Clusters

Briseyda Amancaya
María Anciones Polo
Laura Gil García
José Miguel Hernández Cabrera

Introducción

En este documento realizaremos un análisis de conglomerados o cluster aplicado a datos de conformación de diversas dietas de países europeos, utilizando los métodos de clasificación jerárquica, K-medias y PAM.

Paquetes

Los paquetes utilizados para realizar este análisis de clúster son `stats` y `cluster`. Además, utilizaremos las funciones `stats::hclust()`, `stats:dist()` y `cluster::diana()` para el análisis y los paquetes `factoextra` y `ggplot2` para la visualización de los grupos formados.

Descripción de datos

En primer lugar, importamos los datos teniendo en cuenta que hay que adaptar el directorio donde se encuentran los datos, en cuyo caso se encuentran en la carpeta `data/`.

```
paises = foreign::read.spss("data/PaisesProteinas.sav", to.data.frame = TRUE)
```

Posteriormente, podemos hacer una descriptiva básica de las distintas variables que componen la base de datos mediante la función `summary()`.

```
summary(paises)
```

```
##              Pais      CarneRoja       CarneBlanca         Huevos
##  Albania       : 1   Min.   : 4.400   Min.   : 1.400   Min.   :0.500
##  Alemania Occ. : 1   1st Qu.: 7.800   1st Qu.: 4.900   1st Qu.:2.700
##  Alemania Or.  : 1   Median : 9.500   Median : 7.800   Median :2.900
##  Austria       : 1   Mean   : 9.828   Mean   : 7.896   Mean   :2.936
##  Bélgica       : 1   3rd Qu.:10.600   3rd Qu.:10.800   3rd Qu.:3.700
##  Bulgaria      : 1   Max.   :18.000   Max.   :14.000   Max.   :4.700
##  (Other)       :19
##      Leche           Pescado          Cereales        Feculas
##  Min.   : 4.90   Min.   : 0.200   Min.   :18.60   Min.   :0.600
##  1st Qu.:11.10   1st Qu.: 2.100   1st Qu.:24.30   1st Qu.:3.100
##  Median :17.60   Median : 3.400   Median :28.00   Median :4.700
##  Mean   :17.11   Mean   : 4.284   Mean   :32.25   Mean   :4.276
##  3rd Qu.:23.30   3rd Qu.: 5.800   3rd Qu.:40.10   3rd Qu.:5.700
##  Max.   :33.70   Max.   :14.200   Max.   :56.70   Max.   :6.500
##
##   FrutosSecos    FrutosyVegetales
##  Min.   :0.700   Min.   :1.400
##  1st Qu.:1.500   1st Qu.:2.900
##  Median :2.400   Median :3.800
##  Mean   :3.072   Mean   :4.136
##  3rd Qu.:4.700   3rd Qu.:4.900
##  Max.   :7.800   Max.   :7.900
##
```

Seleccionamos la variable de `paises` para empezar el análisis. Como los métodos cluster son muy sensibles al hecho de que las variables no estén todas medidas en las mismas unidades, es necesario escalar las variables numéricas para que todas las variables tengan la misma importancia en el análisis.

La escala hace que todas las variables tengan media `0` y varianza `1`. Esto se realiza para evitar que el algoritmo de agrupamiento dependa de una unidad variable arbitraria.

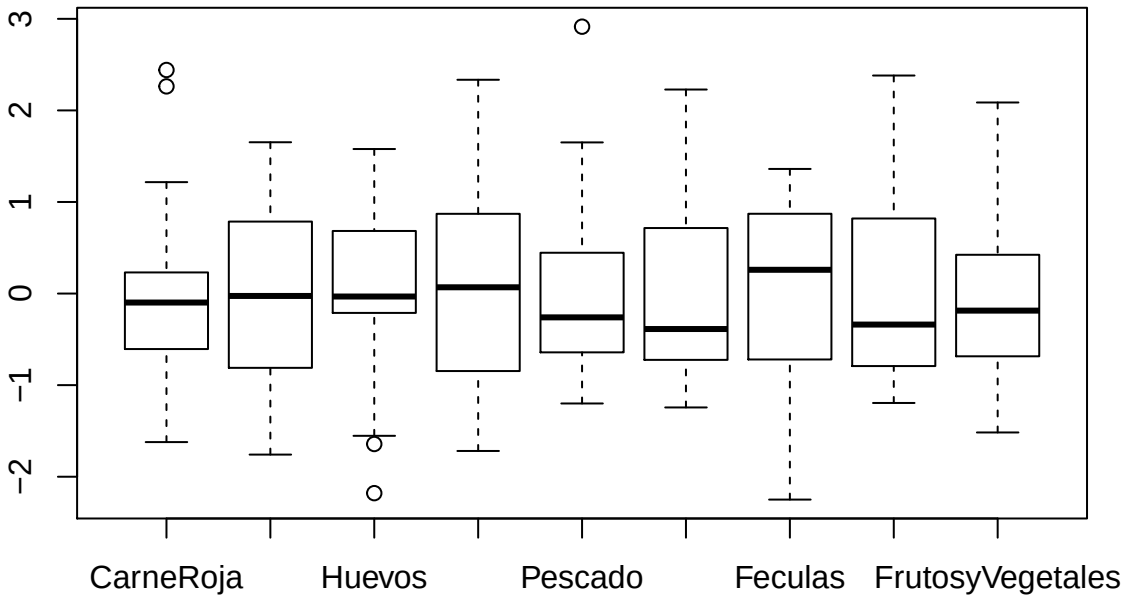
```
# Selección de variables numéricas
var.paises = paises[, 2:ncol(paises)]

# Crear matriz de estandarización
p.esc = scale(var.paises)

# Nombrar las filas con los nombres de los países
rownames(p.esc) = paises[, 1]

# Ver las primeras 6 observaciones
head(p.esc)
```

```
##                  CarneRoja CarneBlanca     Huevos       Leche     Pescado
## Albania         0.08126490  -1.7584889 -2.1796385 -1.15573814 -1.20028213
## Austria        -0.27725673   1.6523731  1.2204544  0.39237676 -0.64187467
## Bélgica         1.09707621   0.3800675  1.0415022  0.05460623  0.06348211
## Bulgaria       -0.60590157  -0.5132535 -1.1954011 -1.24018077 -0.90638347
## Checoslovaquia -0.03824231   0.9485445 -0.1216875 -0.64908235 -0.67126454
## Dinamarca       0.23064892   0.7861225  0.6835976  1.11013912  1.65053488
##                   Cereales    Feculas FrutosSecos FrutosyVegetales
## Albania         0.9159176 -2.2495772   1.2227536      -1.35040507
## Austria        -0.3870690 -0.4136872  -0.8923886       0.09091397
## Bélgica        -0.5146342  0.8714358  -0.4895043      -0.07539207
## Bulgaria        2.2280161 -1.9435955   0.3162641       0.03547862
## Checoslovaquia  0.1869740  0.4430614  -0.9931096      -0.07539207
## Dinamarca      -0.9428885  0.3206688  -1.1945517      -0.96235764
```

Detección de atípicos

La siguiente fase consiste en detectar si existen observaciones aberrantes o atípicas que puedan influir en el modelo.

```
boxplot(p.esc)
```



