UNIVERSITAT DE BARCELONA

Universitat Oberta de Catalunya

Análisis de Conglomerados
Soluciones

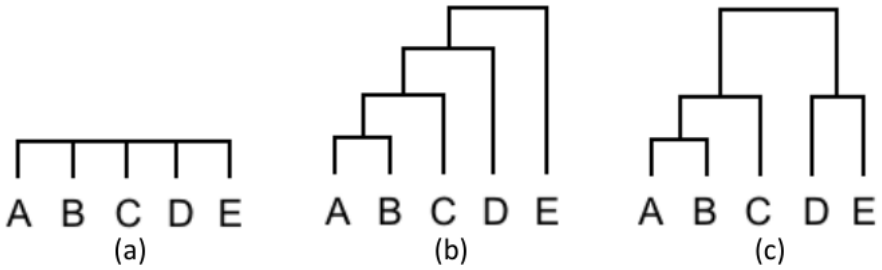
Francesc Carmona y Josep Gregori*

12 de mayo de 2019

Ejercicio 1

Tenemos 5 objetos A, B, C, D, E. Dibujar tres dendogramas en los casos que (a) sólo hay un conglomerado o cluster, (b) los conglomerados son $\{A, B\} \subset \{A, B, C\} \subset \{A, B, C, D\} \subset \{A, B, C, D, E\}$, (c) los conglomerados son $\{A, B\} \subset \{A, B, C\}, \{D, E\}$.
Inventar una distancia entre los objetos de forma que la clasificación jerárquica proporcione los conglomerados del caso (c). Probar que es correcta con la función `hclust()` *y dibujar el dendograma con la ayuda de la función* `plot()` *de* ***R****.*

Dibujamos los tres dendogramas:



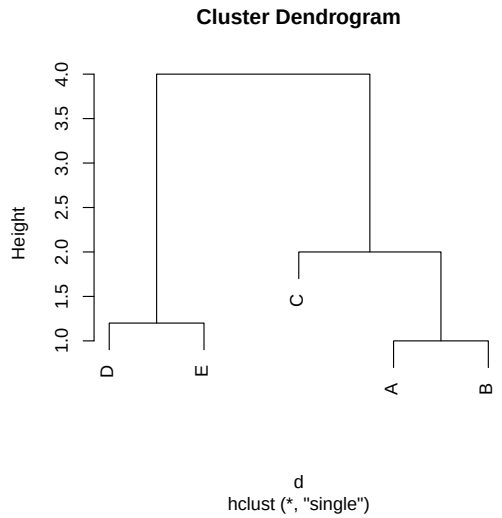
Una posible distancia entre los objetos para obtener una clasificación jerárquica como en el caso (c) es la siguiente:

```
> d <- matrix(c(0,1,2,4,5,
+               1,0,2.5,4,5,
+               2,2.5,0,4,5,
+               4,4,4,0,1.2,
+               5,5,5,1.2,0),nrow=5)
> rownames(d) <- colnames(d) <- c("A","B","C","D","E")
> (d <- as.dist(d))

    A   B   C   D
B 1.0
C 2.0 2.5
D 4.0 4.0 4.0
E 5.0 5.0 5.0 1.2
```

Comprobamos que con esta distancia obtenemos la clasificación (c).

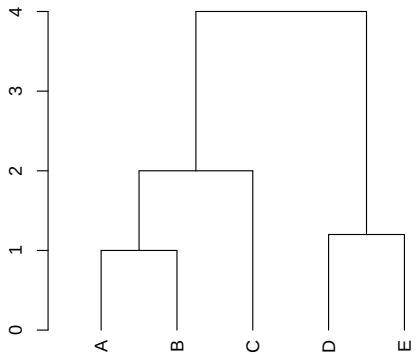
*Alumno del curso 2009-10

```
> hc <- hclust(d, method="single")
> plot(hc)
```



Podemos reordenar el dendograma para que quede como (c):

```
> dd <- as.dendrogram(hc)
> wts <- c(1,2,5,10,12)
> dd.reorder <- reorder(dd, wts = wts)
> plot(dd.reorder)
```



Los otros métodos también proporcionan el mismo dendograma con diferentes alturas.

```
> plot(hclust(d, method="complete"))
> plot(hclust(d, method="average"))
> plot(hclust(d, method="centroid"))
> plot(hclust(d, method="ward.D2"))
```

Ejercicio 2

El data.frame `all.mammals.milk.1956` *del paquete* `cluster.datasets` *contiene los datos de los ingredientes en la leche materna de 25 animales*[1].
Las variables medidas son agua, proteínas, grasa, lactosa y cenizas (los valores están en porcentaje).

```
> data(milk, package = "flexclust")
> str(milk)

'data.frame': 25 obs. of  5 variables:
 $ water  : num  90.1 88.5 88.4 90.3 90.4 87.7 86.9 82.1 81.9 81.6 ...
 $ protein: num  2.6 1.4 2.2 1.7 0.6 3.5 4.8 5.9 7.4 10.1 ...
 $ fat    : num  1 3.5 2.7 1.4 4.5 3.4 1.7 7.9 7.2 6.3 ...
 $ lactose: num  6.9 6 6.4 6.2 4.4 4.8 5.7 4.7 2.7 4.4 ...
 $ ash    : num  0.35 0.24 0.18 0.4 0.1 0.71 0.9 0.78 0.85 0.75 ...
```

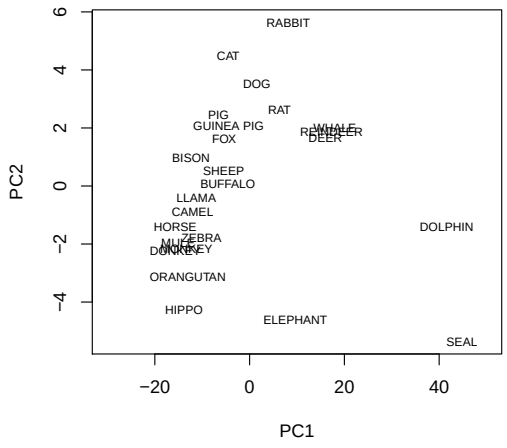
a) *Realizar un análisis de proximidades o escalado multidimensional con la distancia euclídea.*

El escalado multidimensional con la distancia euclídea es

```
> mds <- cmdscale(dist(milk),eig=TRUE)
> mds$eig[1:5]

[1] 6772.015878  195.223575   28.886094    7.337914    1.625682

> mds$GOF

[1] 0.9945968 0.9945968
```

Obtendremos una muy buena representación con las dos primeras coordenadas principales.

```
> plot(mds$points,type="n",xlab="PC1",ylab="PC2",xlim=c(-30,50))
> text(mds$points[,1],mds$points[,2],labels=rownames(milk),cex=0.7)
```

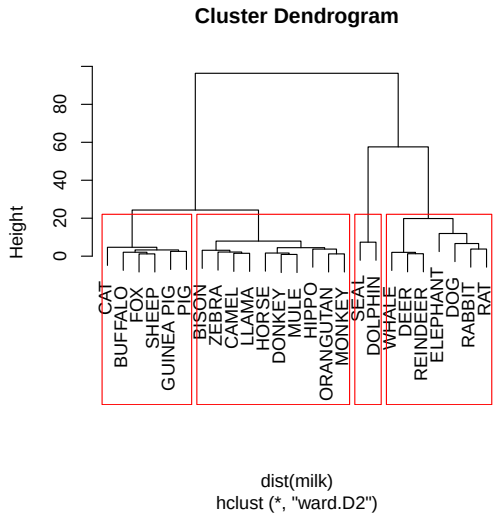


[1] También se pueden obtener en el archivo `milk` del paquete `flexclust` de **R**.

b) *Realizar un análisis de conglomerados jerárquico con las distancias euclídeas y el método de Ward.*

La agrupación por el método de Ward sobre las distancias euclídeas es:

```
> milk.hc <- hclust(dist(milk), method="ward.D2")
> plot(milk.hc)
> rect.hclust(milk.hc, k=4, border="red")
```



c) (∗) *Repetir los dos análisis anteriores con la distancia de Bhattacharyya.*

Calcularemos en primer lugar la distancia de Bhattacharyya. Para ello es necesario escalar los datos entre 0 y 1.

