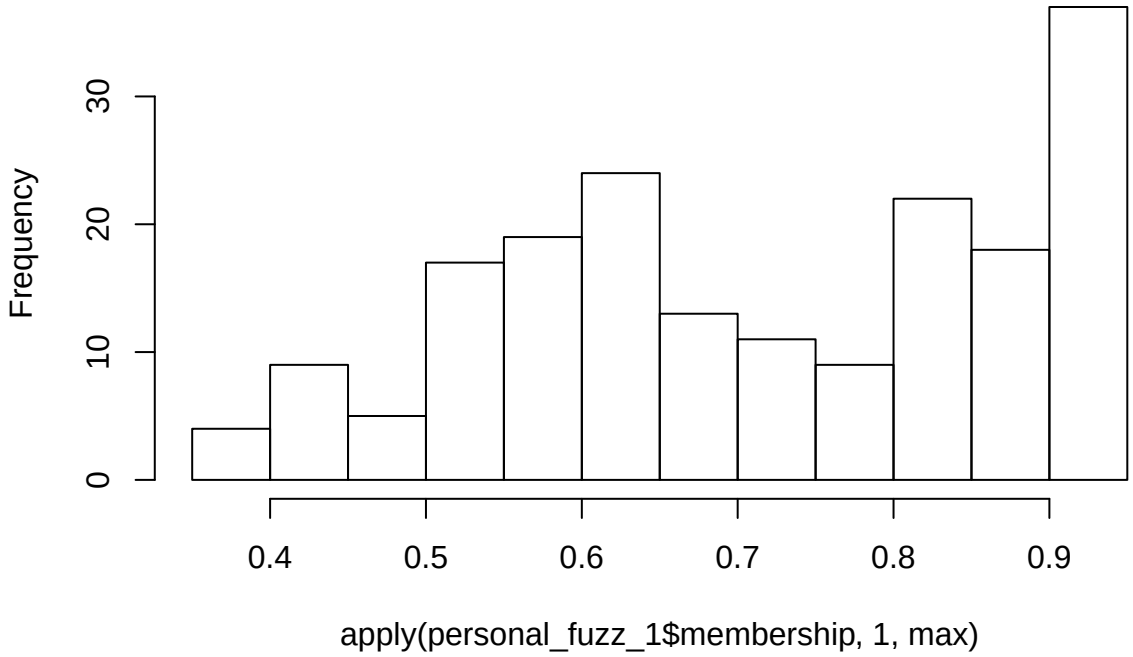
```
# Membresías (matriz y grupo con mayor puntaje)
head(personal_fuzz_1$membership)
```

```
##           [,1]       [,2]       [,3]        [,4]
## 149 0.86262035 0.02117432 0.01493931 0.101266026
## 150 0.61613934 0.12114234 0.08130937 0.181408943
## 161 0.03641430 0.91008890 0.04438446 0.009112332
## 165 0.05040485 0.86912521 0.06541696 0.015052982
## 166 0.82543955 0.04017642 0.02629972 0.108084305
## 167 0.89536282 0.05187947 0.02498703 0.027770680
```

```
head(personal_fuzz_1$clustering, 10)
```

```
## 149 150 161 165 166 167 168 173 175 179
##   1   1   2   2   1   1   2   2   2   1
```

```
# Distribución de las máximas membresías de cada registro:
hist(apply(personal_fuzz_1$membership,1, max), main="")
```




```
# ¿Cuántos registros tienen una membresía menor que 0.6?
fuzz_pers <- apply(personal_fuzz_1$membership,1, max) < 0.6
table(fuzz_pers)
```

```
## fuzz_pers
## FALSE  TRUE
##   134    54
```

```
# A los registros con unamembresía menor a 0.6
# los asignamos a un cluster "0", que corresponde a los
# que no agrupan claramente
fuzz_pers_col <- personal_fuzz_1$clustering
fuzz_pers_col[fuzz_pers] <- 0
```

- ¿Qué indica el coeficente de Dunnet?
- ¿Cuátos registros (docentes) se pueden asignar claramente a un grupo?