El boxplot detecta dos países con alto consumo de carne roja y uno de pescado, así como dos de bajo consumo de huevo. Para saber cuáles, utilizamos el método Tukey para detectar los atípicos, el cual consiste en:

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)]$$

Para detectar de cuáles países se tratan creamos la función `detec_atip()`:

```
detec_atip = function(x) {
  resultado = list()

  # Rangos de los Q1 y Q3
  ran.int = function(x) quantile(x, c(0.25, 0.75))

  # Detección usando el método de Tukey
  inferior = function(x) ran.int(x)[1] - (1.5 * IQR(x))
  superior = function(x) ran.int(x)[2] + (1.5 * IQR(x))

  # Escribir resultados en la lista creada
  resultado$ext.inferior = subset(x, x < inferior(x))
  resultado$ext.superior = subset(x, x > superior(x))

  return(resultado)
}
unlist(apply(p.esc, 2, detec_atip))
```

```
## CarneRoja.ext.superior.Francia
##                              2.441532
## CarneRoja.ext.superior.Reino Unido
##                              2.262272
##    Huevos.ext.inferior.Albania
##                             -2.179639
##    Huevos.ext.inferior.Portugal
##                             -1.642782
##   Pescado.ext.superior.Portugal
##                              2.914299
```

La función nos dice que `Francia` y el `Reino Unido` tienen un consumo alto atípico de carne roja. En el mismo sentido, `Portugal` consuma más pescado de lo normal. Por otra parte, `Portugal` y `Albania` consumen mucho menos huevo que el resto de los 23 países considerados.

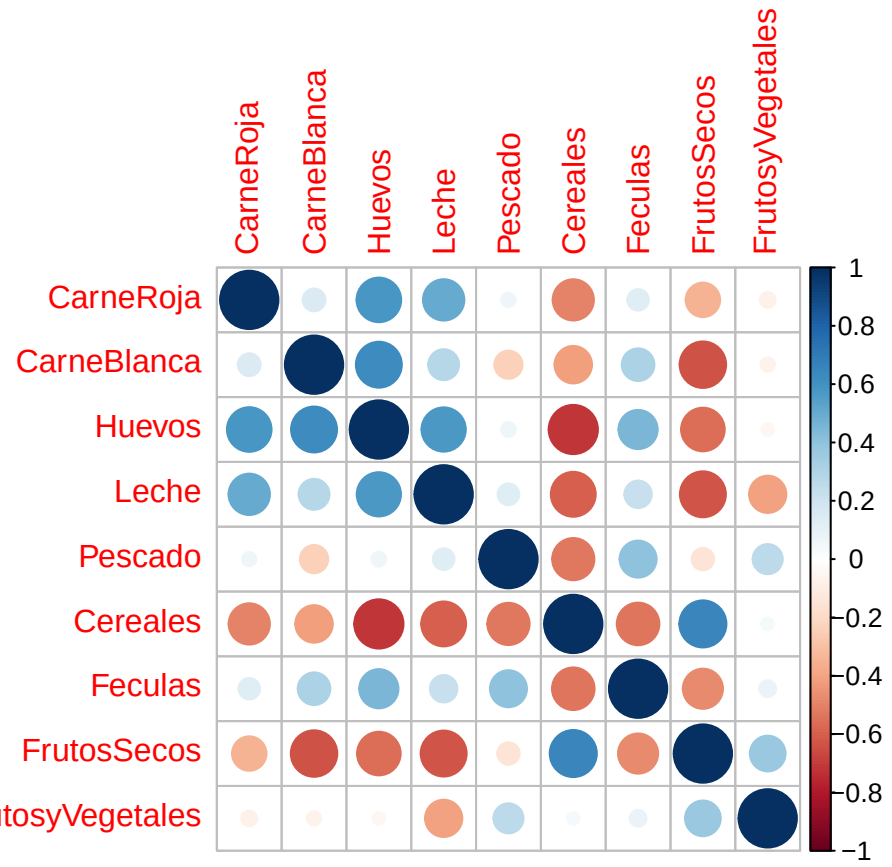
Generalmente, se recomienda hacer el análisis considerando a los atípicos y sin ellos o realizar un tratamiento de atípicos. No obstante, también es necesario notar que los datos en realidad pueden corresponder a una realidad que va más allá de lo que pueda explicar el modelo.

Para efectos del ejercicio, dejamos esas observaciones sin modificaciones.

Colinealidad

Para tener una correcta interpretación de los grupos formados, debemos asegurarnos que no haya colinealidad. En caso contrario, se puede optar por eliminarla o utilizar distancias que amortigüen la colinealidad.

```
corrplot::corrplot(cor(p.esc))
```



Podemos observar que `Cereales` y `FrutosSecos` están inversamente correlacionados con todas las demás variables. A su vez, `Huevos` y `Leche` tienen una relación lineal positiva con con `CarneRoja` y `CarneBlanca`.

Dado que el modelo de clusters requiere que no exista colinealidad, probablemente sea necesaria una reducción de dimensiones o eliminar las variables con alta correlación. No obstante, para efectos del ejercicio, procederemos con los datos completos para mostrar cómo se comportan los objetos.

Distancias

Hay diferentes tipos de distancias con sus propiedades particulares pero las más habituales son las siguientes:

Distancia euclídea: es la medida de similaridad más utilizada frecuentemente. Se trata de la distancia más corta entre dos puntos.

$$d_{euc}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Distancia de Manhattan: define la distancia entre dos puntos x e y como el sumatorio de las diferencias absolutas entre cada dimensión. Esta medida se ve menos afectada por atípicos (es más robusta) que la distancia euclídea debido a que no eleva al cuadrado las diferencias.

$$d_{man}(x,y) = \sum_{i=1}^{n}|(x_i - y_i)|$$

En este ejercicio utilizamos la distancia euclídea mediante la función `dist()`.

```
d.euc = dist(p.esc, method = "euclidean")
```

Clasificación jerárquica

Descripción

El análisis de cluster jerárquico se utiliza tanto para variables cuantitativas como para variables cualitativas. También se emplea si no se conoce el número de cluster o cuando el número de objetos no es muy grande.

Puede subdividirse en **aglomerativos** (fases sucesivas de fusiones de los n individuos) y en **divisivos** (particionan los n individuos).

Métodos de aglomeración

Hay diferentes métodos jerárquicos aglomerativos para el análisis de cluster:

Ward: no calcula distancias entre cluster pero forma un cluster que maximiza la homogeneidad intra cluster.

Método del vecino más próximo: la distancia entre grupos se caracteriza por la del par de individuos que está más cercano (un individuo de cada grupo).

Método del vecino más lejano: la distancia entre grupos es la mayor distancia entre pares de individuos (uno de cada grupo).

Grupo promedio o UPGMA: la distancia se calcula como la media entre todos los pares de individuos de cada grupo.

```
metodos = c("ward.D2",  # Ward
            "single",   # Vecinos más próximos
            "complete", # Vecinos más lejanos
            "average")  # Grupo promedio o UPGMA
names(metodos) = metodos

met.agl = lapply(metodos, function(x) hclust(d.euc, method = x))
```

Utilizamos el **coeficiente de aglomeración**, para saber cuál método se ajusta mejor:

```
coe_agl = sapply(met.agl, cluster::coef.hclust)
coe_agl = round(coe_agl, digits = 2)
coe_agl
```

```
##  ward.D2   single complete  average
##     0.83     0.36     0.70     0.59
```

Podemos observar que de todos los métodos, el de Ward se ajusta más adecuadamente.

