Técnicas de *clustering*

PID_00264730

Xavi Font

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Xavi Font

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Ayza Graells (2019)

Primera edición: marzo 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Oberta UOC Publishing, SL

Índice

Introducción

Las técnicas y los procedimientos descritos en este módulo son en ocasiones los únicos que podemos aplicar a nuestros datos. Si la naturaleza del conjunto de datos no viene caracterizada por una variable determinante u objetivo, como sucedía con los métodos de clasificación, la aproximación más razonable es intentar agrupar nuestros datos. En este contexto no tenemos ninguna etiqueta que determine el tipo de clase a la que pertenece la observación e interesaría disponer de esta información.

El concepto de agrupar observaciones o de aplicar un método clúster es una técnica muy utilizada porque permite agrupar observaciones similares dentro de un determinado grupo y las que son distintas en otro grupo. Hay dos formas de aplicar este concepto en cuanto a que si cada observación debe ir a un grupo (de forma excluyente) o puede formar parte de más de un grupo. Esta última aproximación, donde una observación puede estar en un 70 % en el grupo 4 en un 25 % en el grupo 1 y en un 5 % en el grupo 2, no será descrita en este módulo (*fuzzy clustering*).

Podemos imaginar el procedimiento de *clustering* como el de la generación de una nueva variable. Se puede interpretar como una técnica de segmentación que divide nuestro conjunto de observaciones en diferentes grupos o segmentos. Esta nueva variable que corresponde al grupo puede ser utilizada para enriquecer el conjunto de los datos analizados o para combinarla con los métodos de clasificación que consideremos adecuados para nuestro propósito.

Los métodos de *clustering* tienen especial relevancia en diferentes ámbitos profesionales. Podemos citar, entre otros, los siguientes:

- **Marketing**: las técnicas de *clustering* pueden ayudar a localizar perfiles específicos de clientes. Por ejemplo, un grupo de clientes con alto poder adquisitivo o con unas necesidades específicas. Esta información puede ayudar a desarrollar productos o a realizar campañas mejor adaptadas para este grupo de clientes.

- **Salud**: el uso de métodos de segmentación permite descubrir agrupaciones de interés en las cadenas de ADN, o en la secuencia de aminoácidos en proteínas. También para agrupar pacientes a partir de rasgos comunes y que ayuden a mejorar tratamientos o terapias.

- **Industria**: las características de nuestras piezas pueden utilizarse para segmentar todas nuestras observaciones e identificar grupos de interés.

- **Agricultura y biología**: desde una óptica de las ciencias biológicas, al procedimiento de agrupar se le denomina taxonomía. En cualquier caso, la aplicación de técnicas de segmentación en la agricultura para identificar segmentos de potencial interés es también habitual.

- **Político y social**: las técnicas de *clustering* pueden ayudar a explicar cómo se agrupan los votantes (las observaciones) y que características los definen.

- **Documentación**: cómo se pueden agrupar todos los documentos que tenemos en nuestra empresa u organización. Las estrategias de *clustering* pueden ayudar a colocar los documentos en grupos de temática parecida y que, por tanto, ayuden a su correcta agrupación y comprensión.

Uno de los aspectos complejos de las técnicas de *clustering* es que existe infinidad de propuestas. Es decir, encontraremos multitud de estrategias de *clustering* y para cada una de ellas variaciones que dan lugar a algoritmos diversos. Estos algoritmos pueden partir de alguna suposición, por ejemplo, se necesita de antemano definir el número de grupos que queremos generar o, en cambio, el algoritmo ya decide el número de grupos óptimo. Pueden sugerir su funcionamiento óptimo si los datos presentan algún elemento distintivo.

Los procedimientos de segmentación que analizaremos son los siguientes:

- **Métodos jerárquicos**: aglomerativos y divisivos se basan en el concepto de distancia. Puntos con distancias próximas se conectan al mismo grupo, mientras que puntos con distancias altas van a grupos diferentes.

- **Métodos de partición**: k-means. Basados en centroides caracterizado por las medias.