Combinamos lo que acabamos de hacer con un ordenamiento hecho por escalamiento métrico no dimensional (NMDS). Primero con las asignaciones originales y la otra marcando en negro los “encuestados difusos”.

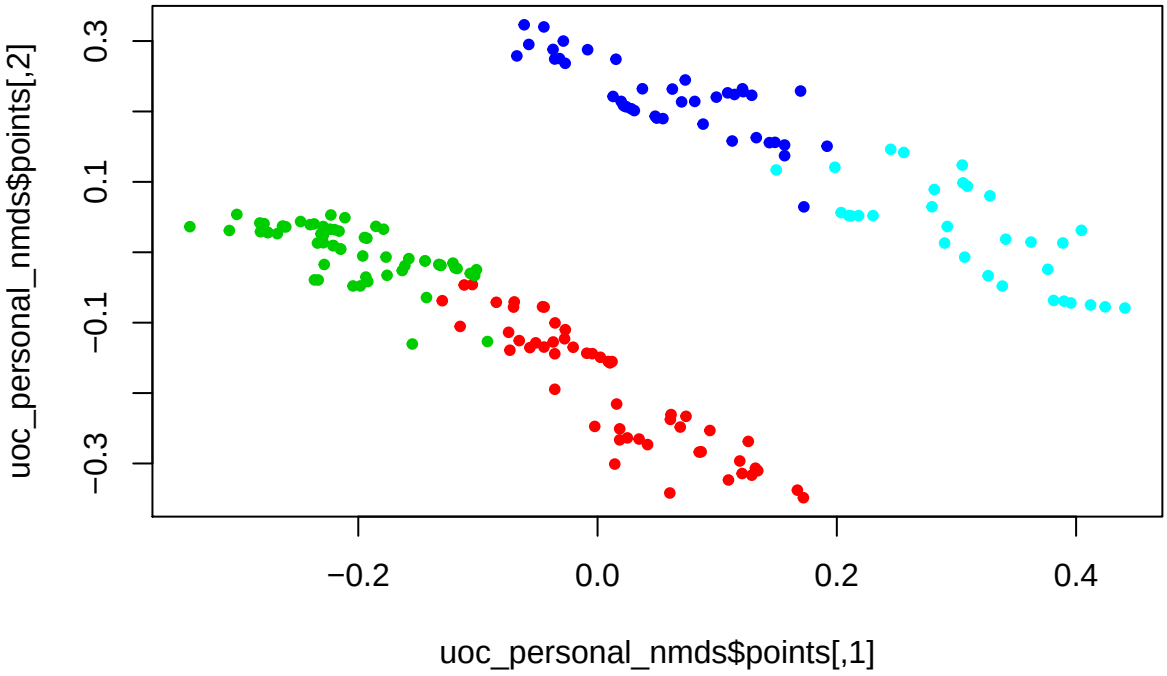
```
uoc_personal_nmds <- isoMDS(uoc_personal_dgower + 0.0001)
```

```
## initial  value 28.725952
## final  value 28.725952
## converged
```