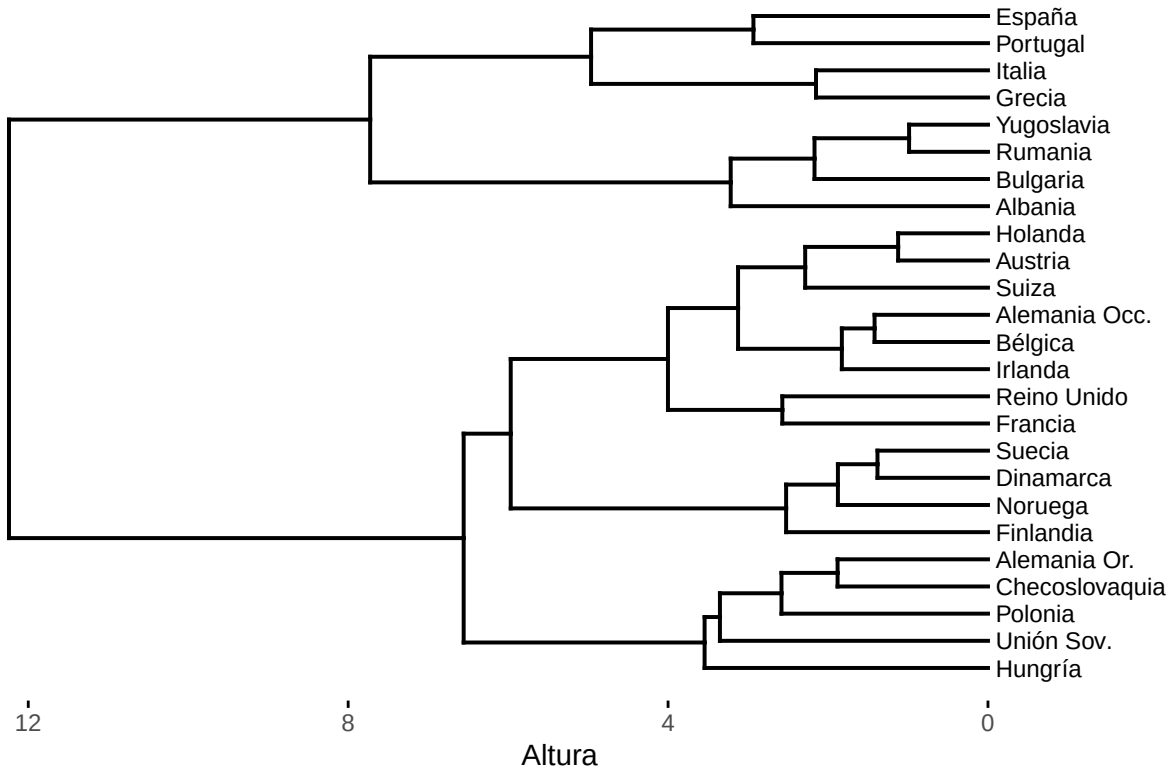
Visualización con dendrogramas

Los resultados del método jerárquico se representa gráficamente mediante un dendrograma, donde se indican las fusiones o divisiones producidas en las fases sucesivas del análisis.

```
library(ggplot2)
library(factoextra)

ttl.met = c(
  "Método de Ward",
  "Vecino más próximo",
  "Vecino más Lejano",
  "Grupo promedio"
)

coe.met = c(paste("Coef. aglo.:", coe_agl), "", "")

dendros = function(x, ttl, stl) {
  graf = fviz_dend(
    x,
    horiz = T,
    main = ttl,
    sub = stl,
    xlab = "",
    ylab = "Altura",
    cex = 0.6
  )
  return(graf)
}


lapply(1:4, function(x) {
  dendros(met.agl[[x]], ttl = ttl.met[x], stl = coe.met[x])
})
```