- **Métodos espectrales**: spectral Clustering se basa en la información contenida en los vectores propios de una matriz de afinidad (similitud de elemento a elemento) para detectar estructura en los datos.

- **Métodos basados en densidad**: DBSCAN y pdfCluster (no paramétrico) se basan en agrupar por zonas densas de puntos.

Para evaluar los diferentes segmentos, existen diferentes medidas. Entre ellas, el concepto de pureza o la información mutua normalizada (NMI).

Objectivos

Los objetivos que el módulo plantea son los siguientes:

1. Comprender el contexto de los problemas de aprendizaje no supervisado.
2. Saber identificar variantes disponibles para realizar segmentación de nuestras observaciones.
3. Entender y aplicar métodos jerárquicos. Interpretar correctamente las representaciones gráficas que generan: dendogramas.
   - Saber definir el número de grupos.
   - Saber visualizar las observaciones de un grupo específico.
   - Saber utilizar diferentes variantes.
   - Determinar cómo agrupamos nuevos puntos.
4. Entender y aplicar métodos de partición basados en centroides k-means.
5. Entender y aplicar métodos espectrales basados en vectores propios (Laplacian).
6. Conocer algunos métodos adicionales, como los basados en densidad.

Debe considerarse la forma de evaluar los diferentes grupos (en general, alta similitud intragrupo y poca similitud extragrupo).

La explotación de esta segmentación puede ayudar a descubrir nuevo conocimiento de los datos bajo estudio.

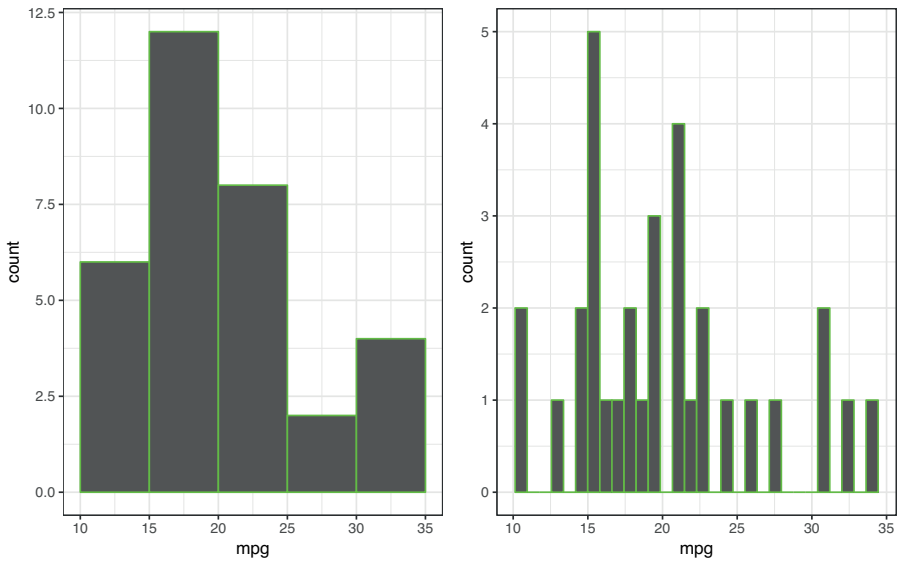
1. Problemas de aprendizaje no supervisado

1.1. Introducción

La utilización de métodos supervisados parte de que en el conjunto de los datos disponemos de una variable adicional, que en el caso de regresión es una medida cuantitativa y que en el caso de la clasificación es una etiqueta. El problema con muchos conjuntos de datos es que no disponemos de etiquetas. En el caso de reconocimiento de dígitos, tenemos la certeza de que las etiquetas corresponden correctamente a la representación del dígito. ¿Qué problema hay en general? Que el coste de poner etiquetas es elevado. Por tanto, hay ocasiones en las que no hay más remedio que acudir a algoritmos no supervisados.

Otro argumento en favor de los métodos no supervisados es obtener información adicional de la estructura de los datos. Podemos imaginar la utilización de un histograma como una buena aproximación a un método de *clustering*. Del conjunto de los datos que tenemos en una variable queremos conocer su distribución, cómo se comportan en realidad los datos de esta variable de una manera sintética.