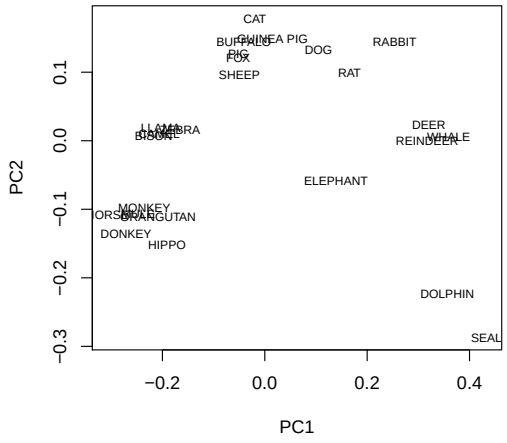
```
> Q <- sqrt(as.matrix(milk)/100)
> BC <- Q %*% t(Q)
> BC <- ifelse(BC>1., 1., BC)    # dado que algunos elementos son >1
> # Distancia de Bhattacharyya (al cuadrado)
> D2B <- acos(BC)
```

El MDS con esta distancia es:

```
> mdsB <- cmdscale(sqrt(D2B), eig=T)
> mdsB$eig[1:5]

[1] 1.30259766 0.38677430 0.18007846 0.13346800 0.07751937

> mdsB$GOF

[1] 0.6638957 0.7264805
```

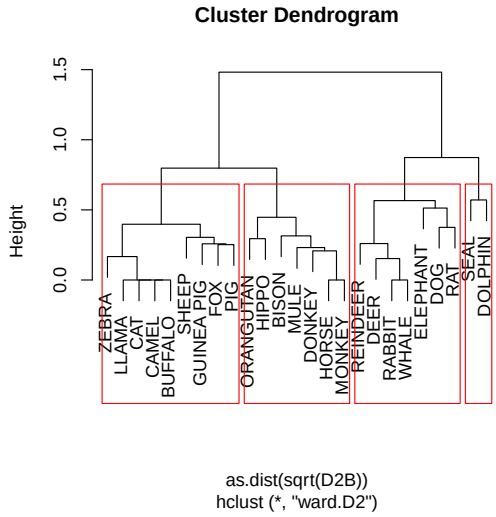
y su representación

```
> plot(mdsB$points,type="n",xlab="PC1",ylab="PC2")
> text(mdsB$points[,1],mdsB$points[,2],labels=rownames(milk),cex=0.7)
```



Las agrupaciones por el método de Ward sobre las distancias de Bhattacharyya y su representación en forma de dendograma son:

```
> milkB.hc <- hclust(as.dist(sqrt(D2B)), method="ward.D2")
> plot(milkB.hc)
> rect.hclust(milkB.hc, k=4, border="red")
```



Finalmente, podemos comparar las dos agrupaciones:

```
> hr <- cutree(milk.hc,4)
> hrB <- cutree(milkB.hc,4)
> (tt <- table(hr,hrB))

   hrB
hr  1 2 3 4
  1 7 3 0 0
  2 0 6 0 0
  3 0 0 7 0
  4 0 0 0 2
```

El número de coincidencias (en porcentaje) es:

```
> 100*sum(diag(tt))/sum(tt)

[1] 88
```

Ejercicio 3

El paquete `cluster` *de* ***R*** *dispone de los datos Ruspini, muy conocidos en la literatura del análisis de conglomerados.*

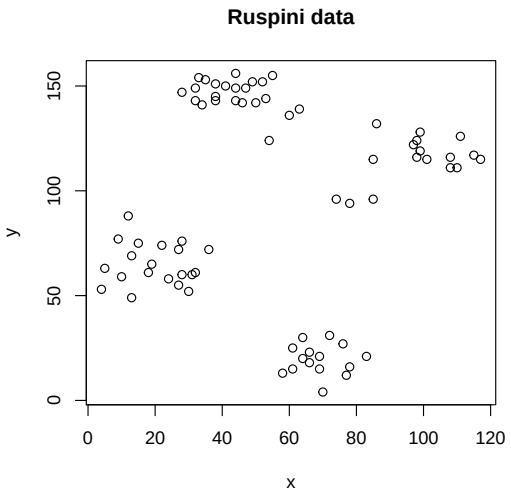
```
> library(cluster)
> data(ruspini)
> # help(ruspini)
> str(ruspini)

'data.frame': 75 obs. of  2 variables:
 $ x: int  4 5 10 9 13 13 12 15 18 19 ...
 $ y: int  53 63 59 77 49 69 88 75 61 65 ...
```

a) *Aplicar el procedimiento de particionado de las k-medias con el algoritmo de Hartigan-Wong para 4 grupos.*

Dibujar los puntos con un símbolo distinto para cada grupo. Hallar los centros de cada grupo y dibujarlos en el gráfico anterior. Hallar las sumas de cuadrados dentro de cada grupo y los tamaños de los conglomerados.

Como se trata de unos datos bivariantes, los podemos representar con 2 ejes para hacernos una idea de sus posibles agrupaciones .ª ojo".

```
> plot(ruspini)
> title(main="Ruspini data")
```



En principio parece que hay cuatro grupos con alguna duda. Tal vez tres...

Procedemos con el particionado de las k-medias con el algoritmo de Hartigan-Wong para 4 grupos.

```
> # Fijamos la semilla aleatoria
> set.seed(123)
> # Formamos 4 grupos por k-means
> km4 <- kmeans(ruspini, centers=4, algorithm="Hartigan-Wong"); km4$cluster

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  1  1  1  1  1
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
 2  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4
```

Los centros de cada grupo son:

```
> km4$centers

         x        y
1 43.91304 146.0435
2 98.17647 114.8824
3 20.15000  64.9500
4 68.93333  19.4000
```

las sumas de cuadrados dentro de cada grupo:

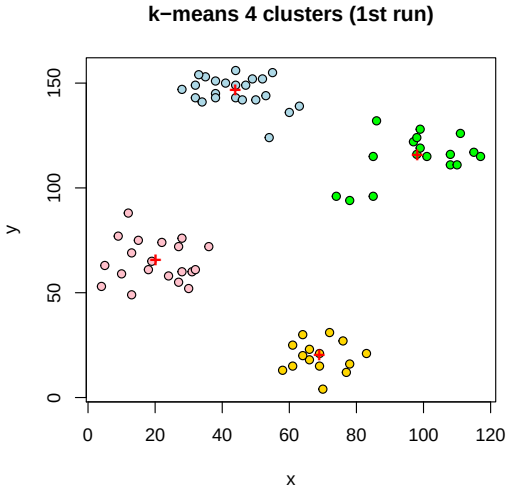
```
> km4$withinss

[1] 3176.783 4558.235 3689.500 1456.533
```

y los tamaños de cada grupo:

```
> km4$size

[1] 23 17 20 15
```

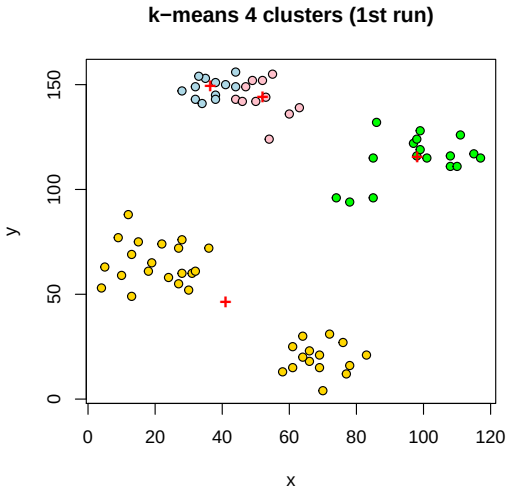
Finalmente hacemos la representación gráfica de los 4 conglomerados y sus puntos medios.

```
> gcol <- c("lightblue","green","pink","gold")
> plot(ruspini, pch=21, bg=gcol[km4$cluster])
> title(main="k-means 4 clusters (1st run)")
> text(km4$centers,labels="+",col="red",font=2,cex=1.2)
```



El resultado es el intuitivamente esperado. Repetimos para ver qué ocurre.

```
> km4 <- kmeans(ruspini, centers=4, algorithm="Hartigan-Wong")
> plot(ruspini, pch=21, bg=gcol[km4$cluster])
> title(main="k-means 4 clusters (1st run)")
> text(km4$centers,labels="+",col="red",font=2,cex=1.2)
```



Observamos con un cierto asombro que las agrupaciones no son las que esperábamos. Así, con este ejemplo, queda demostrado que el algoritmo de k-means puede llevarnos a un resultado inesperado si no acertamos en los puntos medios o los dejamos al azar.

b) *Realizar un particionado alrededor de los medoides*[2] *o k-medoides o PAM con la función* `pam()` *del paquete* ***cluster*** *de* ***R*** *y hallar los medoides.*

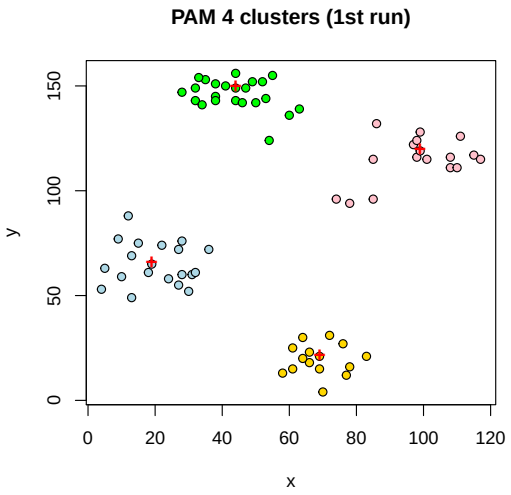
```
> pam4 <- pam(ruspini,4,diss=FALSE); pam4$medoids

    x   y
```

[2] `http://en.wikipedia.org/wiki/K-medoids`

```
10 19  65
32 44 149
52 99 119
70 69  21
```

La representación de los conglomerados por este método es:

```
> plot(ruspini,pch=21,bg=gcol[pam4$clustering])
> title(main="PAM 4 clusters (1st run)")
> text(pam4$medoids[,1], pam4$medoids[,2], labels="+", col="red",font=2, cex=1.2)
```



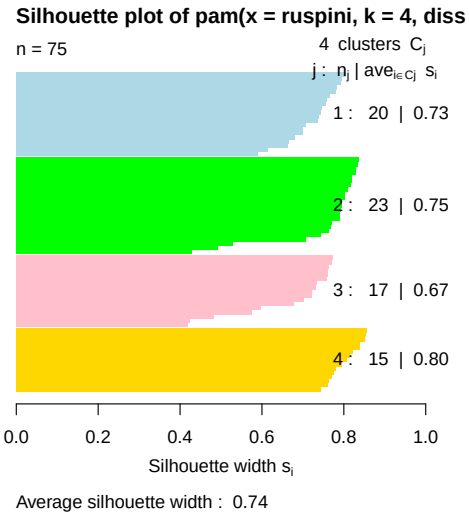
Observamos la clasificación "natural" esperada y la coincidencia de los medoides con cuatro observaciones, una de cada grupo.

La repetición del cálculo del PAM en veces sucesivas proporciona el mismo resultado, lo que muestra una mayor robustez de este método frente al de las k-medias.

c) *Calcular la silueta*[3] *como medida de la calidad de los conglomerados formados.*

```
> plot(silhouette(pam4),col=gcol)
```

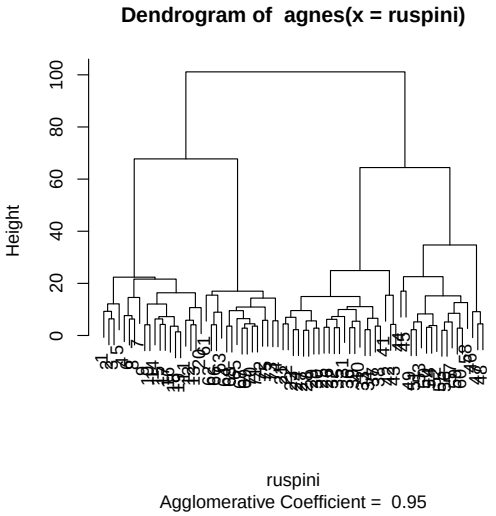
[3] http://en.wikipedia.org/wiki/Silhouette_(clustering)

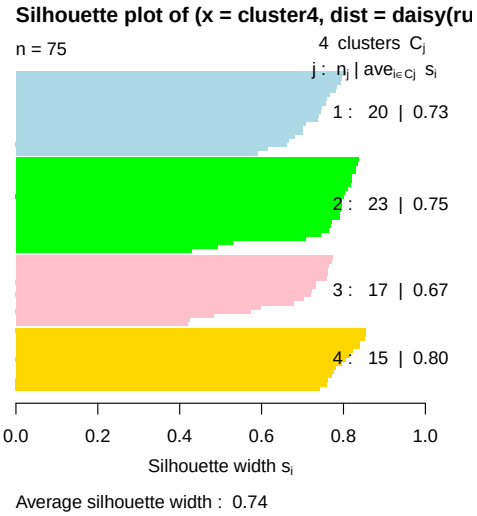


El índice medio de la silueta para este particionado es de 0.74.

d) *Realizar un particionado jerárquico con la función* `agnes()` *del paquete* ***cluster****. Probar con diversos números de conglomerados y decidir la mejor partición.*

```
> ar <- agnes(ruspini)
> plot(ar, which.plots = 2)
> cluster4 <- cutree(ar, k=4)
> si4 <- silhouette(cluster4, daisy(ruspini))
> plot(si4, col=gcol)
```

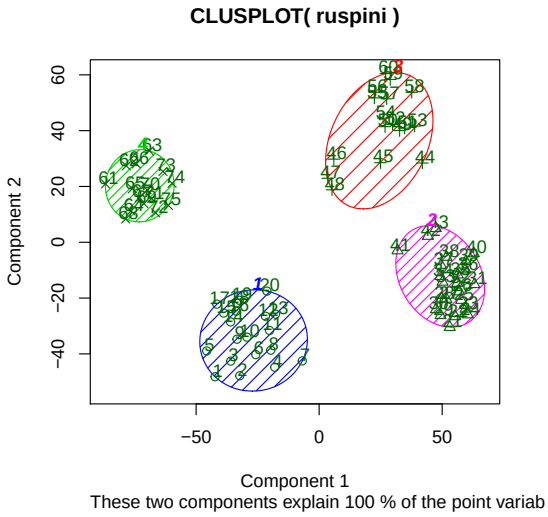




El resultado coincide con el PAM.