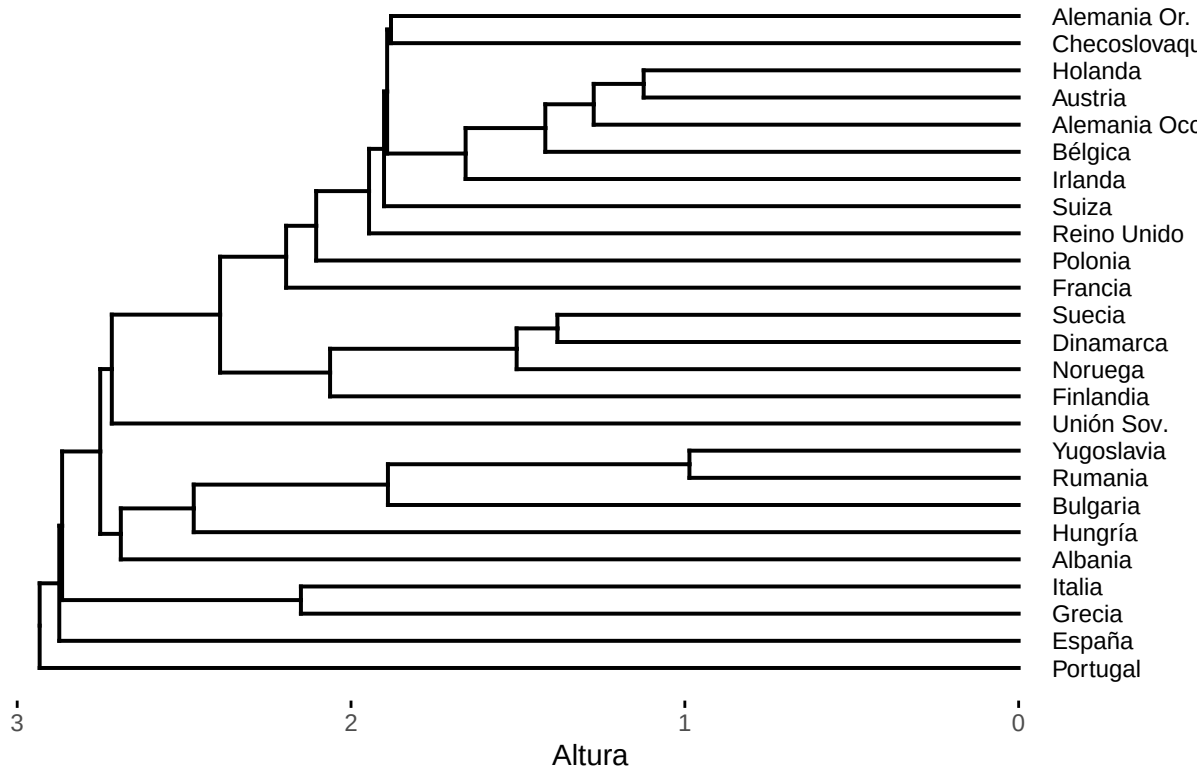
```
## [[1]]
```

Método de Ward



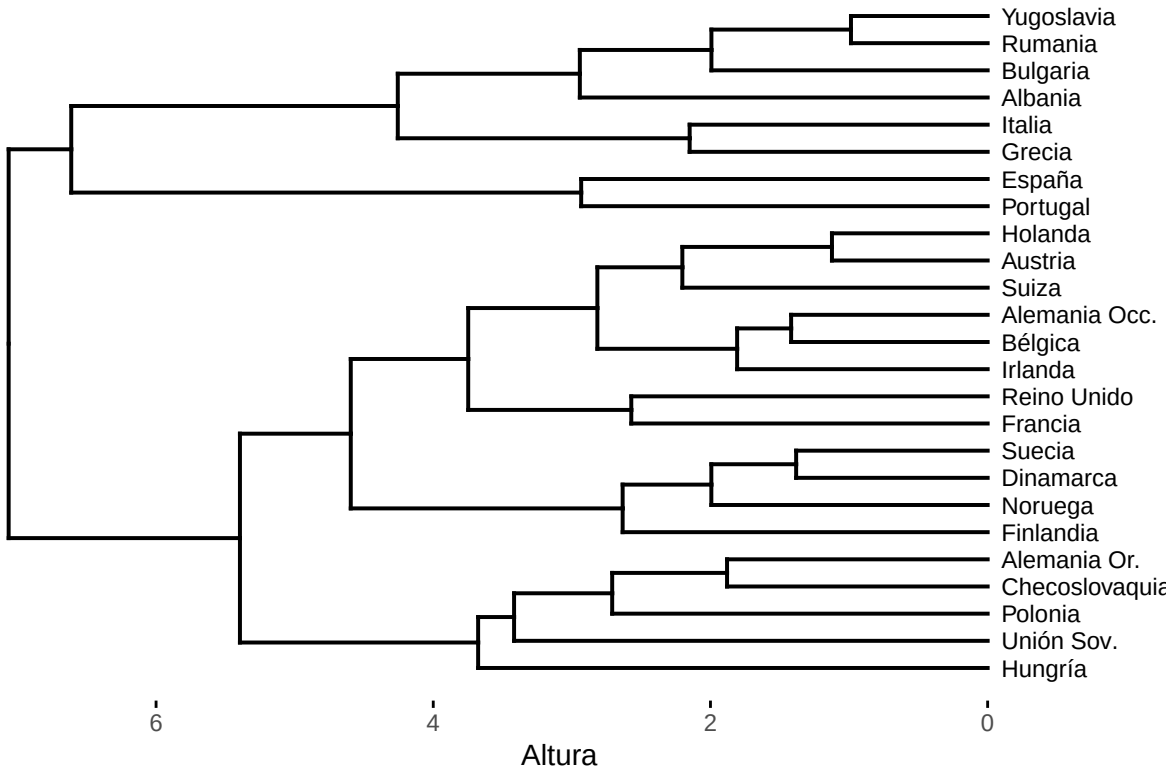

```
##
## [[2]]
```

Vecino más próximo

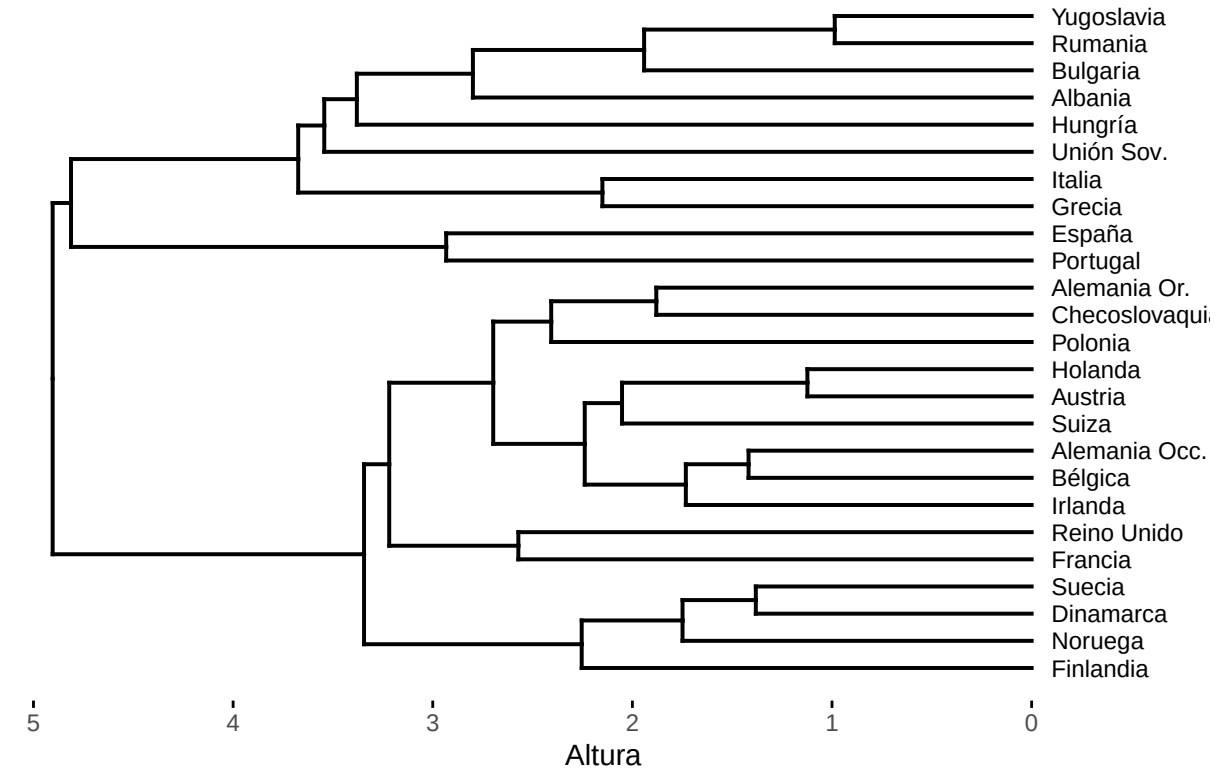



```
##
## [[3]]
```

Vecino más Lejano




```
##
## [[4]]
```



Método por disimilitud

Las medidas de disimilitud miden la distancia entre dos objetos de forma que, cuanto mayor sea su valor, más diferentes son los objetos y menor la probabilidad de que los métodos de clasificación los pongan en el mismo grupo.

El algoritmo construye los grupos a partir de uno solo que englobe a todas las n observaciones. Luego, se realizan diversas etapas en las que los conglomerados son divididos hasta que contenga una sola observación, seleccionando aquellos que tengan el diámetro más grande (la disimilitud).

Para dividir el cluster seleccionado en cada etapa, el algoritmo busca la observación con la disimilitud promedio más grande del grupo. Esta observación inicia el modo "grupo fragmentario", en donde el algoritmo reasignará las observaciones que están más cercanas a este grupo que al conglomerado inicial. De tal forma que al final de ese paso se han formado dos nuevos clusters.