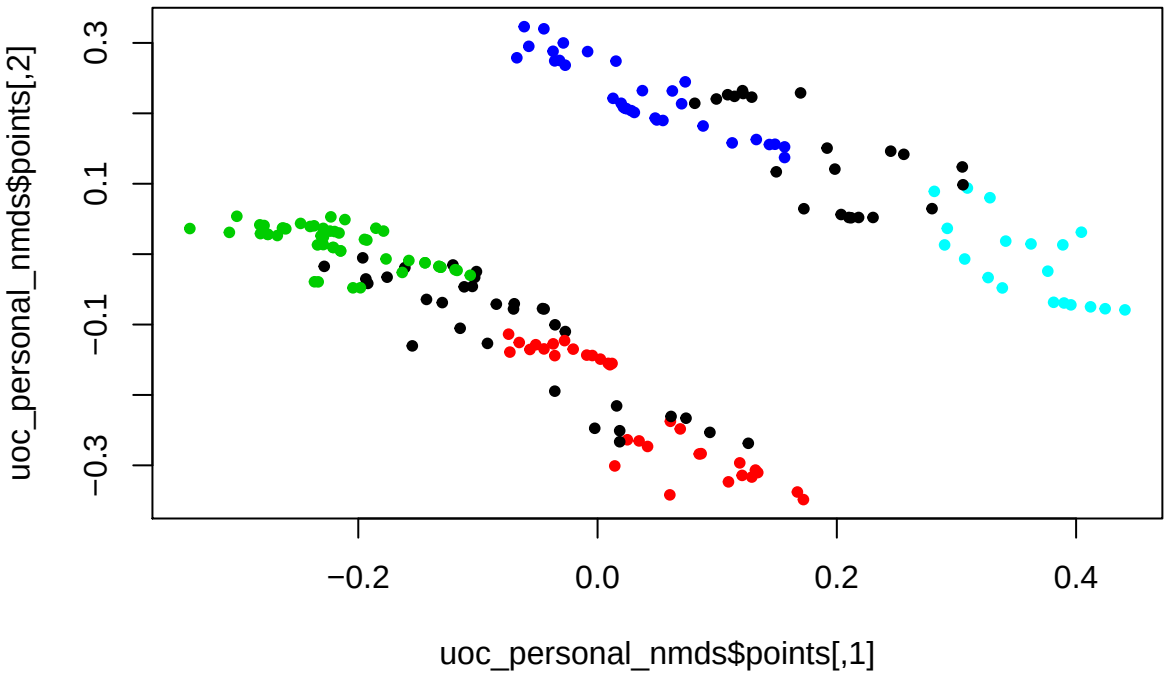
```
uoc_personal_nmds$stress
```

```
## [1] 28.72595
```

```
plot(uoc_personal_nmds$points, col=personal_fuzz_1$clustering+1, pch=20)
```




```
#Cuidado: Si el ID de un cluster es cero, R no le va asignar color
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 método 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]  79 111 159  19  72
```

```
# como se agrupan los encuestados
head( preguntas_pam$clustering, 15)
```

```
## 149 150 161 165 166 167 168 173 175 179 181 182 188 191 192
##   1   2   3   3   4   5   1   3   5   4   4   4   4   2   4
```

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

```
##     USERWIKI PU1 PU2 PU3 PEU1 PEU2 ENJ1 ENJ2 Qu1 Qu2 Qu3 Qu4 Qu5 Vis1 Vis3
## 409        0   4   4   4    4    4    4    4   4   4   4   4   4    4    4
## 508        0   3   3   3    4    4    3    4   2   3   2   3   2    2    1
## 686        0   2   2   3    4    4    3    3   2   3   2   3   2    2    2
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 382        0   3   3   4    5    5    4    4   4   4   3   3   4    3    1
##     Im1 Im2 Im3 SA1 SA2 SA3 Use1 Use2 Use3 Use4 Use5 Pf1 Pf2 Pf3 JR1 JR2
## 409   4   4   4   4   4   4    4    4    4    4    4   4   4   4   4   4
## 508   2   3   2   4   4   4    1    1    2    2    3   1   3   1   3   2
## 686   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3   2
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 382   3   4   3   5   5   4    2    1    4    3    4   1   3   2   3   3
##     BI1 BI2 Inc1 Inc2 Inc3 Inc4 Exp1 Exp2 Exp3 Exp4 Exp5
```

```
## 409   4   4    4    4    4    4    4    4    4    4    4
## 508   2   2    4    4    3    4    2    3    4    1    1
## 686   2   2    3    3    3    3    2    3    2    1    1
## 210   3   3    3    3    3    3    3    4    4    2    3
## 382   4   3    4    4    4    3    4    4    5    1    1
```