Figura 1. Histogramas de la variable mpg con dos configuraciones



Podemos apreciar que un simple histograma puede ofrecernos visiones distintas de los mismos datos. La figura 1 utiliza en el primer caso menos grupos y en el segundo caso hay muchos grupos generados (muchos de ellos sin observaciones). Al hacer un histograma es determinante el número de grupos

que se representan o la amplitud de cada grupo (*binwidth*). Si miramos el detalle, podemos apreciar la distribución de puntos y los rangos para cada grupo generado.

El primer histograma genera la siguiente agrupación de datos:

	y	count	x	xmin	xmax
1	6	6	12.5	10	15
2	12	12	17.5	15	20
3	8	8	22.5	20	25
4	2	2	27.5	25	30
5	4	4	32.5	30	35

El segundo histograma genera una agrupación mucho más diversa (visualizamos primeros y últimos grupos):

	y	count	x	xmin	xmax
1	2	2	10.53448	10.12931	10.93966
2	0	0	11.34483	10.93966	11.75000
3	0	0	12.15517	11.75000	12.56034
4	1	1	12.96552	12.56034	13.37069
5	0	0	13.77586	13.37069	14.18103
26	2	2	30.79310	30.38793	31.19828
27	0	0	31.60345	31.19828	32.00862
28	1	1	32.41379	32.00862	32.81897
29	0	0	33.22414	32.81897	33.62931
30	1	1	34.03448	33.62931	34.43966

Estas dos agrupaciones provienen del mismo conjunto de datos y generan obviamente interpretaciones diferentes. En este sentido, si en el caso de una variable hay cierta complejidad en cómo agrupar, en el caso multidimensional el problema se agudiza.

Observaciones y aclaraciones de terminología

En ocasiones se utiliza el término *segmentación* como sinónimo de *clustering*. Esto también ocurre con el término *particionar*. Siendo puristas, estos dos conceptos provienen de otras disciplinas y aproximaciones. En estas notas entendemos estos conceptos como sinónimo de *clustering*.

Como sucede con la simple representación de un histograma, la aplicación de métodos de *clustering* nos puede ayudar a identificar patrones no conocidos y que puedan ser potencialmente interesantes para nuestros intereses.

1.2. Objetivos generales de los métodos de *clustering*

La información básica que utiliza cualquier método de *clustering* son observaciones y su conjunto de variables o características. En general, el objetivo de estos métodos es encontrar agrupaciones con una gran similitud entre observaciones del mismo clúster y, por tanto, se pretende minimizar esta medida. Al mismo tiempo, se pretende que observaciones pertenecientes a grupos diferentes tengan poca similitud. En otras palabras, se busca maximizar la distancia entre los grupos.

Matemáticamente, se pretende minimizar las similitudes intraclase mientras se maximiza la interclase.

¿Qué problemas se identifican bajo este concepto?

En realidad, la definición de similitud. La similitud es un concepto que depende de algunos ingredientes subjetivos. Por ejemplo, si mostramos un conjunto de jugadores de fútbol, ¿deberían Mesi y Cristiano estar en el mismo grupo o en grupos distintos? Para algunos, deberían estar juntos porque hay una justificación objetiva de rendimiento en el campo. Pero para otros, no. En este caso porque también hay datos objetivos que justificarían esta decisión (asignados a grupos diferentes). Otro ejemplo habitual lo encontramos en la asignación de etiquetas a nuestros hijos e hijas en cuanto a si el parecido es por parte de madre o padre. En general, se aprecia que en función del criterio del familiar (excepto en casos muy claros) cada uno ve una agrupación y, por tanto, emite una predicción y asignación (entendiendo que su criterio es el óptimo).

1.2.1. Aproximación matemática

Un planteamiento más formal sí que ayuda a entender la complejidad del problema y a visualizar la diversidad de aproximaciones que podemos encontrar.

