Técnicas de validación de clusters
Un ejemplo con iris data set

Antoni Enric Heinrichs Maquilón, Marco Sánchez Beeckman, Adrián Tobar Nicolau

14 de enero de 2019

Contents

Resumen

En este documento presentamos diferentes técnicas de validación de procesos de clustering y las aplicamos sobre el conjunto de datos `iris`.

Comenzamos estudiando la tendencia del clustering, donde estudiamos la uniformidad de los datos tanto de forma visual como con métodos estadísticos, y determinamos que en `iris` se puede realizar un clustering significativo.

Continuamos calculando el número óptimo de clusters, de nuevo tanto visualmente como a través de estadísticos, y decidimos esta cifra sobre `iris` a partir de las decisiones de 30 métodos distintos, donde acabamos por quedarnos con 3 clusters.

Una vez determinado este número, validamos la calidad del clustering a partir de la compacidad, separación y conectividad de este, y comparando los resultados obtenidos con la clasificación por especies de las flores. Obtenemos que tanto el clustering por k-medias como el jerárquico y PAM dan resultados válidos con 3 clusters en `iris`.

Finalmente, comparamos los distintos algoritmos de clustering que han resultado válidos, basándonos en medidas internas del clustering y en su estabilidad. Concluimos que en `iris` no hay un algoritmo de 3 clusters superior al resto, sino que este se debe elegir según las medidas que se quieran minimizar. Usando otro número de clusters, destacamos el jerárquico de 2 clusters y el PAM de 6.

Planteamiento del problema

Este trabajo forma parte del estudio práctico de las técnicas de clustering vistas en clase y junto al resto de informes aporta una visión general sobre este campo.

En este informe nos centraremos en las técnicas para la validación de clusters, esto incluye analizar la tendencia de cluster de los datos, determinar el número óptimo de clusters, validar estos clusters mediante diferentes estadísticos y elegir el mejor algoritmo de cluster según los datos que tratemos. Estos apartados se corresponen con los capítulos 11-14 del libro `Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning` de Alboukadel Kassambara.

Los datos en este caso serán principalmente el dataset incluido en R `iris`, el cual contiene 150 muestras de flores iris y las diferentes medidas de sus pétalos y sépalos.

Entre los paquetes de `R` que usaremos se encuentran `factoextra` para visualizar datos y llevar a cabo PCAs, `clustertend` para evaluar la tendencia del cluster, `NBClust` que contiene 30 métodos diferentes para determinar el número óptimo de clusters, `fpc` para calcular diferentes estadísticos de validación de clusters y `clValid`, un paquete que compara diferentes algoritmos de clustering para un mismo conjunto de datos.

Visualización de los datos y evaluación de la tendencia de cluster

Empezamos cargando y estandarizando los datos con los que vamos a trabajar. Por un lado tenemos el dataset `iris`, y por otro, un conjunto de datos aleatorios generados a partir de él. En este último, los valores para cada una de las variables siguen una distribución uniforme entre el mínimo y el máximo observados en el conjunto de datos `iris` original.

```
#Establecemos una semilla para poder reproducir los resultados
set.seed(123)

# Guardamos el data set iris, excluyendo la variable Species
df <- iris[, -5]

# Generamos 150 muestras aleatorias
random_df <- apply(df, 2,
                   function(x){runif(length(x), min(x), (max(x)))})

#Guardamos los datos en un data frame
random_df <- as.data.frame(random_df)

# Normalizamos los datos
df <- scale(df)
random_df <- scale(random_df)

#Visualizamos las 3 primeras filas de cada uno
head(df,3)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.8976739   1.0156020    -1.335752   -1.311052
## [2,]   -1.1392005  -0.1315388    -1.335752   -1.311052
## [3,]   -1.3807271   0.3273175    -1.392399   -1.311052
```

```
head(random_df,3)
```

```
##      Sepal.Length  Sepal.Width Petal.Length Petal.Width
## [1,]   -0.7542653  1.282313529    1.0166309  0.05987618
```

```
## [2,]    0.9894828  0.008998208   -1.6739056 -0.52719066
## [3,]   -0.3315006 -0.389882189    0.9975077 -0.89582148
```

Vemos como en ambos conjuntos tenemos las mismas 4 variables que han sido normalizadas.

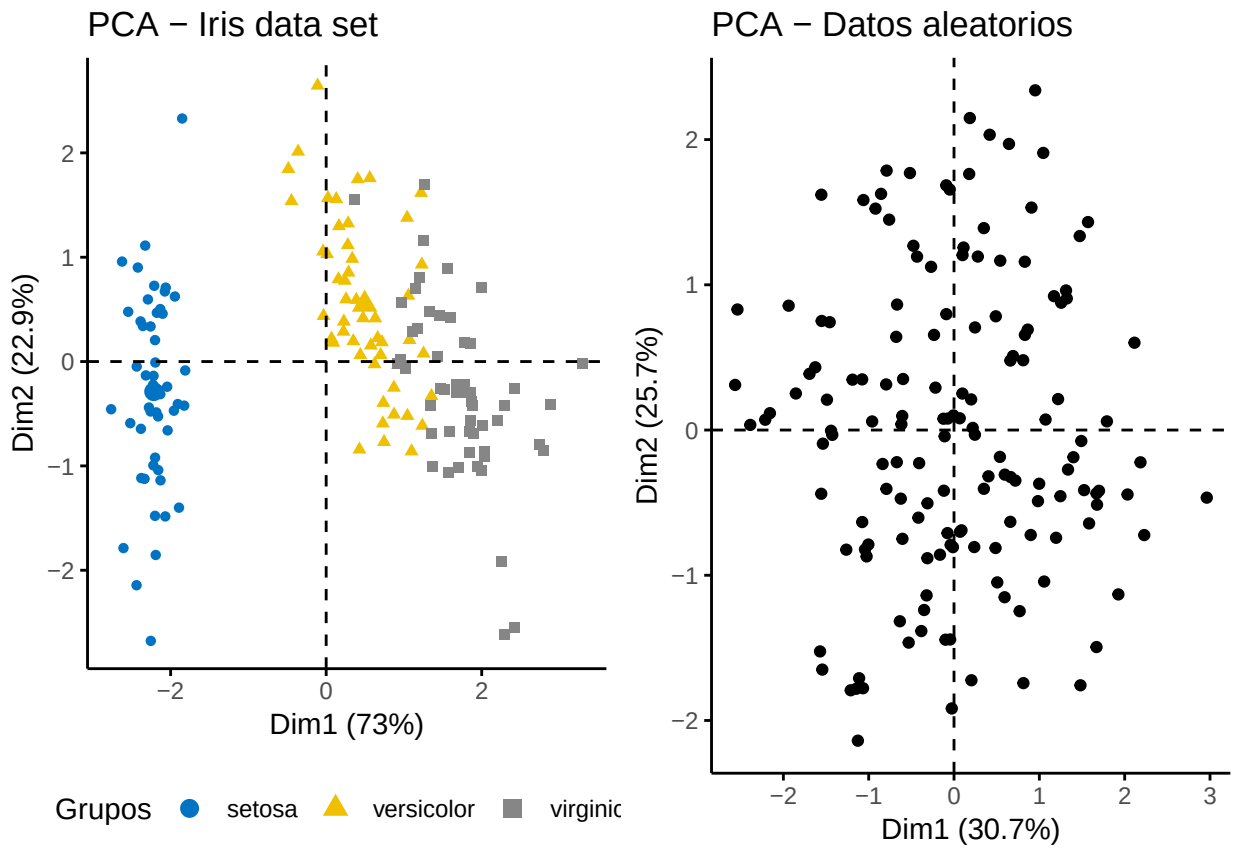
La primera tarea que debemos llevar a cabo es comprobar si los datos que tenemos son apropiados para agruparlos en clusters, esto es, que no sean agrupaciones aleatorias.