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)
```

```
##     USERWIKI PU1 PU2 PU3 PEU1 PEU2 ENJ1 ENJ2 Qu1 Qu2 Qu3 Qu4 Qu5 Vis1 Vis3
## 409        0   4   4   4    4    4    4    4   4   4   4   4   4    4    4
## 508        0   3   3   3    4    4    3    4   2   3   2   3   2    2    1
## 686        0   2   2   3    4    4    3    3   2   3   2   3   2    2    2
## 686        0   2   2   3    4    4    3    3   2   3   2   3   2    2    2
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 382        0   3   3   4    5    5    4    4   4   4   3   3   4    3    1
## 409        0   4   4   4    4    4    4    4   4   4   4   4   4    4    4
## 686        0   2   2   3    4    4    3    3   2   3   2   3   2    2    2
## 382        0   3   3   4    5    5    4    4   4   4   3   3   4    3    1
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
## 508        0   3   3   3    4    4    3    4   2   3   2   3   2    2    1
## 210        0   4   4   4    5    4    5    5   3   4   4   3   3    3    2
##     Im1 Im2 Im3 SA1 SA2 SA3 Use1 Use2 Use3 Use4 Use5 Pf1 Pf2 Pf3 JR1 JR2
## 409   4   4   4   4   4   4    4    4    4    4    4   4   4   4   4   4
## 508   2   3   2   4   4   4    1    1    2    2    3   1   3   1   3   2
## 686   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3   2
## 686   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3   2
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 382   3   4   3   5   5   4    2    1    4    3    4   1   3   2   3   3
## 409   4   4   4   4   4   4    4    4    4    4    4   4   4   4   4   4
## 686   1   3   2   4   3   4    2    1    2    2    3   1   3   2   3   2
## 382   3   4   3   5   5   4    2    1    4    3    4   1   3   2   3   3
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
## 508   2   3   2   4   4   4    1    1    2    2    3   1   3   1   3   2
## 210   3   3   4   4   4   5    2    2    3    3    4   3   3   3   3   3
##     BI1 BI2 Inc1 Inc2 Inc3 Inc4 Exp1 Exp2 Exp3 Exp4 Exp5
## 409   4   4    4    4    4    4    4    4    4    4    4
## 508   2   2    4    4    3    4    2    3    4    1    1
## 686   2   2    3    3    3    3    2    3    2    1    1
## 686   2   2    3    3    3    3    2    3    2    1    1
## 210   3   3    3    3    3    3    3    4    4    2    3
## 382   4   3    4    4    4    3    4    4    5    1    1
## 409   4   4    4    4    4    4    4    4    4    4    4
## 686   2   2    3    3    3    3    2    3    2    1    1
```

```
## 382   4   3    4    4    4    3    4    4    5    1    1
## 210   3   3    3    3    3    3    3    4    4    2    3
## 210   3   3    3    3    3    3    3    4    4    2    3
## 210   3   3    3    3    3    3    3    4    4    2    3
## 210   3   3    3    3    3    3    3    4    4    2    3
## 508   2   2    4    4    3    4    2    3    4    1    1
## 210   3   3    3    3    3    3    3    4    4    2    3
```

```
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 ($keep.dis = T$).

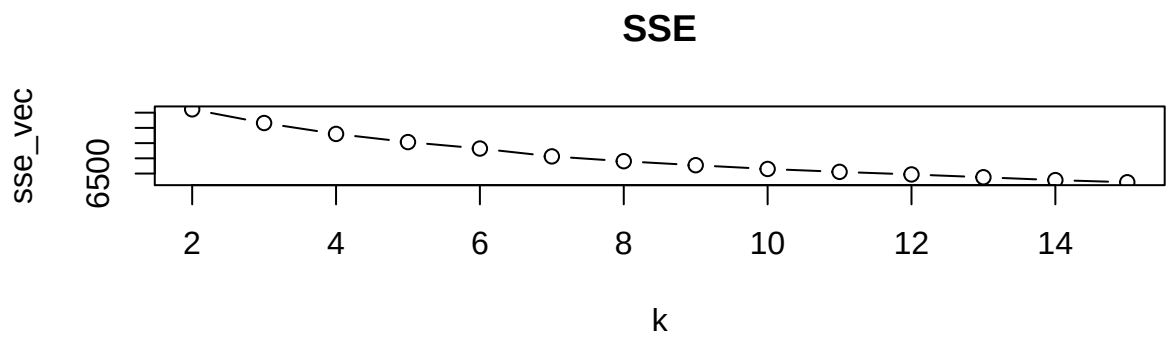
Como en este caso *pam()* "ve" los datos, en *medoids* no solo guarda el ID del prototipo, sino sus datos completos. Por lo tanto, para recuperar la distancia entre un objeto y su medoide necesitamos especificar que necesitamos el ID del medoide, por eso *vec.meds* tiene una asignación diferente de la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