Una buena idea es representar los conglomerados resultantes.

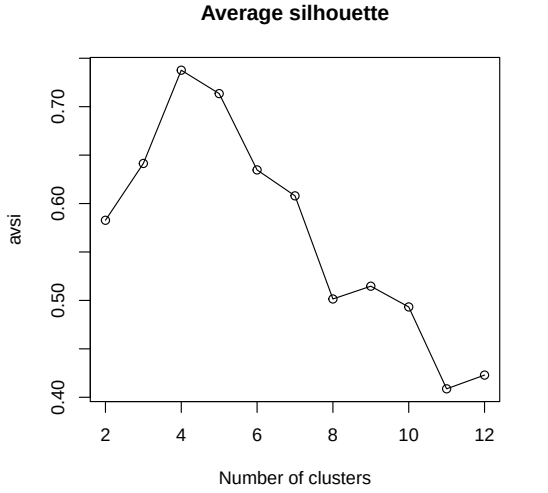
```
> clusplot(ruspini, cluster4, color=TRUE, shade=TRUE, labels=2, lines=0)
```



Esta representación utiliza como ejes las dos primeras componentes principales.

Finalmente vamos a probar con diversos números de conglomerados.

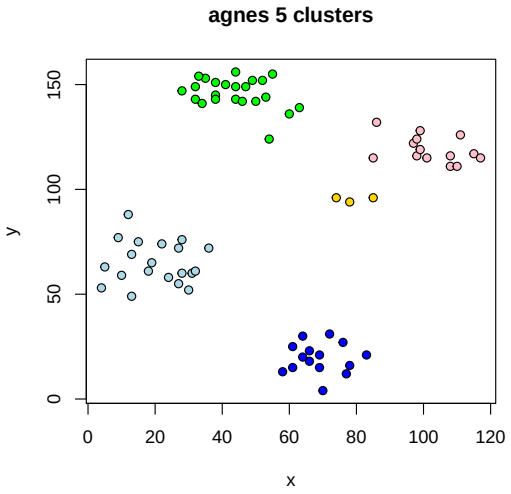
```
> d.eucl <- daisy(ruspini)
> avsi <- c(0)
> for(i in 2:12){
+   si <- silhouette(cutree(ar, k=i), d.eucl)
+   avsi[i-1] <- mean(si[,"sil_width"])
+ }
> plot(2:12,avsi,type="o",xlab="Number of clusters")
> title(main="Average silhouette")
```



Según la media general de las siluetas, el mejor número de conglomerados es 4.
Sin embargo, otras opciones no resultan nada descabelladas y pueden tener sentido.

```
> ar5 <- cutree(ar,k=5)
> gcol2 <- c("lightblue","green","pink","gold","blue")
```

```
> plot(ruspini,pch=21,bg=gcol2[ar5])
> title(main="agnes 5 clusters")
```



Resumen

En este ejercicio se ha mostrado como la respuesta de k-means depende de la selección de los centros iniciales, ya sea por designación o al azar, de modo que el resultado podría ser absolutamente inesperado.

También hemos visto como el PAM parece dar siempre el mismo resultado y se ajusta a los cuatro grupos "naturales" que obsrvamos con los datos de Ruspini. Aquí en lugar de un centro geométrico para cada cluster, se utiliza como centro un elemento del propio cluster, llamado medoide.

Las siluetas son también una buena representación de lo compacto o difuso que resulta un cluster y en situaciones multidimensionales es una buena alternativa de representación de la calidad de la partición.

La utilización de la media conjunta de la silueta nos da un criterio para decidir el número de conglomerados óptimo (con este criterio). En el caso de los datos de Ruspini hemos visto que ese número es 4. Sin embargo, con los mismos datos, también tenemos una justificación visual intuitiva para $k = 5, 6$ o 7.

En definitiva hay que valorar el criterio del experto, en el análisis de conglomerados y en otros, que siempre debe estar por encima del numérico o estadístico y en los que nos podemos apoyar para decidir.

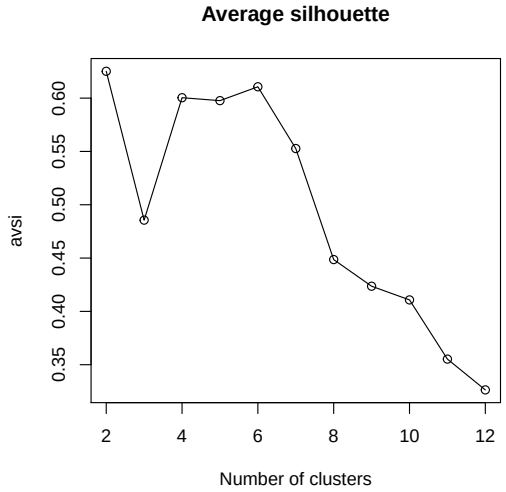
Ejercicio 4

Realizar un particionado alrededor de los k-medoides con los datos de la leche materna del ejercicio 2.
Estudiar el mejor número de conglomerados con ayuda de sus siluetas.
Dibujar un gráfico bidimensional con los conglomerados hallados.

Para determinar el número óptimo de conglomerados utilizaremos las siluetas.

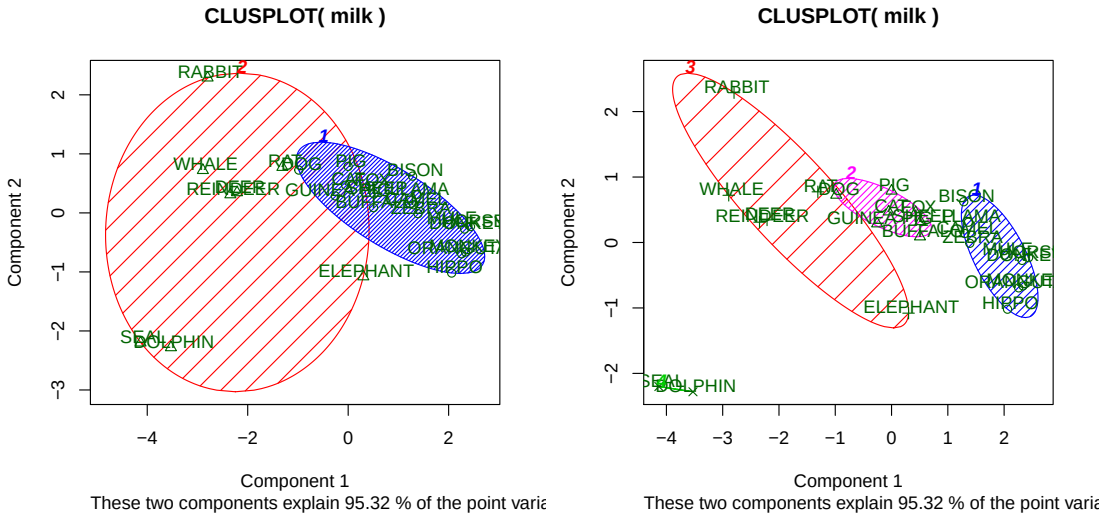
```
> avsi <- c(0)
> for(i in 2:12){
+   si <- silhouette(pam(milk,k=i))
+   avsi[i-1] <- mean(si[,"sil_width"])
+ }
```