Para ello, primero vamos a tener que reducir la dimensionalidad del problema mediante un análisis de componentes principales, para poder representarlos en un diagrama de dispersión.

```
# Generamos el gráfico del los datos Iris
fviz_pca_ind(prcomp(df), title = "PCA - Iris data set",
             habillage = iris$Species, palette = "jco", # Coloreamos por especie
             geom = "point", ggtheme = theme_classic(),
             legend = "bottom", legend.title = "Grupos") -> p1

# Repetimos para los datos aleatorios
fviz_pca_ind(prcomp(random_df), title = "PCA - Datos aleatorios",
             geom = "point", ggtheme = theme_classic()) -> p2

# Los mostramos juntos
grid.arrange(p1, p2, ncol = 2)
```



A partir de la visualización anterior podemos ver que en el caso de `iris` hay dos claros clusters (los puntos azules por un lado y el resto en otro) e incluso podríamos distinguir 3. En cambio, en los datos aleatorios no intuimos una división tan clara. Estos primeros gráficos ya muestran como los primeros datos son más adecuados para llevar a cabo clustering que los segundos.

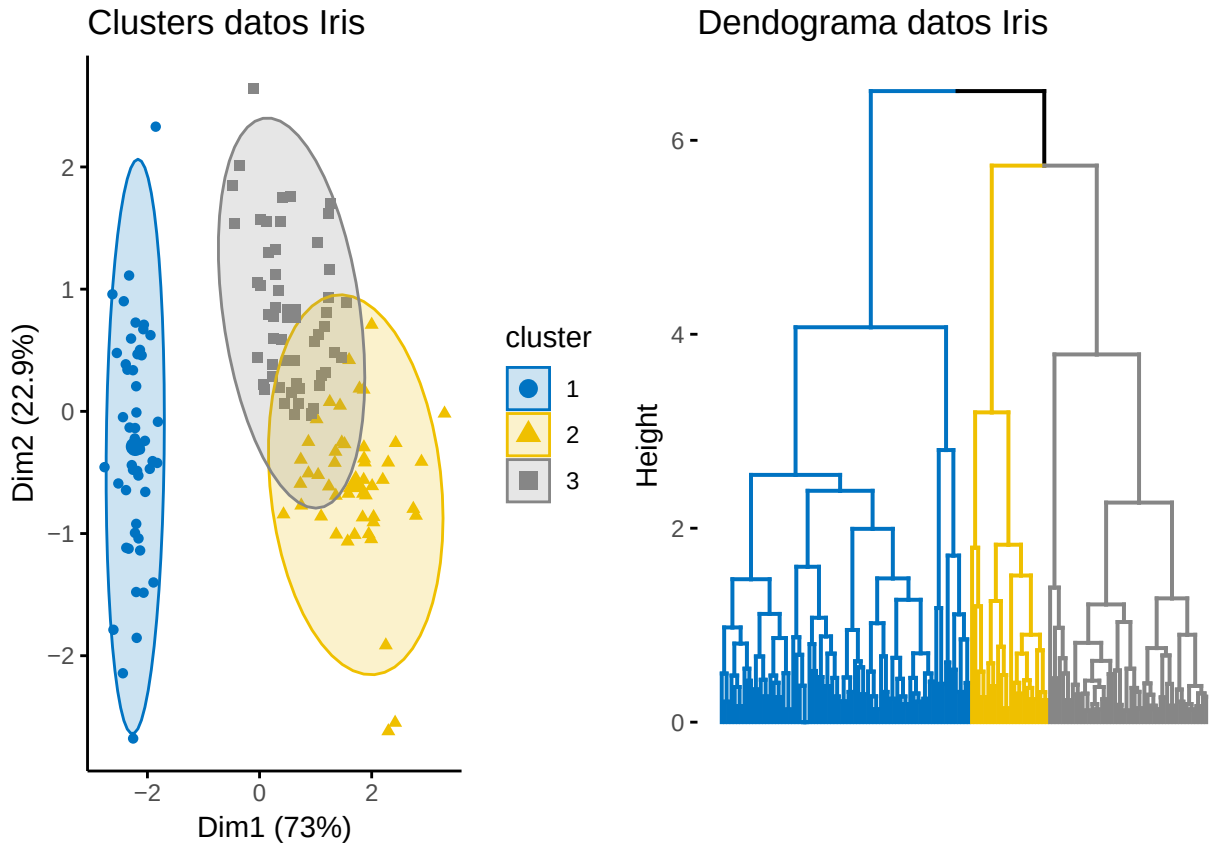
Veamos qué pasa si aplicamos cluster de 3-medias y cluster jerárquico a ambos conjuntos de datos:

```
set.seed(123)

# k-medias con 3 clusters con los datos Iris
km.res1 <- kmeans(df, 3)
fviz_cluster(list(data = df, cluster = km.res1$cluster),
             ellipse.type = "norm", geom = "point", stand = FALSE,
             palette = "jco", ggtheme = theme_classic(),
             main = "Clusters datos Iris") -> i1

# Cluster jerárquico con los datos Iris
fviz_dend(hclust(dist(df)), k = 3, k_colors = "jco",
          as.ggplot = TRUE, show_labels = FALSE,
          main = "Dendograma datos Iris") -> i2

# Mostramos los dos gráficos juntos
grid.arrange(i1, i2, ncol = 2)
```