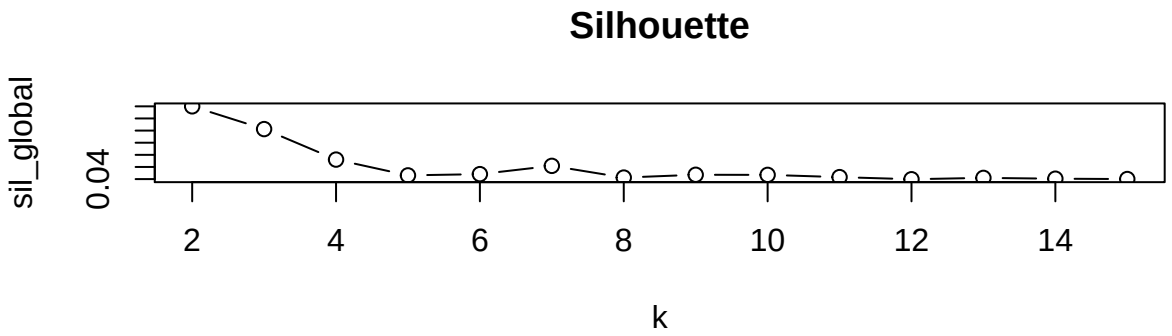
- vec_meds <- row.names(preguntas_pam$medoids)[preguntas_pam$clustering] *

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)
```

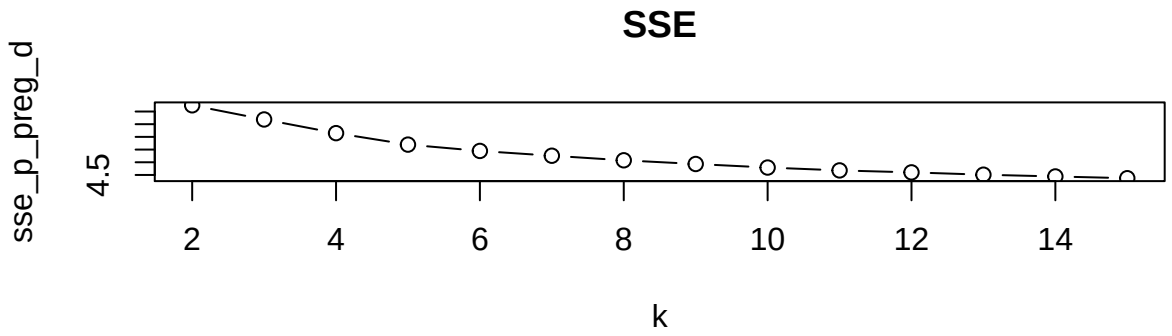
```
## [1] 0.3988095
```

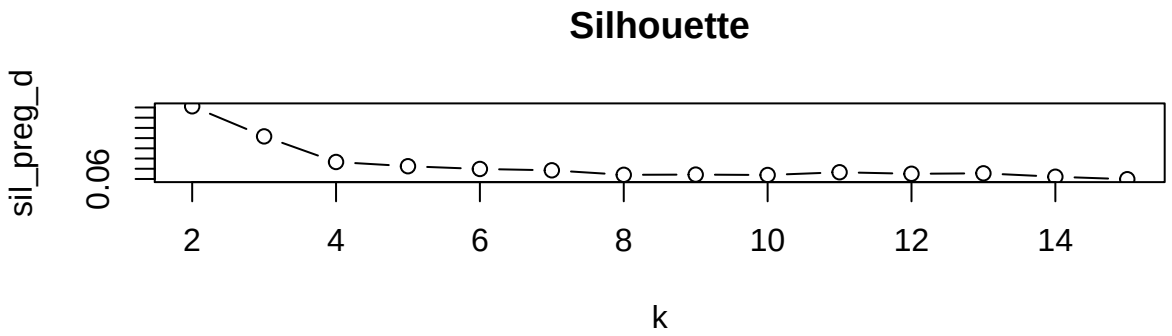
Vamos a repetir esto para todos los encuestados para crear la matriz de distancias.