Sea x_i una observación en un espacio p dimensional (tenemos p columnas), para cada $x_i, i \in 1,\dots,N$ debemos aprender a asignarle un clúster $y_i \in 1,\dots,K$. Además, cada clúster C estará caracterizado por un conjunto de parámetros θ. Se debe definir una función objetivo que describa la calidad de las agrupaciones. Esta función tiene, por un lado, el proceso de asignación de clústeres y, por otro, los parámetros:

$$f(\theta, Y)$$

La formulación de un problema de *clustering* implica optimizar la función objetivo descrita previamente:

$$\underset{\theta, Y}{\operatorname{argmin}} \quad f(\theta, Y)$$

La lista de variaciones que podemos encontrar depende en gran medida de:

- La función que utilizar: f.
- Cómo medimos la similitud entre observaciones $d(x_i, x_j)$.
- Qué estructura subyacente asumimos para asignar grupos
- Cómo se decide sobre el número de clústeres que tenemos. ¿Se define de forma previa o lo identifica el algoritmo de manera automática según algún criterio?

Esta metodología pretende ser el hilo conductor del objetivo que persiguen los métodos de *clustering*. Cuanto mayor sea la homogeneidad dentro del grupo y cuanto mayor sea la diferencia entre grupos, mejor y más reconocibles serán los clústeres generados.

Los algoritmos de *clustering* o segmentación parten de la premisa de que todas las variables tienen el mismo peso. Esto quiere decir que una variable con un rango de valores que va desde los -50.000.000 a 1.000.000.000 (imaginemos euros) ha de tener la misma importancia que una variable que indica los minutos semanales de navegación, que pueden oscilar entre los 0 minutos y los 500 minutos. Incluso una variación más pequeña sería posible en casos de medidas de peso de una flor de 0,05 a 50 g. Un requisito previo será garantizar que todas y cada una de las variables han sido estandarizadas: con media 0 y varianza 1. De esta manera, las variables quedan sin el efecto de la unidad o la escala.

Es decir, de manera natural aplicaremos por defecto:

$$\frac{x_i - center(x)}{scale(x)}$$

1.3. Medidas de similitud y distancias

Entre las medidas de similitud que podemos utilizar o encontrar en las diferentes técnicas citamos las más comunes:

1.3.1. Distancia euclidiana

La medida más utilizada es la distancia euclídea entre dos puntos (en un espacio de n dimensiones):

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

1.3.2. Distancia de Manhattan

Como su nombre indica, es una medida para calcular distancias en un entorno de calles tipo Manhattan.

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

1.3.3. Distancia de Pearson

Es una medida de correlación. Correlación de Pearson.

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

1.3.4. Distancia de Spearman

Es otra medida de correlación entre los rangos de las observaciones:

$$d_{spear}(x,y) = 1 - \frac{\sum_{i=1}^{n}(x'_i - \bar{x'})(y'_i - \bar{y'})}{\sqrt{\sum_{i=1}^{n}(x'_i - \bar{x'})^2 \sum_{i=1}^{n}(y'_i - \bar{y'})^2}}$$

donde $x'_i = rank(x_i)$ y $y'_i = rank(y_i)$

El cálculo de la matriz de distancias se puede realizar a través del método *dist* (de la librería stats cargada por defecto) y que permite calcular las siguientes distancias: euclidean, maximum, manhattan, canberra, binary, o minkowski, o bien las que figuran en la librería factoextra, que extiende la lista con medidas de correlación:

- euclidean
- maximum
- manhattan
- canberra
- binary
- minkowski
- pearson
- spearman
- kendall

Estas matrices de distancias se pueden visualizar gráficamente con el método *fviz_dist*.

1.4. Métodos jerárquicos

Los métodos de *clustering* jerárquicos son junto con los métodos de centroides, como el k-means, de los más antiguos. Gozan todavía de un uso extendido por su simplicidad y facilidad en la interpretación.

Existen dos aproximaciones a las agrupaciones jerárquicas:

- **Aglomerativas**: en este caso cada punto empieza en un clúster. En cada iteración se fusionan los clústeres más próximos.

- **Divisivas**: se empieza por un único clúster. En cada iteración el algoritmo debe decidir qué clúster dividir. Repetirá este proceso hasta que todas las observaciones estén en un clúster.