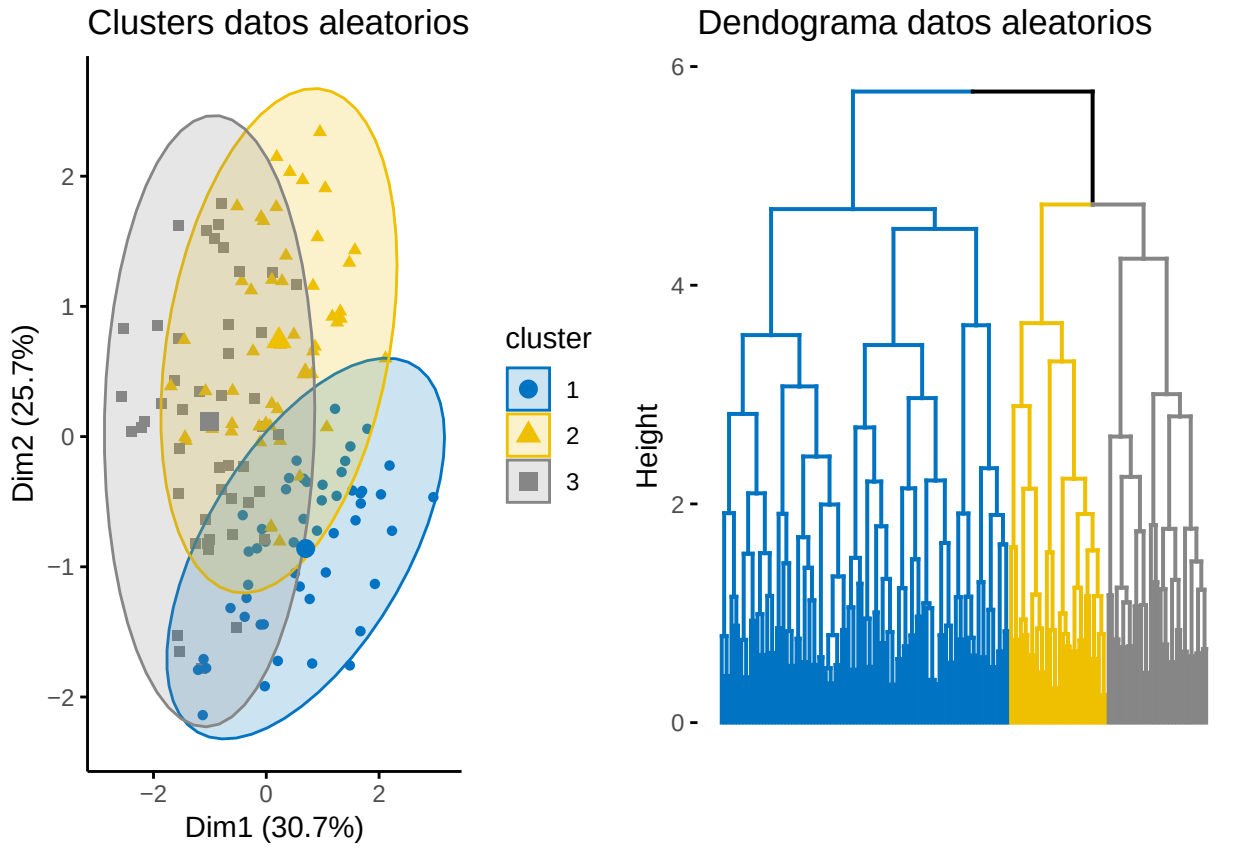



```
# k-medias con 3 clusters con los datos aleatorios
km.res2 <- kmeans(random_df, 3)
fviz_cluster(list(data = random_df, cluster = km.res2$cluster),
             ellipse.type = "norm", geom = "point", stand = FALSE,
             palette = "jco", ggtheme = theme_classic(),
             main = "Clusters datos aleatorios") -> p3

# Cluster jerárquico con los datos aleatorios
```

```r
fviz_dend(hclust(dist(random_df)), k = 3, k_colors = "jco",
          as.ggplot = TRUE, show_labels = FALSE,
          main = "Dendograma datos aleatorios") -> p4

# Mostramos los gráficos juntos
grid.arrange(p3, p4, ncol = 2)
```



En ambos casos, al aplicar algoritmos de clustering se han formado clusters independientemente de la naturaleza de los datos y de si las agrupaciones tienen sentido. Esto nos muestra la necesidad de evaluar los datos antes de tratarlos, ya que los métodos que usamos siempre nos van a dar un resultado, significativo o no. Vista esta necesidad, veamos dos métodos para evaluar la tendecia de clusters.

Métodos para la evaluación de clusters

Método de Hopkins

El estadístico de Hopkins mide la probabilidad de que un conjunto de datos siga una distribución uniforme. En este caso definimos las hipótesis como:

$$\begin{cases} H_0: & \text{El conjunto de datos se distribuye de forma uniforme} \\ H_1: & \text{El conjunto de datos no se distribuye de forma uniforme (contiene clusters significativos)} \end{cases}$$

El estadístico de Hopkins se calcula como:

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

Donde $\{x_i\}_{i=1}^n$ es la distancia entre la i-ésima observación y su vecino más cercano del conjunto estudiado y $\{y_i\}_{i=1}^n$ es la distancia entre el i-ésimo dato y su vecino más cercano de un conjunto de datos generado a partir del conjunto original tal y como hemos hecho en el apartado anterior.

Si el valor de H es cercano a 0.5, significa que el conjunto estudiado se distribuye de forma uniforme ya que $\sum_{i=1}^n x_i$ y $\sum_{i=1}^n y_i$ son parecidos. Por otra parte, si H es cercano a 0, podemos rechazar la hipótesis nula.

En R el estadístico mediante la siguiente función:

```
set.seed(123)

#Calculamos H para Iris
hopkins(df, n = nrow(df)-1)
```

```
## $H
## [1] 0.1815219
```

```
set.seed(123)

#Calculamos H para los datos aleatorios
hopkins(random_df, n = nrow(random_df)-1)
```

```
## $H
## [1] 0.5344466
```

Los resultados de ambos tests no nos soprenden: para los datos aleatorios tenemos un valor de H cercano a 0.5 ya que estos han sido creados a partir de distribuciones uniformes; para `iris` tenemos un valor suficientemente alejado de 0.5 como para rechazar la hipótesis nula. Para finalizar la evaluacion de los datos, veamos si con un método visual llegamos a las mismas conclusiones.

Método visual

Alternativamene podemos estudiar los datos visualmente con el algoritmo VAT (Visual Assessment of cluster Tendency). Este algoritmo presenta 3 pasos:

1. Calcula la matriz de desemejanzas (DM) entre las observaciones del conjunto estudiado usando la distancia euclidea.
2. Reordena esta matriz para que las observaciones más similares estén más cercanas. Así se obtiene la matriz de desemejanzas ordenada (ODM).
3. Muestra esta matriz como una imagen de desemejanzas ordenada (ODI), en la que las partes en rojo presentan alta similitud, y las azules baja.

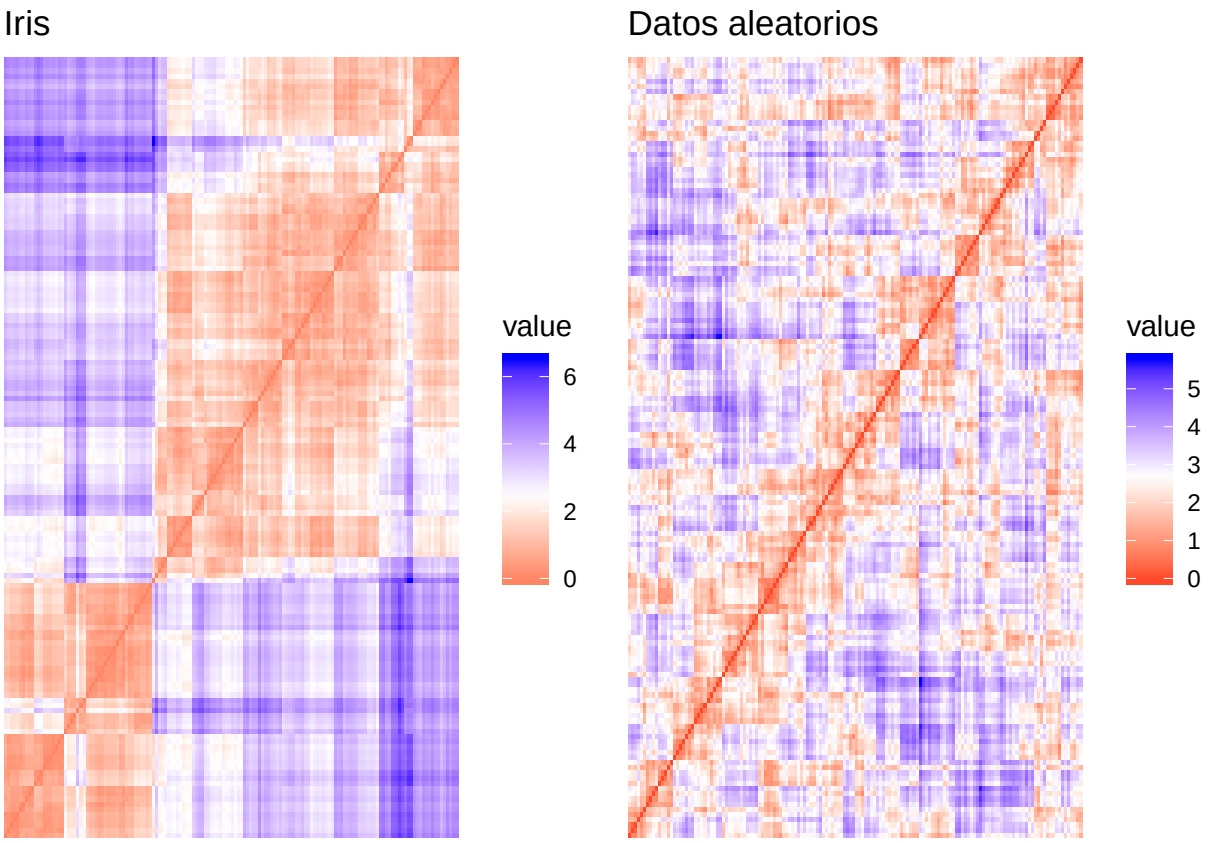
El criterio se reduce a estudiar la imagen. Si podemos distinguir claramente diferentes bloques cuadrados oscuros hay una tendencia de cluster más clara que en el caso de un patrón más complejo.