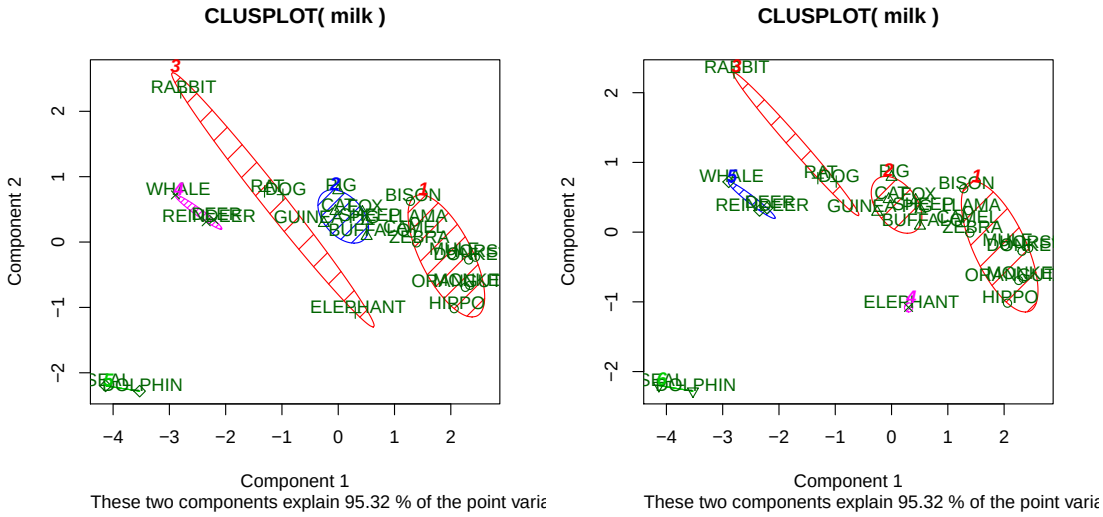
```
> plot(2:12, avsi, type="o", xlab="Number of clusters")
> title(main="Average silhouette")
```



A la vista de la calidad de las siluetas tenemos dudas sobre si el mejor número de conglomerados es 2, 4, 5 o 6.
Hagamos una representación por componentes principales:

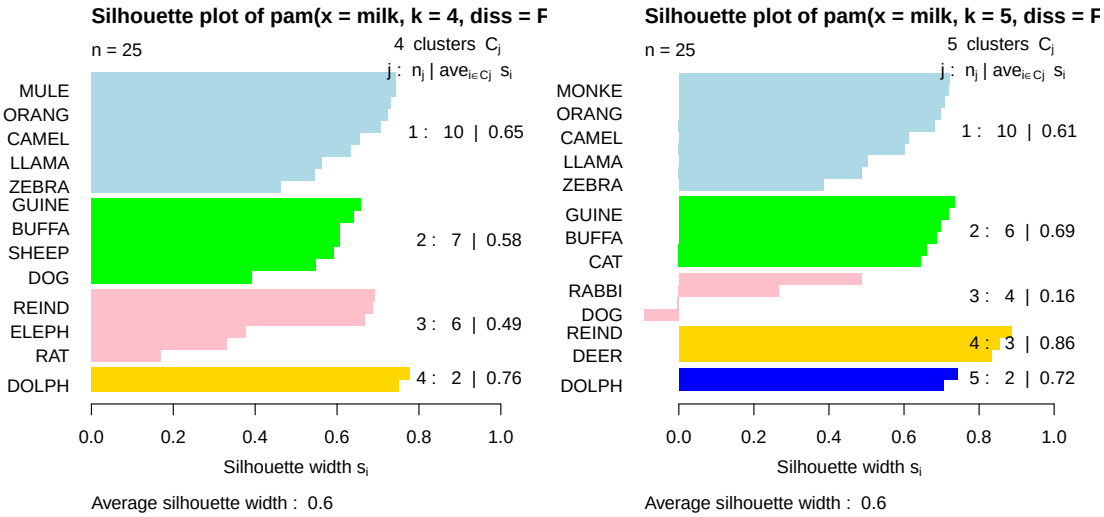
```
> pam2 <- pam(milk,2,diss=FALSE)
> clusplot(milk, pam2$clustering, color=TRUE, shade=TRUE, labels=2, lines=0)
> pam4 <- pam(milk,4,diss=FALSE)
> clusplot(milk, pam4$clustering, color=TRUE, shade=TRUE, labels=2, lines=0)
> pam5 <- pam(milk,5,diss=FALSE)
> clusplot(milk, pam5$clustering, color=TRUE, shade=TRUE, labels=2, lines=0)
> pam6 <- pam(milk,6,diss=FALSE)
> clusplot(milk, pam6$clustering, color=TRUE, shade=TRUE, labels=2, lines=0)
```





Está claro que la mayor variabilidad de la primera componente explica que los grupos se formen en esa dirección.
A falta del criterio de un experto, las gráficos anteriores muestran la atomización en el caso de 6 conglomerados y la grosería en el caso de 2.
En cuanto a las siluetas:

```
> si4 <- silhouette(pam4)
> plot(si4,col=gcol)
> si5 <- silhouette(pam5)
> plot(si5,col=gcol2)
```



Parece que con 4 conglomerados obtenemos un buen resultado.

Ejercicio 5

En el data.frame `birth.death.rates.1966` *del paquete* `cluster.datasets` *tenemos los datos en porcentaje de nacimientos y muertes de 70 países*[4].

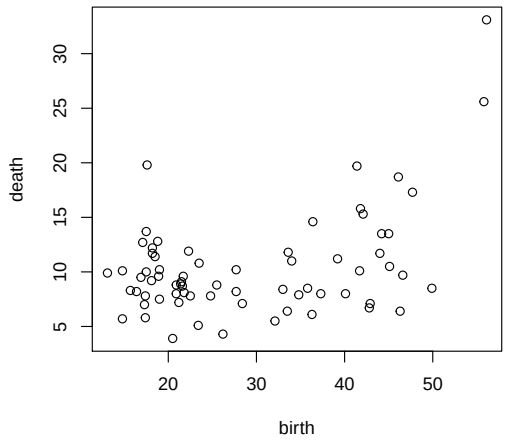
[4] También se pueden obtener en el archivo `birth` del paquete flexclust de **R**.

```
> data(birth, package = "flexclust")
> str(birth)

'data.frame': 69 obs. of  2 variables:
 $ birth: num  36.4 37.3 42.1 55.8 56.1 41.8 46.1 41.7 41.4 35.8 ...
 $ death: num  14.6 8 15.3 25.6 33.1 15.8 18.7 10.1 19.7 8.5 ...
```

a) *Realizar de forma exploratoria un análisis de conglomerados jerárquico con la función* `agnes()` *y su dendograma.*

Como se trata de datos bivariantes, hacemos primero una representación en diagrama de dispersión.