La mayoría de las veces el resultado de este método se visualiza con el uso de la representación gráfica denominada dendograma. El dendograma muestra una representación de tipo árbol donde se muestran los clústeres y subclústeres formados en cada iteración.

1.4.1. Métodos aglomerativos

La mayoría de los algoritmos aglomerativos se basan en alguna variación de esta aproximación simple:

1) Calcular la matriz de proximidad

2) **repeat**

3) Unir los dos clústeres más próximos

4) Actualizar la matriz de proximidad con el nuevo clúster

5) **until** que solo quede un clúster

La clave del comportamiento de estos métodos está en la función de proximidad utilizada.

Costes del método:

La complejidad algorítmica en notación *big O* es:

$$O(n^2 logn)$$

La lista de variantes que podemos utilizar son:

- ward.D
- ward.D2
- single
- complete
- average (= UPGMA)

- mcquitty (= WPGMA)
- median (= WPGMC)
- centroid (= UPGMC)

Cada uno de estos métodos determina cómo de similares son los clústeres. Por ejemplo, la opción "single" busca la distancia más corta entre dos puntos de dos clústeres diferentes. La opción "complete" busca la distancia más larga entre dos puntos en dos clústeres diferentes. Cada una de estas opciones generará un árbol de agrupación específico.

Debemos ser prudentes en la interpretación de los diagramas obtenidos. En ocasiones, los resultados ofrecidos pueden no tener ningún sentido y es función del profesional o investigador determinar el grado de validez de la solución facilitada.

Otro aspecto que también cabe tener en consideración es el último punto añadido en nuestro dendograma. Corresponde a la observación más extraña, o desde una óptica de *clustering* la observación más diferente del resto. Esta información puede ser valorada para identificar valores atípicos o en la identificación de anomalías o fraudes.

Para realizar un ejemplo, utilizamos los datos ya conocidos de mtcars:

	mpg	cyl	disp	hp	carb
Mazda RX4	21.0	6	160	110	4
Mazda RX4 Wag	21.0	6	160	110	4
Datsun 710	22.8	4	108	93	1
Ferrari Dino	19.7	6	145	175	6
Maserati Bora	15.0	8	301	335	8
Volvo 142E	21.4	4	121	109	2

Es imprescindible normalizar los datos de manera que todas las columnas tengan la misma media y varianza:

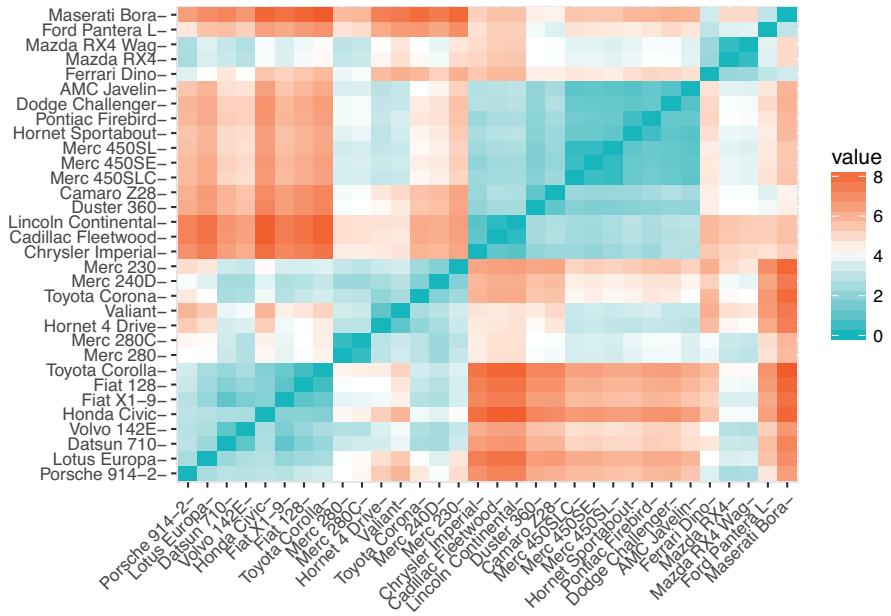
```
scale(mtcars)
```

Todas las variables del conjunto de datos tienen la misma media y la misma desviación.