Obtenemos la ODI para los dos conjuntos que estudiamos:

```
# ODI para los datos Iris
fviz_dist(dist(df), show_labels = FALSE) +
  labs(title = "Iris") -> t1

# ODI para los datos aleatorios uniformes
fviz_dist(dist(random_df), show_labels = FALSE) +
  labs(title = "Datos aleatorios") -> t2

# Imágenes juntas
grid.arrange(t1, t2, ncol = 2)
```



En la ODI de `iris` vemos dos bloques rojos muy diferenciados mientras en la ODI de los datos aleatorios no distinguimos bloques. Con esta salida y todo lo visto anteriormente, podemos concluir que los datos `iris` son apropiados para agruparlos en clusters.

El siguiente aspecto a tratar es el número de clusters que debemos usar para agrupar los datos.

Determinación del número óptimo de clusters

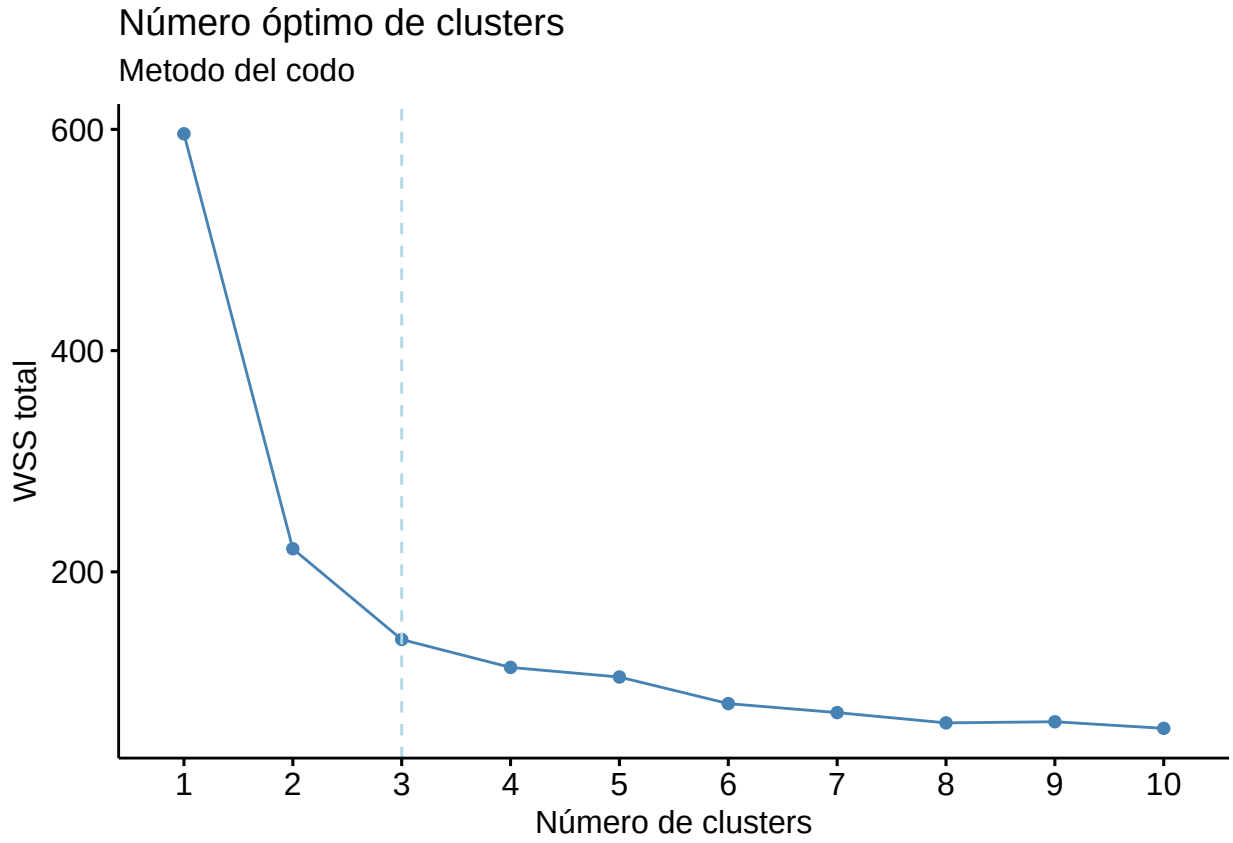
Una vez tenemos claro que los datos son apropiados para aplicar algoritmos de clustering, necesitamos conocer el número de clusters que vamos a usar. Para determinar esta cifra contamos con métodos directos como el método del codo o el método de la silueta y métodos de tests estadísticos como el estadístico del hueco (`gap statistic`). Además, R nos proporciona 30 tests diferentes que nos permiten decidir el número de clusters según lo que diga la mayoría de estos.

Método del codo

Este método consiste en calcular la variación total dentro de los clusters (WSS total) para diferentes números de clusters usando un algoritmo de clustering (k-medias en nuestro caso). A continuación representamos la curva que compara la WSS con el número de clusters y elegimos como número óptimo de clusters la localización del "codo", esto es, el punto en el que la curva empieza a decrecer de forma más lenta.

En R, usamos la función `fviz_nbclust` que permite calcular el número óptimo de clusters para diferentes métodos y particiones:

```r
# Método del codo
fviz_nbclust(df, kmeans, method = "wss") +
  #Indicamos la posición en la que situamos el codo
  geom_vline(xintercept = 3, linetype = 2, colour="light blue") +
  labs(subtitle = "Metodo del codo", title = "Número óptimo de clusters",
       x = "Número de clusters", y = "WSS total")
```



Vemos como el descenso más brusco se produce en el primer tramo y después se ralentiza a medida que aumenta la k. A partir de $k = 3$, la pendiente es muy reducida, de modo que el número óptimo de clusters siguiendo el método del codo es 3.