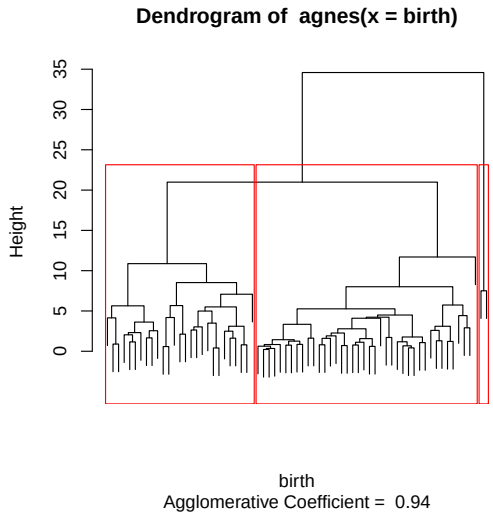
```
> plot(birth)
```



Vemos dos observaciones muy alejadas del centro y, tal vez, dos grupos en el resto.

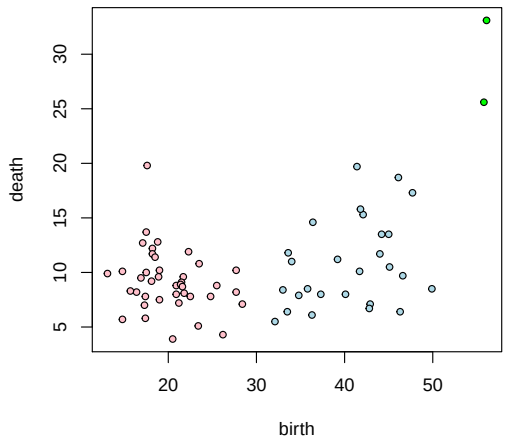
El análisis jerárquico es:

```
> ab <- agnes(birth)
> plot(ab,which.plots=2,labels=FALSE)
> rect.hclust(ab, k=3, border="red")
```



Aunque también es posible formar 5 grupos, veamos que con 3 obtenemos el resultado intuitivo.

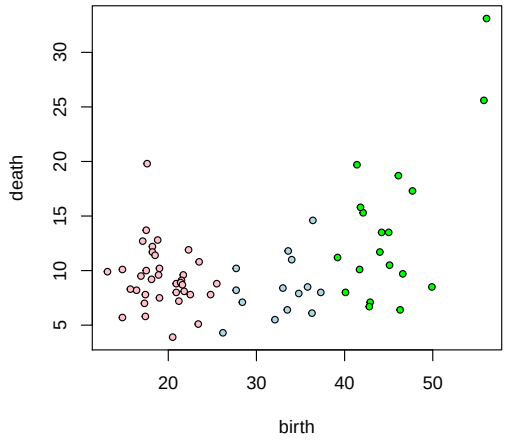
```
> ab3 <- cutree(ab,k=3)
> gcol <- c("lightblue","green","pink")
> plot(birth, xlab="birth", ylab="death",cex=0.8,pch=21,bg=gcol[ab3])
```



b) *Por el apartado anterior parece que hay tres grupos. Realizar un PAM con $k = 3$ grupos y probar distintos números con ayuda de sus siluetas.*

Veamos el PAM con $k = 3$ grupos.

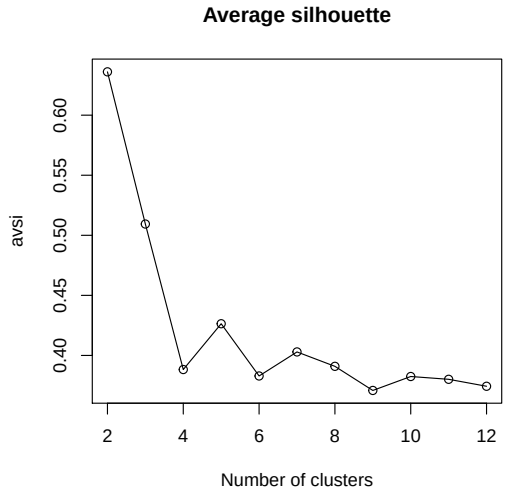
```
> pam3 <- pam(birth,k=3)
> plot(birth,xlab="birth",ylab="death",cex=0.8,pch=21,bg=gcol[pam3$clustering])
```



El resultado parece un poco sorprendente.

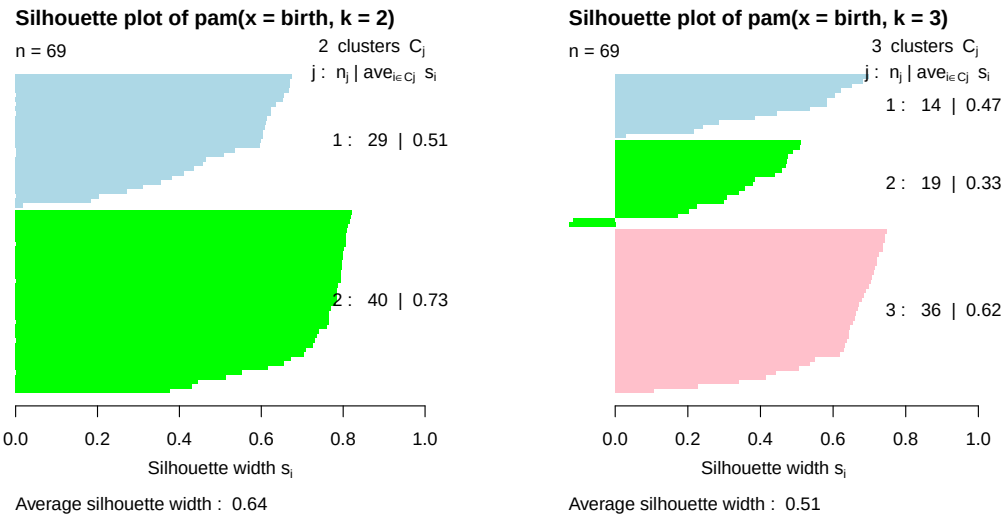
Vamos a determinar el número óptimo de clusters por el criterio de la silueta.

```
> avsi <- c(0)
> for(i in 2:12){
+   si <- silhouette(pam(birth,k=i))
+   avsi[i-1] <- mean(si[,"sil_width"])
+ }
> plot(2:12,avsi,type="o",xlab="Number of clusters")
> title(main="Average silhouette")
```



Según este criterio debemos hacer 2 clusters. Veamos las siluetas:

```
> pam2 <- pam(birth,k=2)
> si2 <- silhouette(pam2)
> plot(si2,col=gcol[1:2])
> si3 <- silhouette(pam3)
> plot(si3,col=gcol)
```



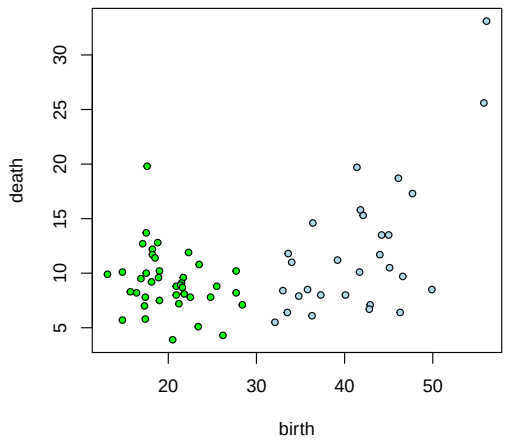
Dos observaciones tienen coeficiente negativo, son las dos más alejadas del grupo 2.