	mpg	cyl	disp	hp	carb
Mazda RX4	0.1509	-0.1050	-0.5706	-0.5351	0.7352
Mazda RX4 Wag	0.1509	-0.1050	-0.5706	-0.5351	0.7352
Datsun 710	0.4495	-1.2249	-0.9902	-0.7830	-1.1222
Ferrari Dino	-0.0648	-0.1050	-0.6916	0.4129	1.9734
Maserati Bora	-0.8446	1.0149	0.5670	2.7466	3.2117
Volvo 142E	0.2173	-1.2249	-0.8853	-0.5497	-0.5030

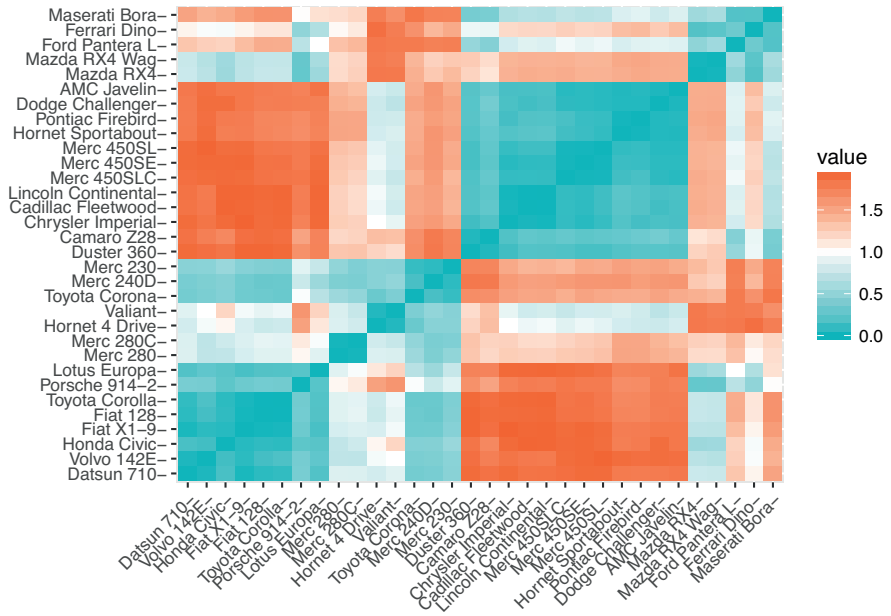
A partir de esta transformación de las columnas (variables) de nuestro conjunto de datos se puede aplicar el procedimiento de *clustering* aglomerativo. Una interesante etapa previa a la generación del clúster aglomerativo es visualizar la matriz de distancias con el método seleccionado. Por ejemplo, para distancias euclídeas:

Figura 2. Matriz de distancias euclídeas entre las observaciones



O el cálculo de las distancias con la correlación de Pearson:

Figura 3. Matriz de distancias con la correlación de Pearson entre las observaciones



A continuación vemos dos dendogramos obtenidos por el método de ward.D2 y posteriormente el de average.