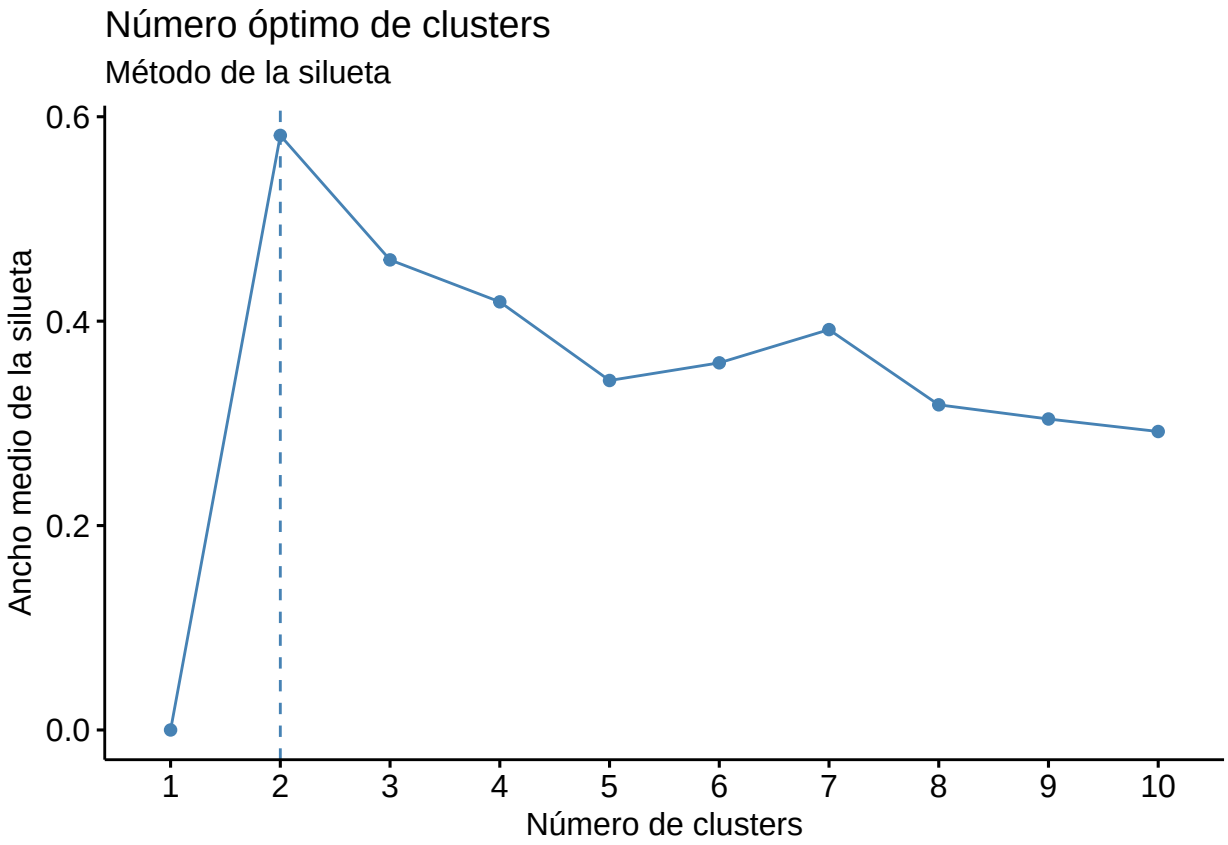
Este método puede resultar ambiguo, así que como alternativa podemos utilizar otro método visual llamado el de la silueta media.

Método de la silueta media

De forma breve podemos decir que este método mide la calidad del clustering para diferente número de clusters y se queda con aquel que obtenga un mayor coeficiente. Veremos más en detalle el cálculo de este coeficiente en la sección de validación del cluster.

Usando la misma función que en el apartado anterior podemos mostrar la curva que relaciona cada k con su índice:

```r
# Método de la silueta
fviz_nbclust(df, kmeans, method = "silhouette") +
  labs(subtitle = "Método de la silueta", title = "Número óptimo de clusters",
       x = "Número de clusters", y = "Ancho medio de la silueta")
```



De la gráfica anterior vemos que el óptimo determinado por el método de la silueta media para el dataset `iris` es de 2 clusters.

Método estadístico del hueco

El método `gap statistic` compara la variación total W_k dentro de cada cluster para un número fijo $k \in \{1, \dots, k_{max}\}$ de clusters con B conjuntos de datos de referencia (con distribución uniforme aleatoria).

Para cada conjunto $b \in \{1, \dots, B\}$ de datos con una distribución uniforme aleatoria, agrupamos en k clusters y calculamos la variación total dentro de cada cluster W_{kb}.

Ahora, calculamos el estadístico como

$$Gap(k) = \frac{1}{B} \sum_{b=1}^{B} \log(W_{kb}) - \log(W_k)$$

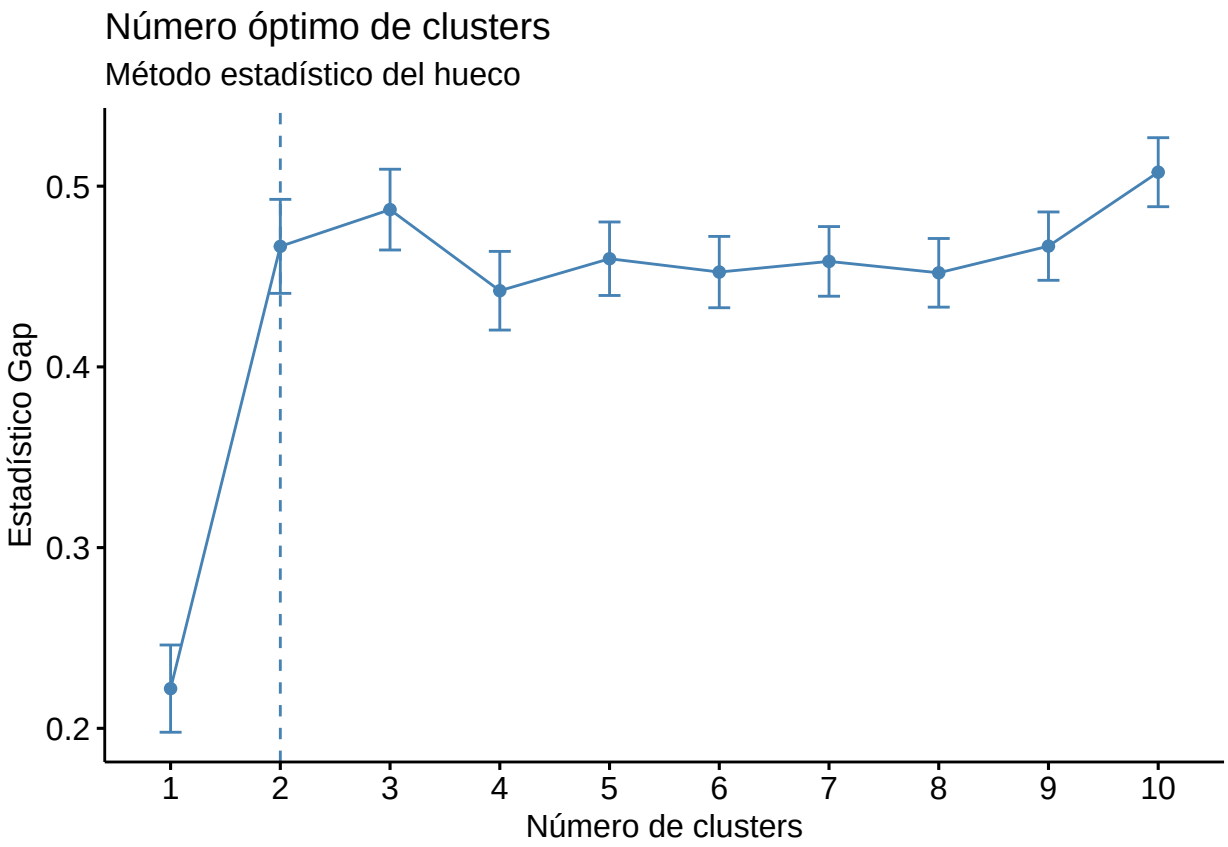
Además también calculamos la desviación estándar de este estadístico.