Para calcularlo en R, usamos la función `cluster::diana()` para iniciar la clasificación jerárquica por disimilitud. Cabe señalar que además del vector, la función también brinda el coeficiente de disimilitud, análogo al de aglomeración.

```
library(cluster)

met.dis = diana(d.euc)

# Coeficiente de disimilitud
met.dis$dc
```

```
## [1] 0.6931511
```

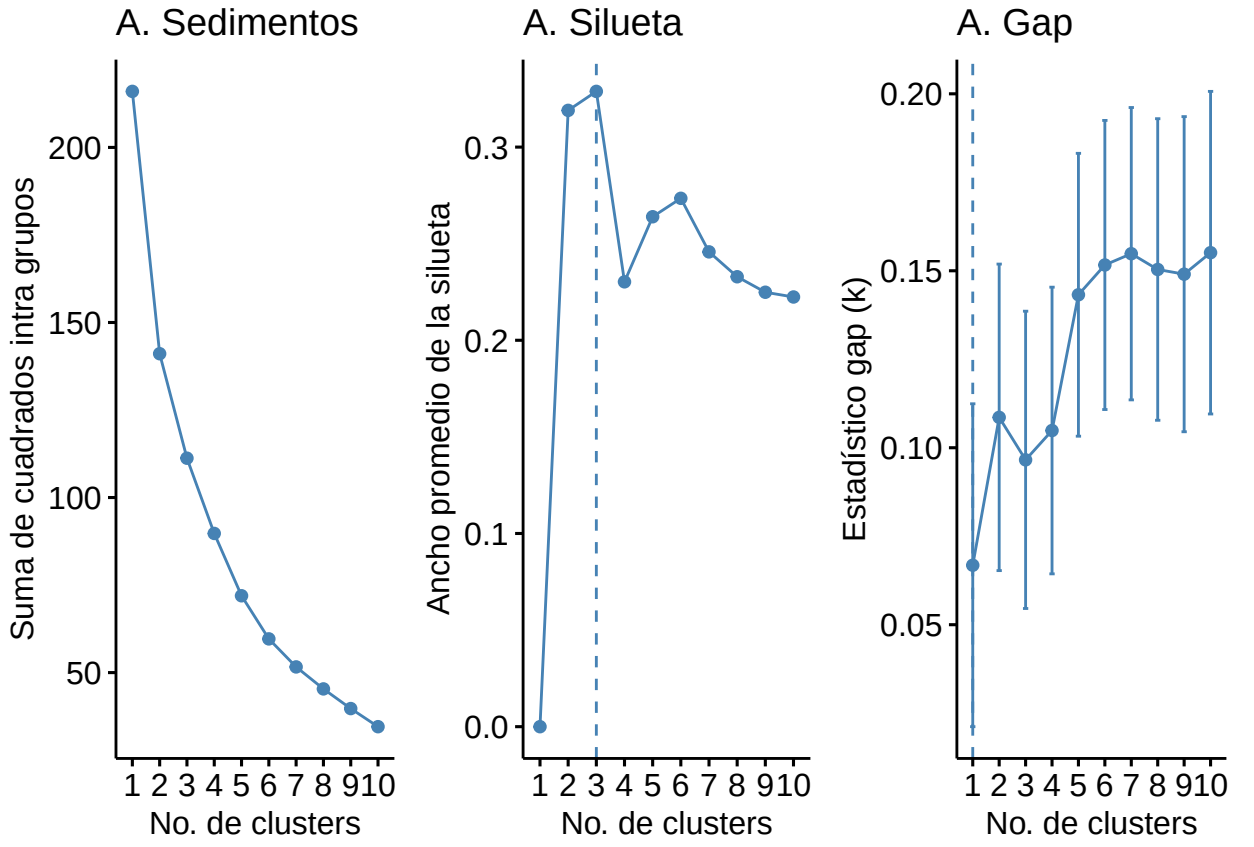
Podemos ver que el coeficiente de disimilitud es similar al método del vecino más lejano.

Número óptimo de clusters

Uno de los problemas que nos encontramos a la hora de aplicar alguno de los métodos de Clustering (K-means) es la elección del número de Clusters.

Aunque no exista un criterio objetivo para la selección del número de Clusters, existen técnicas para decidir cuántos cluster es recomendable utilizar. Tres de ellas son el método de sedimentos (o de codo), de silueta (Rousseeuw 1987) y del estadístico *gap* (Tibshirani, Walther, y Hastie 2001). Para obtenerlos, utilizamos la función `factoextra::fviz_nbclust()`.

```
g1 = fviz_nbclust(p.esc, FUN = hcut, method = "wss", k.max = 10) +
  labs(title = "A. Sedimentos", x = "No. de clusters", y = "Suma de cuadrados intra grupos")
g2 = fviz_nbclust(p.esc, FUN = hcut, method = "silhouette", k.max = 10) +
  labs(title = "A. Silueta", x = "No. de clusters", y = "Ancho promedio de la silueta")
g3 = fviz_nbclust(p.esc, FUN = hcut, method = "gap_stat", k.max = 10) +
  labs(title = "A. Gap", x = "No. de clusters", y = "Estadístico gap (k)")
gridExtra::grid.arrange(g1, g2, g3, nrow = 1)
```



Los tres métodos arrojan resultados distintos, pero tanto el método de sedimentos como el de silueta sugieren 3 clusters. Cabe señalar que SPSS utiliza el método silueta para decidir el número óptimo de grupos.

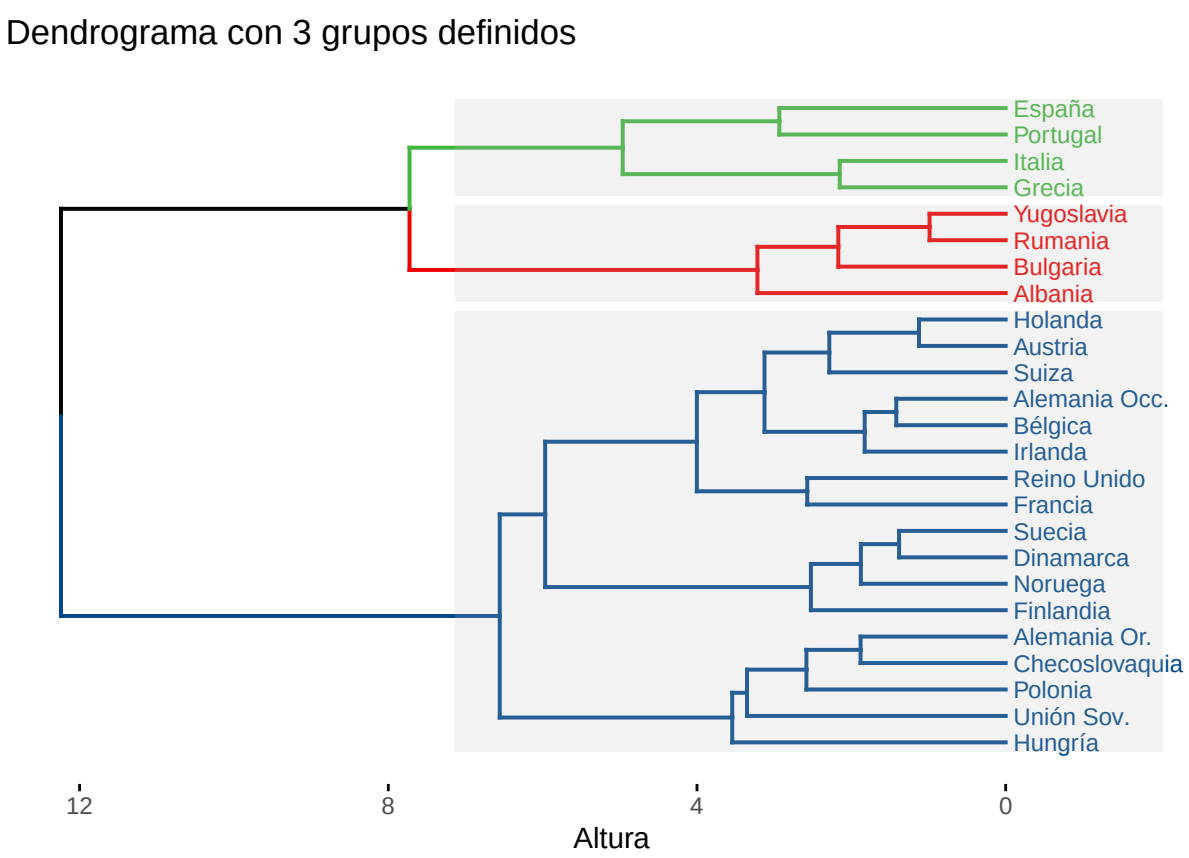
En adelante utilizaremos 3 grupos.