```
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")
```






```
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=3.

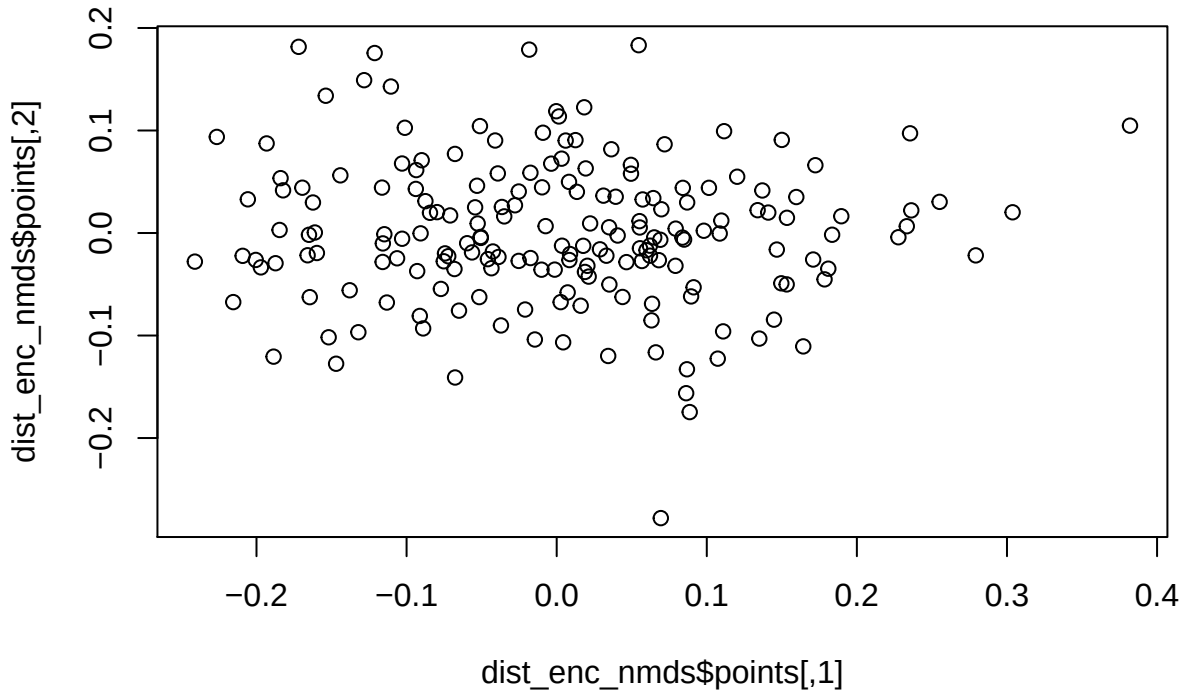
```
preguntas_d_pam <- pam(dist_enc, 3, diss = T)

dist_enc_nmds <- isoMDS(dist_enc + 0.0001)
```

```
## initial  value 24.647056
## iter   5 value 19.363468
## iter   5 value 19.360578
## iter   5 value 19.351987
## final  value 19.351987
## converged
```

```
dist_enc_nmds$stress
```