Parece que el algoritmo PAM no funciona muy bien con estos datos.

c) *Comparar el resultado del método de los k-medoides con el de las k-medias. Realizar una tabla cruzada de clasificación.*

Sabemos de la poca robustez del método de las k-medias, por lo que habrá que hacer algunas pasadas para ver si hay diferencias en el resultado.

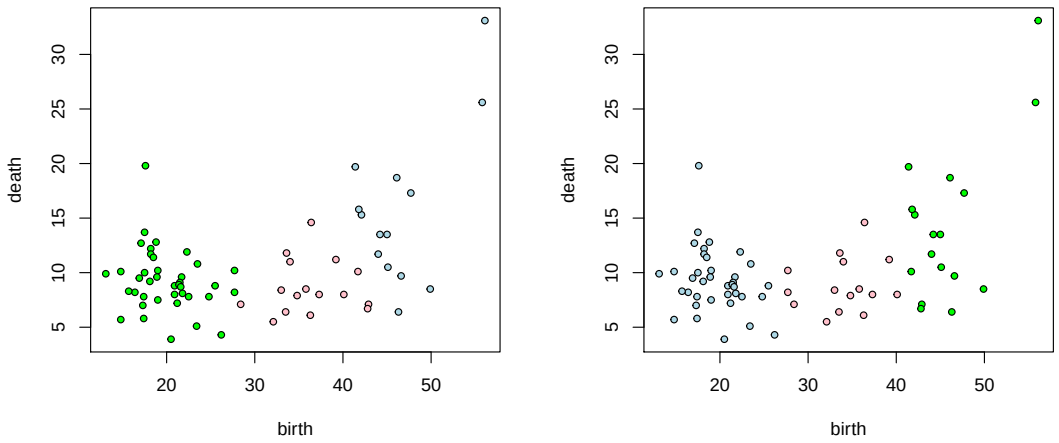
```
> # set.seed(123)
> km2 <- kmeans(birth,2)
> plot(birth,xlab="birth",ylab="death",cex=0.8,pch=21,bg=gcol[km2$cluster])
> km2.2 <- kmeans(birth,2)
> plot(birth,xlab="birth",ylab="death",cex=0.8,pch=21,bg=gcol[km2.2$cluster])
```



Vemos únicamente un cambio de color (debido al cambio de número de cluster), pero los grupos son iguales.

Ahora con 3 conglomerados.

```
> # set.seed(123)
> km3 <- kmeans(birth,3)
> plot(birth,xlab="birth",ylab="death",cex=0.8,pch=21,bg=gcol[km3$cluster])
> km3.2 <- kmeans(birth,3)
> plot(birth,xlab="birth",ylab="death",cex=0.8,pch=21,bg=gcol[km3.2$cluster])
```



El resultado es claramente distinto. No podemos aceptar este método a no ser que fijemos nosotros los puntos medios.

Las tablas cruzadas son:

```
> table(pam2$clustering,km2$cluster)


    1  2
1 29  0
2  0 40

> table(pam2$clustering,km2.2$cluster)


    1  2
1 29  0
2  0 40

> table(pam3$clustering,km3$cluster)


    1  2  3
1  0  3 11
2 14  0  5
3  0 36  0

> table(pam3$clustering,km3.2$cluster)
```

```
   1  2  3
1  1  0 13
2  0 17  2
3 36  0  0
```

El número de cluster es arbitrario. Un ajuste mejora la comparación. Por ejemplo, en la última tabla intercambiar el grupo 2 y el 3 para el PAM mostraría una mayor coincidencia.

Ejercicio 6 (∗)

El paquete `dendextend` *proporciona un conjunto de funciones para mejorar gráficamente los dendogramas. Con la clasificación jerárquica en cuatro grupos obtenida por el método de Ward en el ejercicio 2 con los datos de la leche materna de 25 animales, realizar el siguiente análisis gráfico:*

a) *Empezaremos con un gráfico del tipo matriz de diagramas de dispersión (*scatterplot matrix *o SPLOM) que podemos implementar con la función* `pairs()` *con los puntos en colores distintos según el conglomerado.*

En primer lugar recuperamos la clasificación jerárquica del ejercicio 2.

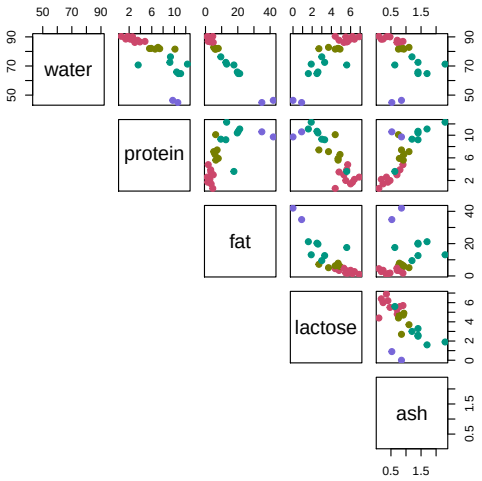
```
> data(milk, package = "flexclust")
> milk.hc <- hclust(dist(milk), method="ward.D2")
> hr <- cutree(milk.hc,4)
```

Para comparar resultados con el paquete `dendextend`, tomamos exactamente los mismos colores que utiliza dicho paquete y que se basan en la paleta `rainbow_hcl` del paquete `colorspace`.

```
> library(colorspace)
> my_colors <- rainbow_hcl(4,c = 90,l=50)[hr]
```

Ahora ya podemos dibujar la matriz de gráficos de dispersión:

```
> pairs(milk, col = my_colors, lower.panel = NULL,
+             pch=19, cex = 1.1)
```

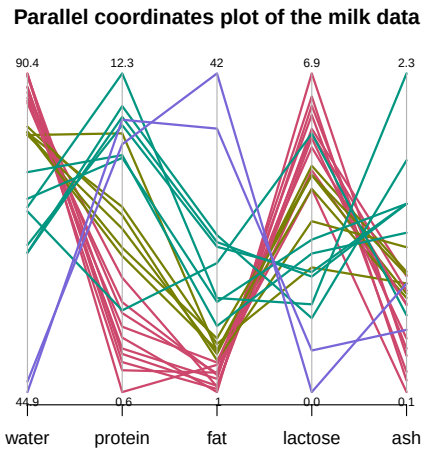


En algunos gráficos de dispersión se observa claramente la separación entre los grupos.

b) *Como tenemos pocas variables, también se puede hacer un gráfico de coordenadas paralelas con la ayuda de la función* `parcoord()` *del paquete* `MASS`*.*

Este gráfico se hace así:

```
> MASS::parcoord(milk, col = my_colors, var.label = TRUE, lwd = 2)
> title("Parallel coordinates plot of the milk data")
```