Visualización de grupos

Como ya hemos decidido, con la ayuda de diferentes criterios, prefijar en 3 el número de cluster, procedemos a visualizar con distancia euclídea y algoritmo de Ward los diferentes grupos.

```
fviz_dend(
  met.agl$ward.D2,
```

```
  horiz = TRUE,
  k = 3,
  rect = TRUE,
  rect_fill = TRUE,
  k_colors = "lancet",
  cex = 0.6,
  main = "Dendrograma con 3 grupos definidos",
  ylab = "Altura"
)
```



Observamos que en un grupo se encuentran los países asociados a la dieta mediterránea (`España`, `Portugal`, `Italia` y `Grecia`), los países balcánicos (`Yugoslavia`, `Rumania`, `Bulgaria` y `Albania`) y el resto.

Variables que más influyen

Derivado del dendrograma, usamos la función `cutree` para extraer los países de cada cluster. Posteriormente utilizaremos la función `aggregate` para calcular la tabla de pesos. Así podremos interpretar la importancia que tiene cada variable en los diferentes grupos de países.

```
grupos = cutree(met.agl$ward.D2, k = 3)

sapply(1:3, function(x) trimws(names(grupos[grupos == x])))
```

```
## [[1]]
## [1] "Albania"    "Bulgaria"   "Rumania"    "Yugoslavia"
##
## [[2]]
##  [1] "Austria"        "Bélgica"        "Checoslovaquia" "Dinamarca"
```

```
##  [5] "Alemania Or."   "Finlandia"      "Francia"        "Hungría"
##  [9] "Irlanda"        "Holanda"        "Noruega"        "Polonia"
## [13] "Suecia"         "Suiza"          "Reino Unido"    "Unión Sov."
## [17] "Alemania Occ."
##
## [[3]]
## [1] "Grecia"   "Italia"   "Portugal" "España"
```

En el cluster 1 están los balcánicos, el cluster 3 los mediterráneos y el resto en el cluster 2.

```
influencia = aggregate(p.esc, by = list(Cluster = grupos), mean)

lapply(1:3, function(x) sort(influencia[x, 2:10], decreasing = T))
```

```
## [[1]]
##   Cereales FrutosSecos FrutosyVegetales CarneRoja CarneBlanca   Pescado
## 1 1.720033   0.9961313       -0.6436044  -0.80757  -0.8719354 -1.038638
##       Leche   Feculas    Huevos
## 1 -1.078332 -1.423427 -1.553306
##
## [[2]]
##   CarneBlanca   Huevos     Leche   Feculas CarneRoja    Pescado
## 2   0.4660556 0.462539 0.4494997 0.3782653 0.3097346 0.01334646
##   FrutosyVegetales  Cereales FrutosSecos
## 2       -0.2319154 -0.435308  -0.5428273
##
## [[3]]
##   FrutosyVegetales FrutosSecos   Pescado  Cereales   Feculas    Huevos
## 3         1.629245    1.310885 0.9819154 0.1300253 -0.184201 -0.412485
##   CarneRoja      Leche CarneBlanca
## 3 -0.508802 -0.8320414   -1.108801
```

Vemos que los balcanes se caracterizan por consumir más cereales y frutos secos; los mediterráneos consumen más frutos y vegetales, frutos secos, pescado y cereales; el resto tiene una dieta rica en proteínas.

Veamos ahora el ANOVA de los grupos por cada variable

```
paises_clust = data.frame(p.esc, grupos)
ANOVA = aov(grupos ~ ., data = paises_clust)

summary(ANOVA)
```

```
##                  Df Sum Sq Mean Sq F value   Pr(>F)
## CarneRoja         1 0.0595  0.0595   0.716 0.410786
## CarneBlanca       1 0.0545  0.0545   0.656 0.430718
## Huevos            1 2.2668  2.2668  27.271 0.000103 ***
## Leche             1 0.1385  0.1385   1.667 0.216241
## Pescado           1 1.7290  1.7290  20.801 0.000375 ***
## Cereales          1 0.0520  0.0520   0.625 0.441428
## Feculas           1 0.0019  0.0019   0.023 0.881224
## FrutosSecos       1 0.8351  0.8351  10.047 0.006345 **
## FrutosyVegetales  1 1.6157  1.6157  19.437 0.000508 ***
## Residuals        15 1.2468  0.0831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aunque observamos que las variables `Huevos`, `Pescado`, `FrutosSecos` y `FrutosyVegetales` son las que más

influyen en el modelo, es necesario aclarar que, en este caso, las pruebas F sólo se deben utilizar con una finalidad descriptiva. Los niveles críticos no son corregidos, por lo que no pueden interpretarse como contrastes de hipótesis de igualdad entre los centros de los conglomerados.

K-medias

Descripción

El algoritmo de **K-medias** pertenece a los cluster **no jerárquicos**. Esto implica un conocimiento *a priori* del número de cluster, mediante el cual asignación de cada observación al cluster más cercano y una parada del método si no se produce reasignación o si la reasignación satisface la regla de parada.

El objetivo de este algoritmo es separar las observaciones en k-cluster, de manera que cada dato pertenezca únicamente a un grupo y se maximice la homogeneidad dentro de cada uno de ellos.

Obtención de K-medias

La función `kmeans()` recibe tres parámetros: `p.esc` (nuestros datos ya estandarizados), `centers` (número de grupos a formar), `nstart` (número de casos, en este caso países).

Con la posterior sentencia se obtiene el peso que tiene cada una de nuestras variables en cada uno de los 3 grupos.

Y para finalizar, vemos el tamaño que tiene cada uno de los 3 grupos: el grupo 1 lo conforman 4 países, el grupo 2 se compone de 6 países y el grupo 3 de 15 países.

```
k.medias = kmeans(p.esc, centers = 3, nstart = 25)

k.medias$centers
```

```
##    CarneRoja CarneBlanca     Huevos      Leche    Pescado   Cereales
## 1 -0.5088020  -1.1088009 -0.4124850 -0.8320414  0.9819154  0.1300253
## 2 -0.7901419  -0.5267887 -1.1655757 -0.9047559 -0.9504683  1.4383272
## 3  0.4517373   0.5063957  0.5762263  0.5837801  0.1183432 -0.6100043
##      Feculas FrutosSecos FrutosyVegetales
## 1 -0.1842010   1.3108846        1.6292449
## 2 -0.7604664   0.8870168       -0.5373533
## 3  0.3533068  -0.7043759       -0.2195240
```

```
k.medias$size
```

```
## [1]  4  6 15
```

Exploración de clusters

Primero agregamos una columna con el numero de cluster correspondiente a cada fila.