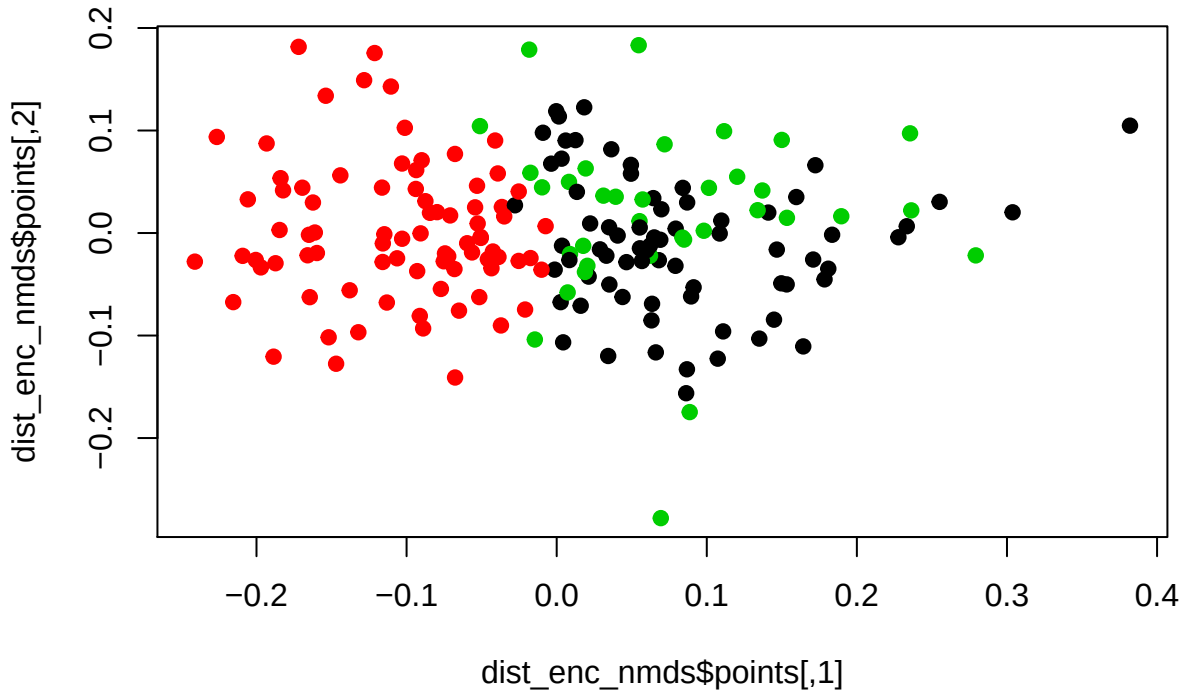
```
## [1] 19.35199
```

```
plot(dist_enc_nmds$points)
```



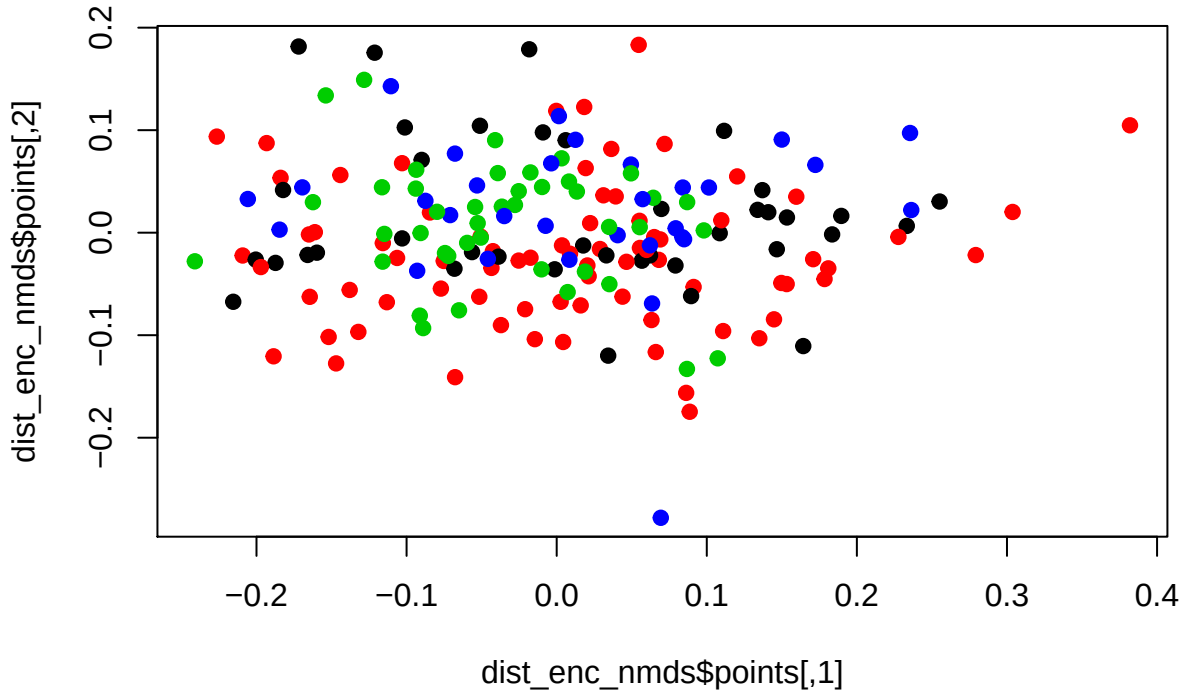
¿Cómo se agrupan los encuestados según sus respuestas?

```
plot(dist_enc_nmds$points, col=preguntas_d_pam$clustering, pch=19)
```