Finalmente, elegimos el valor k más pequeño tal que $Gap(k) \geq Gap(k+1) - s_{k+1}$.

La función `fviz_nbclust` también permite representar los valores del hueco por cada k, y marca el número de clusters que este método considera óptimo.

```
# Estadístico del hueco
# Para esta función se recomienda n=50 para conservar la rapidez
# Preferimos usar n=500 puesto que aporta mayor precisión
# El parámetro verbose = FALSE oculta el progreso de computación
set.seed(123)
```

```r
fviz_nbclust(df, kmeans, nstart = 25, method = "gap_stat", nboot = 500, verbose = FALSE) +
  labs(subtitle = "Método estadístico del hueco", title = "Número óptimo de clusters",
       x = "Número de clusters", y = "Estadístico Gap")
```



Vemos como este método también nos indica que 2 es el número de clusters óptimo para el data set `iris`.

Decisión por mayoría

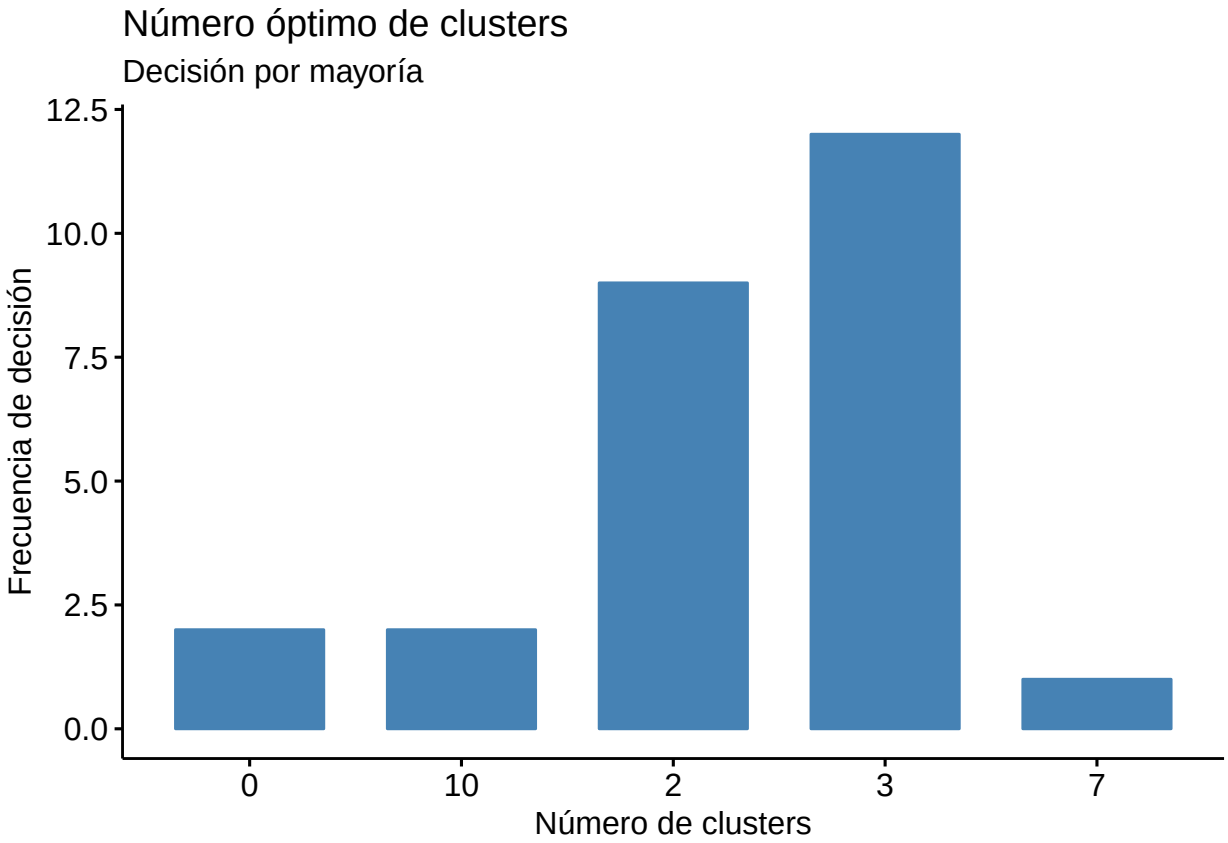
Podemos usar la función `NBClust` para decidir entre 30 criterios cuántos clusters debemos usar:

```r
nb <- NbClust(df, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")
```

```r
# Frecuencias de decisión de 30 diferentes métodos
fviz_nbclust(nb) +
  labs(title = "Número óptimo de clusters", subtitle = "Decisión por mayoría",
       x = "Número de clusters",
       y = "Frecuencia de decisión")
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 9 proposed  2 as the best number of clusters
## * 12 proposed  3 as the best number of clusters
## * 1 proposed  7 as the best number of clusters
## * 2 proposed  10 as the best number of clusters
##
```

```
## Conclusion
## ========================
## * According to the majority rule, the best number of clusters is  3 .
```



Como 12 de los 30 métodos eligen $k = 3$ como el número óptimo de clusters y estos forman una mayoría, decidimos quedarnos con este resultado. A pesar de esto, vemos que bastantes criterios (9 de ellos) han elegido 2 como la cantidad óptima.

Ahora que ya hemos determinado el número de clusters llega el momento de validar este clustering.

Validación del cluster

Una vez hemos decidido en cuántos clusters vamos a dividir los datos, tenemos que saber si esta división es buena. Generalmente, esta validación la podemos hacer de tres formas: de forma interna, que solo utiliza la información del proceso de clustering utilizado; de forma externa, que compara los resultados con otro resultado conocido; o de forma relativa, que evalúa los resultados haciendo variar parámetros dentro del propio algoritmo utilizado. Nos centraremos en las dos primeras.

Validación interna

Este tipo de validación tiene en cuenta la compacidad, separación, y conectividad de los clusters, que miden respectivamente la proximidad de los objetos dentro de un mismo cluster, cómo de separados están los clusters entre sí, y la relación entre los clusters de objetos vecinos.

Con estos criterios, definimos el ancho de silueta S_i de una observación i como

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

Donde a_i es la desemejanza media entre i y todos los demás puntos del cluster al que pertenece i, y b_i es el mínimo de las desemejanzas entre i y los demás clusters (calculadas como la media de las desemejanzas entre i y todos los elementos del cluster).