En este gráfico observamos como las variables separan a los grupos.

c) *Dibujar el dendograma en horizontal con cuatro colores, uno por conglomerado, tanto para las ramas como para las etiquetas.*

En primer lugar convertimos el objeto `hclust` en un objeto `dendrogram` y procuramos que el orden de los animales sea lo más parecido posible al inicial en la base de datos.

```
> library(dendextend)
> dend <- as.dendrogram(milk.hc)
> dend <- rotate(dend, rownames(milk))
```

A continuación coloreamos las ramas y las etiquetas en cuatro colores:

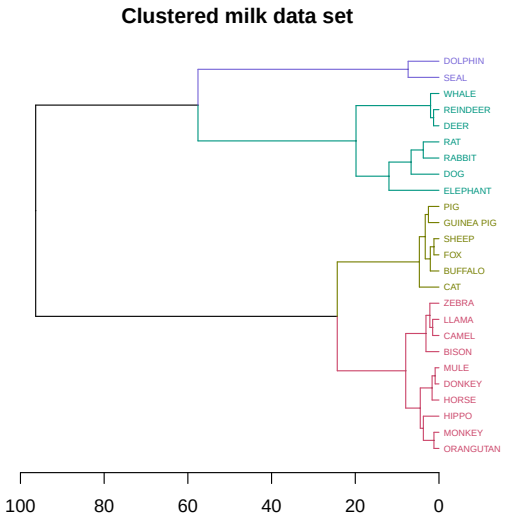
```
> dend <- branches_color(dend, k=4)
> dend <- set(dend, "labels_color", k=4)
```

Reducimos el tamaño de las etiquetas:

```
> dend <- set(dend, "labels_cex", 0.5)
```

Y dibujamos:

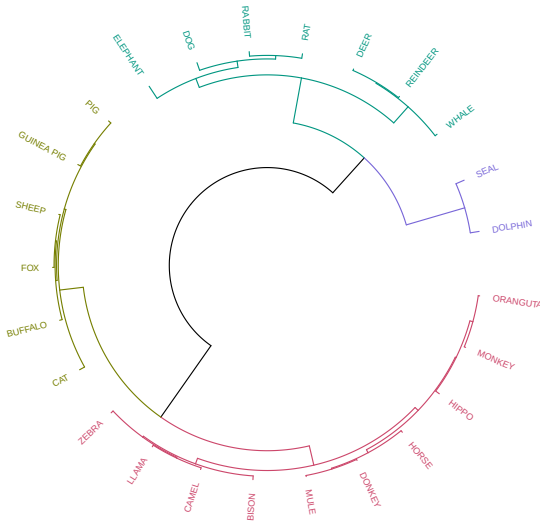
```
> par(mar = c(3,2,2,3))
> plot(dend, main = "Clustered milk data set",
+      horiz =  TRUE,  nodePar = list(cex = .007))
```



d) *Repetir el dendograma en forma circular con la función* `circlize_dendogram()` *que requiere que el paquete* `circlize` *esté instalado.*

Esto es sencillo:

```
> par(mar = rep(0,4))
> circlize_dendrogram(dend)
```

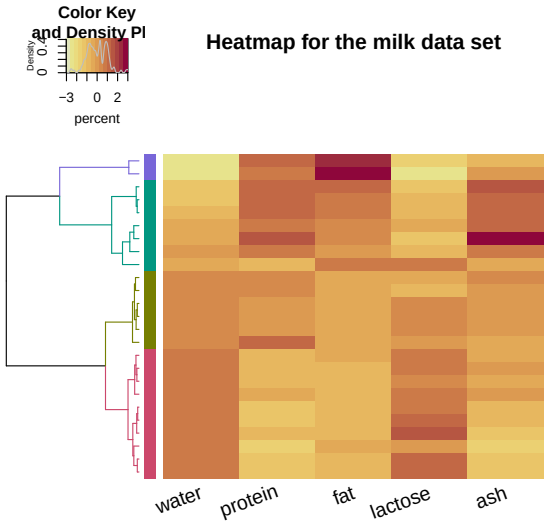


e) *Realizar un heatmap del dendograma con la función* `heatmap.2()` *del paquete* `gplots`.

Ahora vamos a explorar el dendograma en forma de *heatmap*. Las filas están ordenadas en base al orden creado por el análisis de conglomerados jerárquico (con el método `"complete"`). Los colores en el *heatmap* indican la cantidad de cada medida (desde el amarillo pálido al rojo oscuro).

```
> some_col_func <- function(n)
+                     rev(colorspace::heat_hcl(n, c = c(80, 30), l = c(30, 90),
+                                              power = c(1/5, 1.5)))
> # library(gplots)
```

```
> gplots::heatmap.2(scale(as.matrix(milk)),
+                   main = "Heatmap for the milk data set",
+                   srtCol = 20,
+                   dendrogram = "row",
+                   Rowv = dend,
+                   Colv = "NA", # this to make sure the columns are not ordered
+                   trace="none",
+                   margins =c(5,0.1),
+                   key.xlab = "percent",
+                   denscol = "grey",
+                   density.info = "density",
+                   RowSideColors = my_colors, # to add nice colored strips
+                   col = some_col_func
+                   )
```



Vemos que en el primer grupo, el de HORSE, hay mucha agua y lactosa. El grupo formado por DOLPHIN y SEAL se caracteriza por tener mucha grasa y poca agua y lactosa.

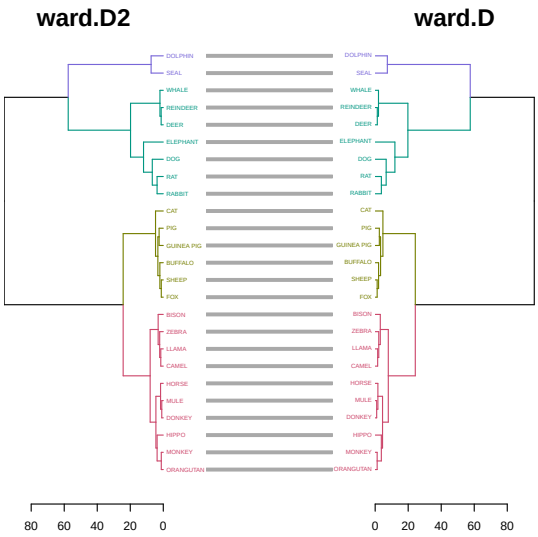
f) *Comparar las clasificaciones que proporcionan los algoritmos* `"ward.D"` *y* `"ward.D2"` *con la función* `tanglegram()`.

En primer lugar vamos a construir una `dendlist` con los dos dendogramas.

```
> milk.hc.d <- hclust(dist(milk), method="ward.D")
> dend2 <- as.dendrogram(milk.hc.d)
> dend2 <- set(dend, "labels_color", k=4)
> dend2 <- set(dend, "labels_cex", 0.5)
> milk_dendlist <- dendlist(dend, dend2)
> names(milk_dendlist) <- c("ward.D2","ward.D")
```

Y ahora procedemos a la comparación. Lo haremos en la forma del paquete `dplyr`.

```
> library(dplyr)
> milk_dendlist %>% ladderize %>%
+      set("branches_k_color", k=4) %>%
+      tanglegram(faster = TRUE)
```



No hay ninguna diferencia entre las dos clasificaciones.