```
# Promedio de K-medias con respecto a los datos
pmd.p.km = aggregate(p.esc, by = list(Cluster = k.medias$cluster), mean)

pmd.p.km
```

```
##   Cluster  CarneRoja CarneBlanca     Huevos      Leche    Pescado
## 1       1 -0.5088020  -1.1088009 -0.4124850 -0.8320414  0.9819154
## 2       2 -0.7901419  -0.5267887 -1.1655757 -0.9047559 -0.9504683
## 3       3  0.4517373   0.5063957  0.5762263  0.5837801  0.1183432
##     Cereales    Feculas FrutosSecos FrutosyVegetales
## 1  0.1300253 -0.1842010   1.3108846        1.6292449
## 2  1.4383272 -0.7604664   0.8870168       -0.5373533
```

```
## 3 -0.6100043  0.3533068  -0.7043759       -0.2195240
```

A continuación, realizamos un Análisis de la Varianza de Clusters respecto a las variables.

```
paises.km = data.frame(p.esc, Cluster = k.medias$cluster)

sapply(colnames(paises.km)[1:9], function(x) {
  summary(
    aov(formula(paste0("Cluster~",x)), data = paises.km)
  )
})
```

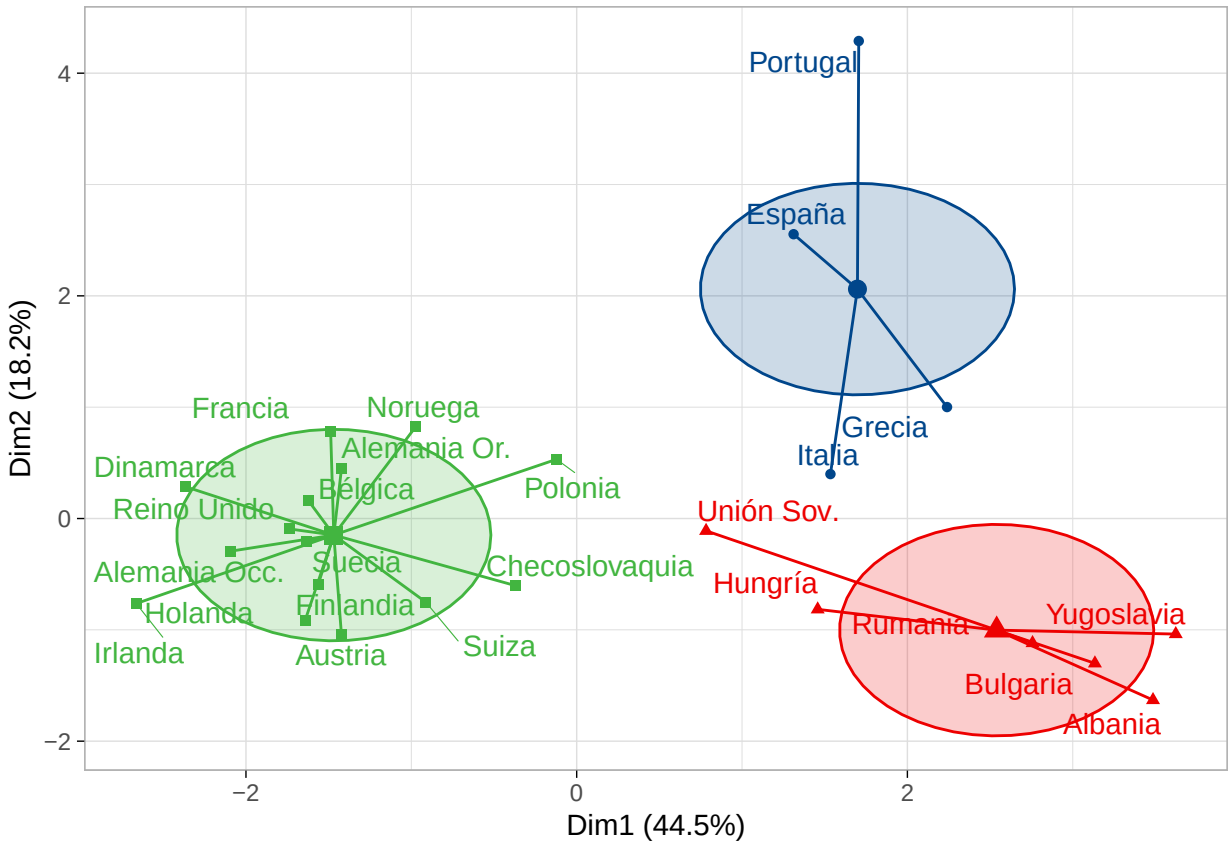
```
## $CarneRoja
##             Df  Sum Sq Mean Sq F value  Pr(>F)
## CarneRoja    1  3.2349  3.2349  6.8103 0.01567 *
## Residuals   23 10.9251  0.4750
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $CarneBlanca
##             Df Sum Sq Mean Sq F value    Pr(>F)
## CarneBlanca  1 6.0312  6.0312  17.065 0.0004062 ***
## Residuals   23 8.1288  0.3534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $Huevos
##             Df Sum Sq Mean Sq F value   Pr(>F)
## Huevos       1 4.4147  4.4147  10.419 0.003722 **
## Residuals   23 9.7453  0.4237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $Leche
##             Df Sum Sq Mean Sq F value    Pr(>F)
## Leche        1 6.0852  6.0852  17.333 0.0003748 ***
## Residuals   23 8.0748  0.3511
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $Pescado
##             Df  Sum Sq Mean Sq F value Pr(>F)
## Pescado      1  0.1931 0.19305  0.3179 0.5783
## Residuals   23 13.9669 0.60726
##
## $Cereales
##             Df  Sum Sq Mean Sq F value   Pr(>F)
## Cereales     1  3.8963  3.8963  8.7314 0.007103 **
## Residuals   23 10.2637  0.4462
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $Feculas
##             Df  Sum Sq Mean Sq F value Pr(>F)
## Feculas      1  1.5183 1.51826  2.7623 0.1101
```

```
## Residuals   23 12.6417 0.54964
##
## $FrutosSecos
##             Df  Sum Sq Mean Sq F value    Pr(>F)
## FrutosSecos  1 10.4138 10.4138  63.935 4.327e-08 ***
## Residuals   23  3.7462  0.1629
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $FrutosyVegetales
##                  Df  Sum Sq Mean Sq F value  Pr(>F)
## FrutosyVegetales  1  4.0097  4.0097  9.0858 0.00618 **
## Residuals        23 10.1503  0.4413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos cómo `FrutosSecos` tiene mucho más peso que el resto de los alimentos. Recordemos que esta tabla nos aporta solamente descripción informativa en este análisis.

Visualización de K-medias

```
fviz_cluster(k.medias,
             data = p.esc,
             palette = "lancet",
             ellipse.type = "euclid",
             star.plot = T,
             repel = T,
             ggtheme = theme_light()) +
  theme(legend.position = "none",
        plot.title = element_blank())
```



Finalmente, el gráfico de centralidades refleja las distancias euclídeas entre cada caso o su centro, de cada uno de cluster. También refleja las diferencias que reportan los países con base en las variables del estudio. Estos datos se obtienen de la tabla de pesos, calculada anteriormente.