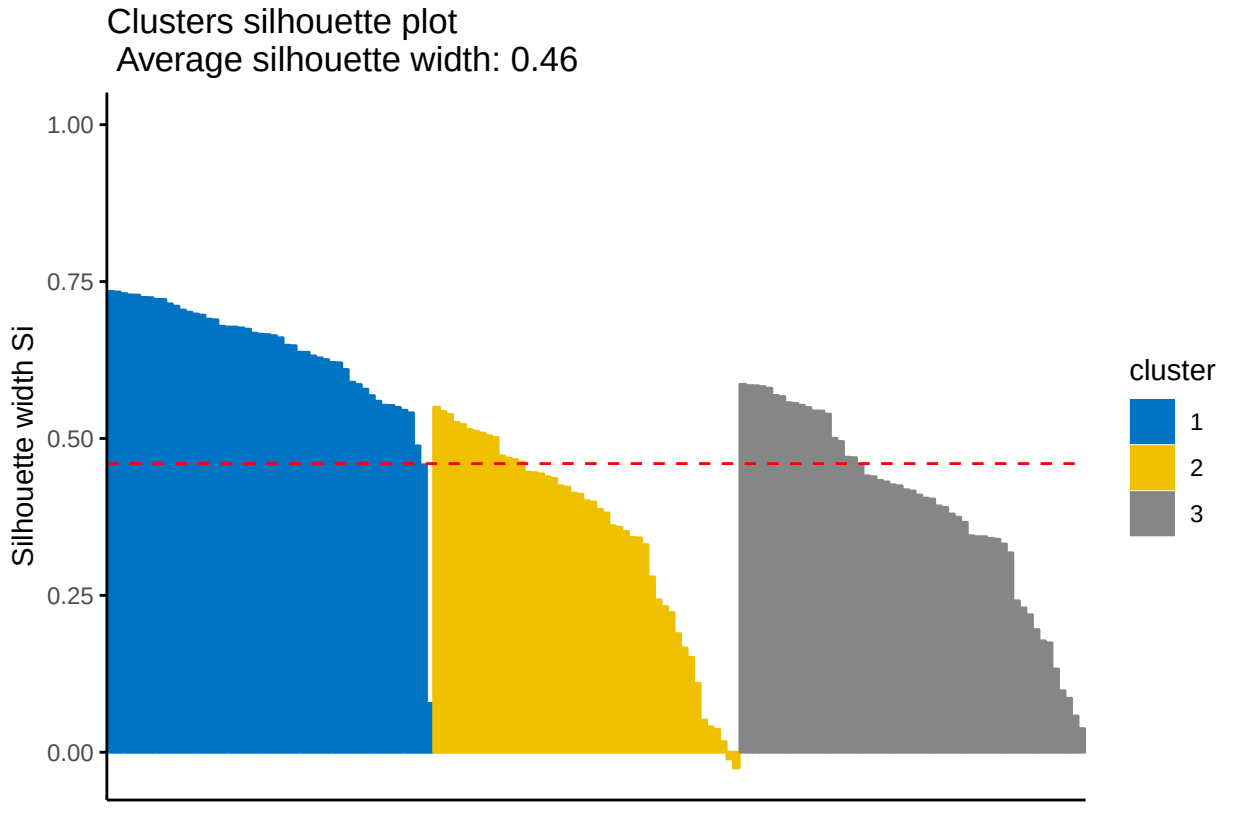
Si este coeficiente es cercano a 1, la observación está en un cluster correcto; si es cercano a 0, esta se encuentra entre dos clusters; y si es negativo, probablemente esté en un cluster equivocado.

Podemos representar estos coeficientes en un gráfico para cada una de las observaciones, y así saber la calidad del clustering. Vemos a continuación los resultados para un clustering de k-medias con 3 clusters.

```
# Usamos k-medias con 3 clusters
km.res <- eclust(df, "kmeans", k = 3, nstart = 25, graph = FALSE)

# Visualizamos los anchos de silueta para k-means
fviz_silhouette(km.res, palette = "jco", ggtheme = theme_classic())
```

```
##   cluster size ave.sil.width
## 1       1   50          0.64
## 2       2   47          0.35
## 3       3   53          0.39
```



Los resultados obtenidos indican que la mayoría de los datos están bien divididos a excepción de unos pocos que se encuentran en el segundo cluster y podrían situarse en otro (su coeficiente es próximo a 0). Este resultado no es de extrañar ya que el segundo y tercer cluster son próximos y sus elipses de confianza tienen

una intersección destacable. Como el valor medio es 0.46, positivo y alejado de 0, podemos decir (a falta de un criterio absoluto) que los clusters son válidos.

Para tener una mayor tranquilidad estudiaremos nuevamente la calidad de los clusters usando el índice de Dunn. Este lo obtenemos como el cociente entre la mínima distancia entre clusters y la máxima distancia entre objetos de un cluster (diámetro), de modo que un valor mayor indica un clustering de más calidad.

```
# Obtenemos el índice de Dunn
km_stats <- cluster.stats(dist(df), km.res$cluster)
km_stats$dunn
```

```
## [1] 0.02649665
```

Dado que el valor es bajo podríamos pensar que todo el trabajo realizado queda invalidado, pero como el índice de Dunn es muy sensible a outliers por usar solo dos distancias, podemos descartarlo dada la estructura de nuestros datos.

Validación externa

Como estos datos son de diferentes especies, un criterio para saber si los clusters son buenos es saber si separan por especie. Con este objetivo comparamos los clusters con las especies.

```
# Observaciones según clasificación por especie y por cluster
table(iris$Species, km.res$cluster)
```

```
##
##                1  2  3
##   setosa      50  0  0
##   versicolor   0 11 39
##   virginica    0 36 14
```

Podemos ver en la tabla que toda la setosa ha sido colocada en el cluster 1, una gran parte de la virginica en el cluster 2, y una gran parte de la versicolor en el cluster 3. Algunas de las virginicas han sido asignadas al cluster 3, del mismo modo que algunas versicolor se encuentran en el cluster 2. Para traducir estos resultados, calculamos el índice Rand corregido, y el Meila VI (que valen -1 en el peor caso y 1 en el mejor).

```
# Pasamos los nombres de las especies a números
species <- as.numeric(iris$Species)

# Obtenemos estadísticas usando las especies como clustering principal,
# y kmeans como clustering alternativo
clust_stats <- cluster.stats(d = dist(df),
                             species, km.res$cluster)

# Índice de Rand corregido
clust_stats$corrected.rand
```

```
## [1] 0.6201352
```

```
# VI
clust_stats$vi
```

```
## [1] 0.7477749
```

A falta de un criterio absoluto podemos decir que ambos índices son adecuados por situarse relativamente cerca del 1, de modo que el método de k-medias es apropiado.

Por dar una comparación podemos hacer el mismo estudio con clusters jerárquicos y PAM.

```r
# Concordancia entre especie y clusters PAM
pam.res <- eclust(df, "pam", k = 3, graph = FALSE)
table(iris$Species, pam.res$cluster)
```

```
##
##                1  2  3
##   setosa      50  0  0
##   versicolor   0  9 41
##   virginica    0 36 14
```

```r
# Coeficiente VI
cluster.stats(d = dist(df),
              species, pam.res$cluster)$vi
```

```
## [1] 0.7129034
```

```r
# Concordancia entre especie y clusters jerárquicos
res.hc <- eclust(df, "hclust", k = 3, graph = FALSE)
table(iris$Species, res.hc$cluster)
```

```
##
##                1  2  3
##   setosa      49  1  0
##   versicolor   0 27 23
##   virginica    0  2 48
```

```r
# Coeficiente VI
cluster.stats(d = dist(df),
              species, res.hc$cluster)$vi
```

```
## [1] 0.6944976
```

Obtenemos que el clustering jerárquico mete casi todas las setosa y virginica en sus propios clusters, pero las versicolor quedan separadas a medias entre los clusters 2 y 3. Así, el coeficiente VI es algo menor que para k-medias, pero aún así próximo a 0.7. En cuanto a PAM, este da resultados parecidos a las k-medias, aunque su coeficiente VI es un poco menor.

Elección del mejor algoritmo

Hemos visto que los distintos algoritmos de clustering pueden llevar a resultados bastante distintos, por lo que necesitamos una manera fácil de compararlos entre ellos y determinar qué método es más adecuado para un conjunto de datos.

Los criterios de comparación de algoritmos de clustering que vamos a seguir dependen de dos tipos de medidas:

- *Medidas internas*, que utilizan información del proceso de clustering para evaluar la calidad de este, donde se incluyen medidas como el ancho de silueta, el índice de Dunn y la conectividad.
- *Medidas de estabilidad*, que evalúan la consistencia del clustering comparándolo con los resultados que se obtienen al eliminar una variable.