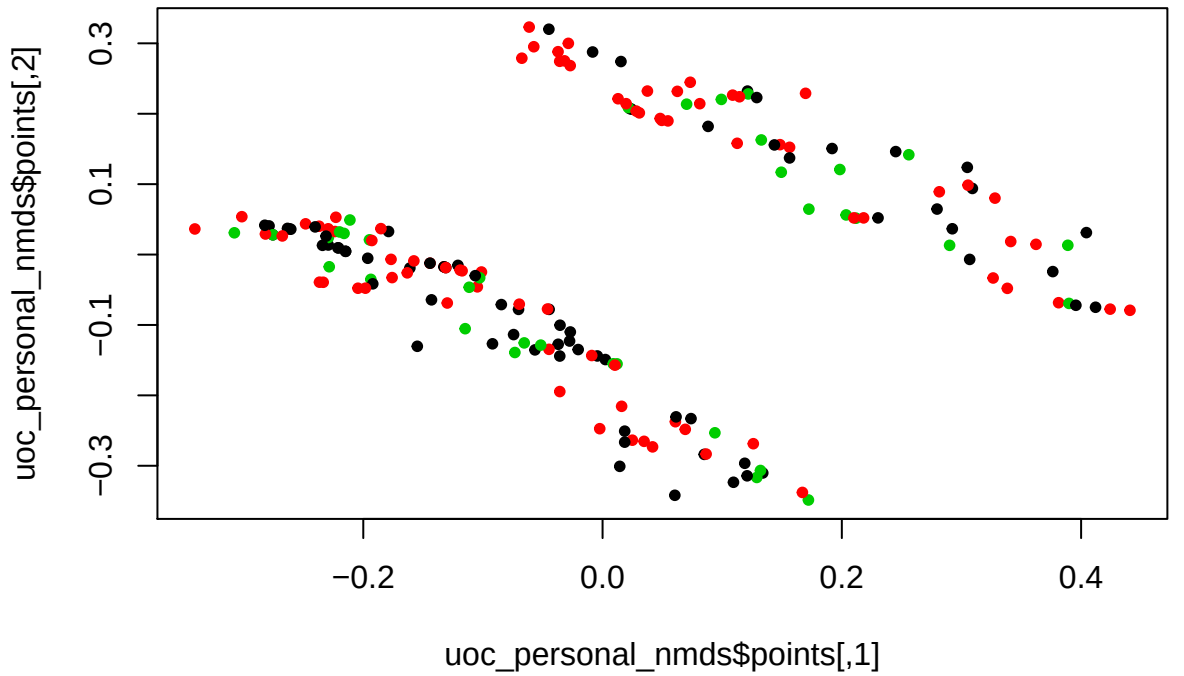
Y según sus características demográficas y profesionales ¿Cómo se distribuyen sobre la nube de respuestas?

```
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 para determinar si hay alguna asociación entre grupos demograficos y perfil de respuestas a la encuesta.

```r
table(personal_pam$clustering, preguntas_d_pam$clustering, dnn=c("grupo demográfico", "grupo de respues
```

```
##                   grupo de respuestas
## grupo demográfico  1  2  3
##                 1 16 14  9
##                 2 34 30 12
##                 3 11 25  6
##                 4 11 12  8
```

Como mencionamos antes, para analizar este trabajo en un contexto real de investigación harían falta algunos pasos más y profundizar algunos de los realizados. Pero el objetivo principal de este laboratorio fue aprender cómo aplicar y evaluar técnicas de clustering.