Como hemos visto anteriormente con el dendrograma, el primer grupo lo componen los *Países Mediterráneos*, el segundo los *Balcánicos* y el tercero lo conforman el *resto* de países de nuestro estudio (Centro Europa).

Método de PAM

El método Partición Alrededor de Medoides (PAM), propuesto por Park y Jun (2009), al igual que K-Means, se encuentra en el grupo de métodos de agrupamiento por particiones, y en el cual se utiliza medianas en vez de medias con el objetivo de limitar la influencia de los atípicos.

Para encontrar k clusters, el modelo PAM determina un objeto representativo para cada clúster. Este objeto representativo, llamado "medoid", es el que se encuentra localizado más al centro dentro del clúster. Una vez que los medoids han sido seleccionados, cada objeto no seleccionado es agrupado con el medoid al cual es más similar. De este modo, la partición se realiza alrededor de medoides.

Descripción

Inicialmente determinamos el número de clusters a formar, en este caso 3 agrupaciones. En cada iteración, se realiza un intercambio entre un objeto seleccionado y un objeto no seleccionado si y solo si el intercambio resulta en un incremento de la calidad del agrupamiento (clustering).

Para calcularlo en R, usamos `cluster::pam()`

```
met.pam = pam(p.esc, k = 3)

met.pam$medoids
```

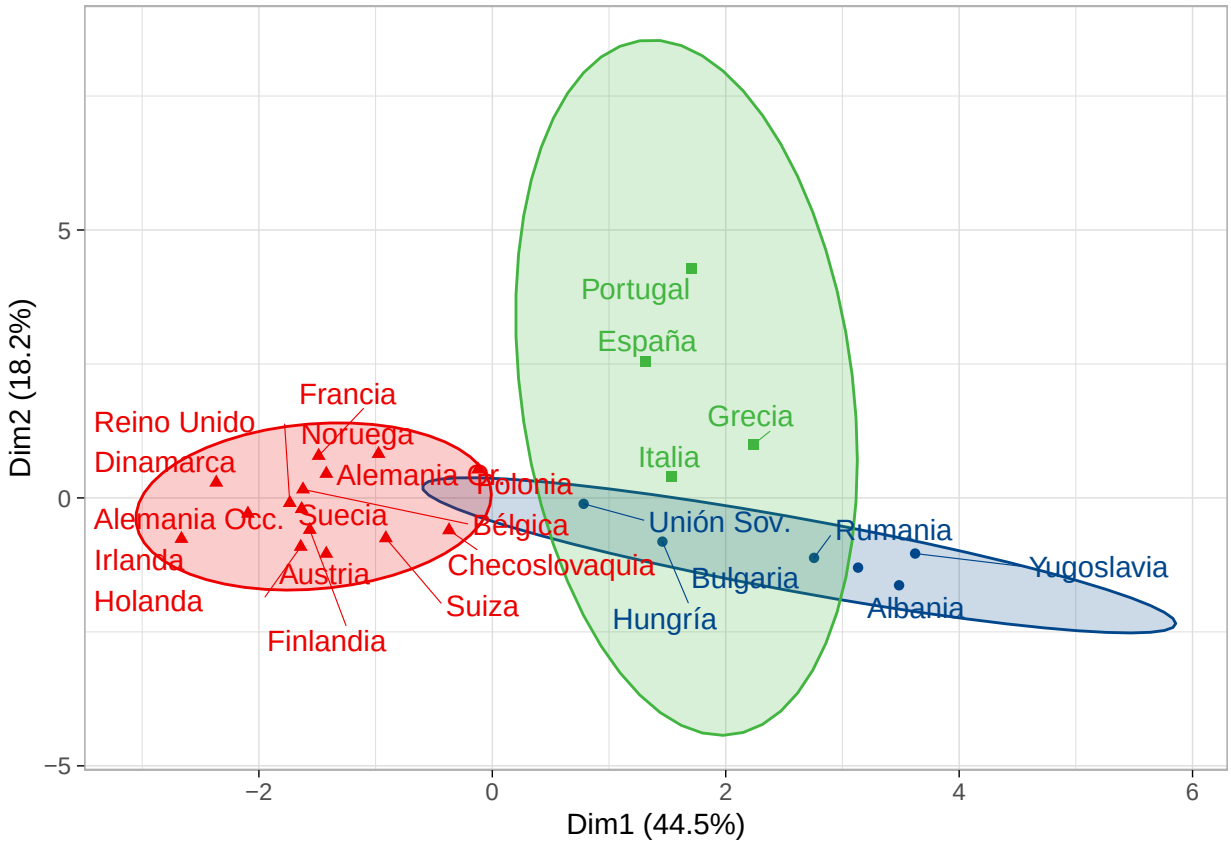
```
##                CarneRoja CarneBlanca     Huevos       Leche     Pescado
## Rumania        -1.0839304  -0.4320425 -1.2848772 -0.84611516 -0.96516320
## Bélgica         1.0970762   0.3800675  1.0415022  0.05460623  0.06348211
## España         -0.8150392  -1.2170822  0.1467409 -1.19795945  0.79822876
##                  Cereales    Feculas FrutosSecos FrutosyVegetales
## Rumania         1.5810786 -0.7196689   1.1220326      -0.74061625
## Bélgica        -0.5146342  0.8714358  -0.4895043      -0.07539207
## España         -0.2777275  0.8714358   1.4241958       1.69853906
```

Podemos observar que PAM de `k = 3` coloca a `Rumania`, `Bélgica` y `España` como los `medoids` (i.e., los centros) de sus respectivos grupos.

Visualización de mediodes

Para obtener una representación gráfica del clustering, se ha empleado en este caso la función `factoextra::fviz_cluster()`.

```
fviz_cluster(met.pam,
             palette = "lancet",
             ellipse.type = "t",
             repel = T,
             ggtheme = theme_light()) +
  theme(legend.position = "none",
        plot.title = element_blank())
```



Los clusters muestran claramente su formación alrededor de los medoides y no centralidades como en K-means. Aunque los resultados son similares, mediante el método PAM se visualiza con mayor especificidad los grupos conformados.

Ventajas y desventajas del método PAM

El método PAM es un método de clustering más robusto que K-means, por tanto es más adecuado para datos que contengan atípicos o ruido.

No obstante, al igual que K-means, necesita que se especifique de antemano el número de clusters que se van a conformar. Esto puede ser complicado de determinar si no se dispone de los grupos *a priori* del análisis.

Finalmente, si bien su funcionamiento es adecuado con datos de n pequeñas, no es escalable debido a su complejidad computacional.

Park, Hae-Sang, y Chi-Hyuck Jun. 2009. «A simple and fast algorithm for K-medoids clustering». *Expert Systems with Applications* 36 (2, Part 2): 3336-41. https://doi.org/https://doi.org/10.1016/j.eswa.2008.01.039.

Rousseeuw, Peter J. 1987. «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis». *Journal of Computational and Applied Mathematics* 20: 53-65. https://doi.org/https://doi.org/10.1016/0377-0427(87)90125-7.

Tibshirani, Robert, Guenther Walther, y Trevor Hastie. 2001. «Estimating the number of clusters in a data set via the gap statistic». *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 411-23. https://doi.org/10.1111/1467-9868.00293.