Dentro de estas últimas, nos encontramos con varias medidas distintas:

- La proporción media de no-solapamiento (APN), que mide la proporción media de observaciones que no han sido colocadas en el mismo cluster cuando se estudia el dataset entero y cuando se elimina una columna de este.

- La distancia media (AD), que mide la distancia media entre observaciones que han sido colocadas en el mismo cluster en el dataset entero y quitándole una columna.
- La distancia media entre medias (ADM), que mide la distancia media entre centros de clusters para las observaciones que han sido colocadas en el mismo cluster en ambos casos.
- La cifra de mérito (FOM), que mide la media de la varianza dentro del cluster de la columna eliminada cuando el clustering se realiza respecto a las demás variables.

Las medidas APN, ADM y FOM toman valores entre 0 y 1, donde una proximidad mayor a 0 indica una consistencia mayor del clustering. La AD toma valores no negativos y no acotados, y también es preferible que estos sean cercanos a 0. En cuanto a las medidas internas, recordamos que el ancho de silueta y el coeficiente de Dunn deben ser grandes, y la conectividad pequeña.

A continuación vamos a comparar los métodos de clustering jerárquico, por k-medias, y PAM sobre `iris` utilizando como criterios las medidas internas que hemos visto.

```
# Guardamos los tres métodos en un vector
clmethods <- c("hierarchical", "kmeans", "pam")

# Usamos clValid para comparar su comportamiento respecto a las medidas
# internas cuando se usan entre 2 y 6 clusters

intern <- clValid(df, nClust = 2:6,
                  clMethods = clmethods, validation = "internal")

# Mostramos los resultados de la comparación
summary(intern)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   0.9762  5.5964  7.5492 18.0508 24.7306
##              Dunn           0.2674  0.1874  0.2060  0.0700  0.0762
##              Silhouette     0.5818  0.4803  0.4067  0.3746  0.3248
## kmeans       Connectivity   0.9762 23.8151 25.9044 40.3060 40.1385
##              Dunn           0.2674  0.0265  0.0700  0.0808  0.0808
##              Silhouette     0.5818  0.4599  0.4189  0.3455  0.3441
## pam          Connectivity   0.9762 23.0726 31.8067 35.7964 44.5413
##              Dunn           0.2674  0.0571  0.0566  0.0642  0.0361
##              Silhouette     0.5818  0.4566  0.4091  0.3574  0.3400
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 0.9762 hierarchical 2
## Dunn         0.2674 hierarchical 2
## Silhouette   0.5818 hierarchical 2
```

Observamos que la función `clValid` nos devuelve que el método óptimo de los tres que le hemos introducido

es el jerárquico de 2 clusters. No obstante, vemos que los valores de conectividad, ancho de silueta y coeficiente de Dunn son iguales en los tres métodos cuando se usan 2 clusters, y de hecho el resultado óptimo que da la salida de la función cambia en función del orden en el que estos le sean introducidos. Por tanto, concluimos que, según las medidas internas, se nos recomienda usar cualquiera de los tres métodos con 2 clusters. Si, por el contrario, hacemos caso a los métodos de elección de número de clusters y queremos usar 3 de ellos, nos damos cuenta de que las medidas internas nombral al clustering jerárquico como el mejor algoritmo, dado que el ancho de silueta y coeficiente de Dunn son mayores que en los otros dos, así como la conectividad es menor.

Repitiendo la comparación, esta vez con las medidas de estabilidad, obtenemos otros resultados:

```
# Comportamiento respecto a las medidas de estabilidad
stab <- clValid(df, nClust = 2:6, clMethods = clmethods,
                validation = "stability")

# Mostramos los resultados de la comparación
summary(stab)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                         2      3      4      5      6
##
## hierarchical APN  0.0033 0.0370 0.0859 0.1088 0.1342
##              AD   1.4924 1.4601 1.4015 1.3269 1.2251
##              ADM  0.0161 0.1451 0.1448 0.3919 0.3176
##              FOM  0.5957 0.5809 0.5571 0.5334 0.5101
## kmeans       APN  0.0128 0.1034 0.1290 0.2130 0.2950
##              AD   1.5060 1.2685 1.1568 1.1269 1.0949
##              ADM  0.0555 0.2152 0.2147 0.4499 0.5147
##              FOM  0.6049 0.5206 0.4888 0.4805 0.4910
## pam          APN  0.0128 0.1162 0.1420 0.1655 0.1886
##              AD   1.5060 1.2721 1.1665 1.0726 1.0043
##              ADM  0.0555 0.2266 0.2500 0.2834 0.2814
##              FOM  0.6049 0.5032 0.4828 0.4789 0.4558
##
## Optimal Scores:
##
##     Score  Method       Clusters
## APN 0.0033 hierarchical 2
## AD  1.0043 pam          6
## ADM 0.0161 hierarchical 2
## FOM 0.4558 pam          6
```

Donde esta vez, desde el punto de vista de la estabilidad, nos conviene hacer clustering jerárquico de 2 clusters si queremos minimizar la APN y la ADM, o PAM de 6 clusters si queremos minimizar la AD y la FOM. Si queremos usar 3 clusters, para minimizar la APN nos conviene usar clustering jerárquico; para minimizar la AD, k-medias; para minimizar la ADM, jerárquico; y para minimizar la FOM, PAM.

Como podemos ver, no existe un método de clustering perfecto, sino que debemos tomar decisiones en función de qué medidas nos es más importante reducir.

Conclusión

En este documento hemos podido ver un seguido de técnicas de validación de clusters que nos han permitido averiguar el ajuste de estos sobre el conjunto de datos `iris`.

Hemos comenzado estudiando la tendencia del clustering, donde hemos visto la necesidad de determinar la no uniformidad de los datos para que estos puedan ser agrupados de manera significativa. Para ello, hemos definido un método estadístico que permite diferenciar una muestra de datos de una uniforme, así como hemos visto métodos visuales con este mismo propósito. Sobre `iris` hemos comprobado claramente que los datos no son uniformes.

A continuación, hemos estudiado cómo saber el número óptimo de clusters a utilizar sobre un conjunto de datos, donde nuevamente hemos explicado dos métodos visuales (métodos del codo y la silueta) y un método estadístico, además de una función de R que compara los resultados de 30 métodos distintos. En `iris` ha habido discrepancias entre los métodos, ya que algunos han distinguido 2 clusters y otros 3, aunque la decisión mayoritaria ha sido esta última cifra.

Después de escoger el número de clusters, hemos proseguido con el análisis de la corrección del proceso de clustering, donde hemos distinguido formas de validación interna y externa. En este primer grupo, hemos definido el ancho de silueta y el coeficiente de Dunn como criterios para determinar la bondad del clustering a partir de su compacidad y separación, además de su conectividad. En cuanto a la validación externa, hemos comparado los resultados con la división por especies, donde hemos usado el índice corregido de Rand y el VI de Meila para determinar su similitud. Tanto la interna como la externa nos han hecho ver que el clustering de 3-medias realizado sobre `iris` está bien ajustado.

Finalmente, hemos visto cómo comparar distintos métodos de clustering para ver cuál es el más adecuado para el conjunto de datos. Hemos distinguido entre criterios basados en medidas internas y en medidas de estabilidad, donde hemos observado que algunas medidas se reducen más con un método u otro. Sobre `iris`, los algoritmos que han funcionado mejor en la reducción de estas han sido el jerárquico de 2 clusters y el PAM de 6 clusters, aunque restringiendo el estudio a 3 clusters, todos los algoritmos probados han resultado ser útiles en la minimización de al menos una de las medidas.