Figura 4. Dendograma generado por hclust con distancia euclídea y método ward.D2

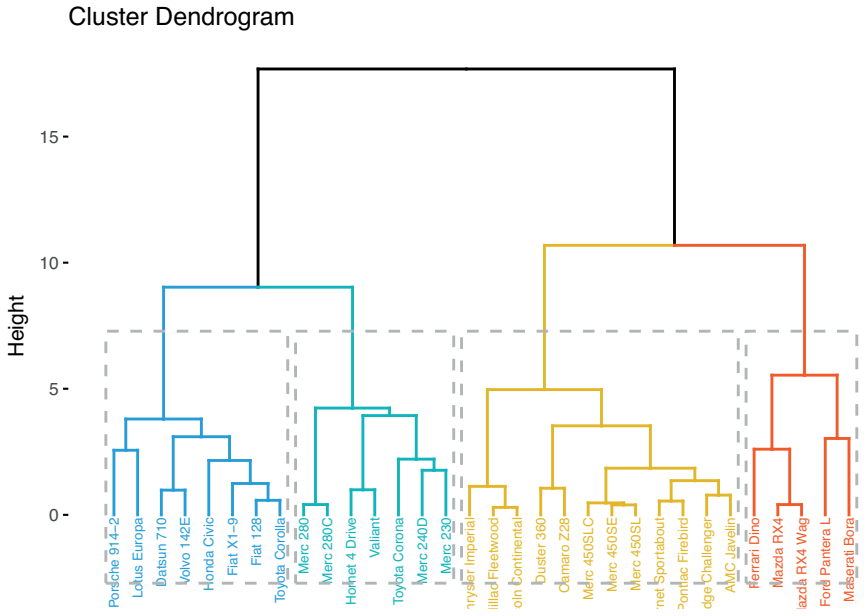
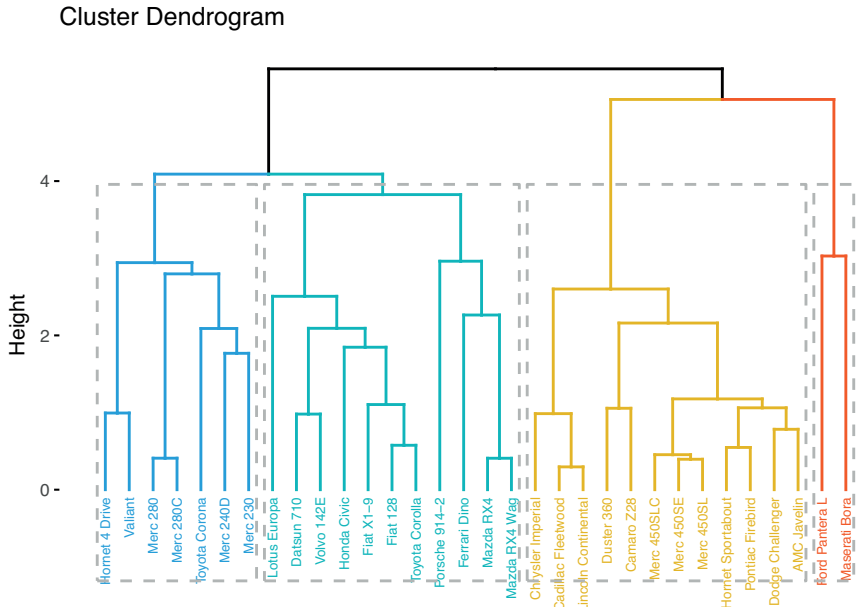


Figura 5. Dendograma generado por hclust con distancia euclídea y método average



Este conjunto de salidas se obtiene fácilmente con las siguientes instrucciones:

```
library("cluster")
library("factoextra")
data(mtcars)
df <- mtcars
```

```
# Cálculo matriz distancias y visualización
mxdist <- get_dist(df, stand = TRUE, method = "euclidean")
fviz_dist(mxdist)
# Calculo del cluster
res <- df %>% scale() %>% dist(method = "euclidean") %>% hclust(method = "ward.D2")
# Visualización con 4 grupos
fviz_dend(res, k = 4,rect = TRUE)
```

1.4.2. ¿Cómo identificar el número de clústeres?

Para la identificación del número de clústeres, las aproximaciones jerárquicas dan los clústeres de forma directa cortando el árbol (dendograma) por el punto especificado.

Si tenemos como resultado el clúster mostrado en la figura 4, podemos visualizar el contenido de cada uno de los grupos de forma directa.

Los objetos que devuelve el método hclust:

```
str(res)
List of 7
 $ merge      : int [1:31, 1:2] -15 -12 -1 -10 -14 -5 -18 -22 -3 -4 ...
 $ height     : num [1:31] 0.296 0.394 0.408 0.408 0.474 ...
 $ order      : int [1:32] 27 28 3 32 19 26 18 20 10 11 ...
 $ labels     : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ method     : chr "ward.D2"
 $ call       : language hclust(d = ., method = "ward.D2")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

Donde podemos ver cómo se distribuye en el caso de 4 grupos:

Si utilizamos la función cutree de la librería dendextend, podemos ver cómo quedan los grupos utilizando las funciones de filtrado conocidas:

```
corte <- cutree(res,k=4)
dfn <- df %>% mutate(label=corte)
 dfn %>% filter(label==1)
```

Vehículos asignados al grupo 1:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	label
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	1
Ford Pantera L	15.8	8	351	264	4.22	3.170	14.50	0	1	5	4	1
Ferrari Dino	19.7	6	145	175	3.62	2.770	15.50	0	1	5	6	1
Maserati Bora	15.0	8	301	335	3.54	3.570	14.60	0	1	5	8	1

Vehículos asignados al grupo 2:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	label
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1	2
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1	2
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2	2
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2	2

y este proceso se puede repetir para cada grupo.

Podemos verificar si esta configuración es estable o no. Existe una función llamada *clusterboot* que permite verificar esta estabilidad a través del coeficiente de Jaccard (que compara los miembros de un par de conjuntos para ver cuáles están compartidos y cuáles no).

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Observad que dos conjuntos que comparten todos los elementos obtendrán una medida del 100 %, mientras que si no comparten nada, la medida será de 0 %.

```
rescb <- clusterboot(df %>% scale() %>%
 dist(method = "euclidean"),clustermethod = hclustCBI,method="ward",k=4)
newlabels <- rescb$result$partition
rescb$bootmean
[1] 0.7672381 0.7368929 0.9059711 0.6994563
```

Los datos que se muestran indican que para el clúster 3 tenemos mucha estabilidad (valor próximo a 1.0), para el 1 y 2, una estabilidad justa y para el 4, poca estabilidad. El principal problema es que no tenemos garantía de que el número de clústeres elegido sea el correcto.

Tampoco sabemos nada sobre qué método da mejor resultado. Una opción a este problema es utilizar la *agnes* y el valor que genera llamado coeficiente aglomerativo. Cuanto más alto, mejor, y podemos ver que el método de "ward" da mejor resultado.

```
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# función para calcular el coeficiente aglomerativo
ac <- function(x) {
  agnes(df, method = x)$ac
}

# esta función requiere la librería purr
library(purr)
map_dbl(m, ac)
  average    single  complete      ward
0.9056177 0.7924041 0.9368123 0.9765103
```

Seguimos con el problema de cómo decidir por el número óptimo de grupos. Existen varias estrategias y vamos a detallar algunas de ellas. Todas estas estrategias están disponibles en las mismas librerías comentadas anteriormente. Todas estas opciones figuran dentro del método: *fviz_nbclust* (de la librería factoextra).

Método del codo

Este método se basa en el cálculo de (wcss = within cluster sums of squares):

$$wcss = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

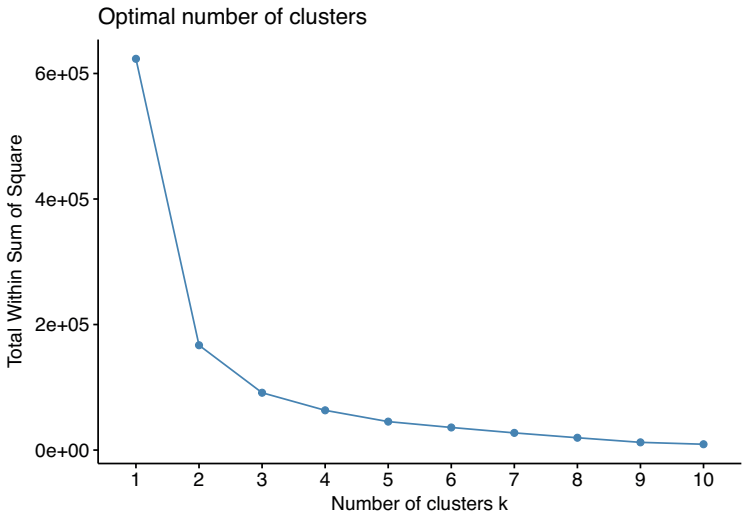
donde

$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

Si aplicamos este método en nuestro caso de los vehículos, veremos el siguiente gráfico:

```
fviz_nbclust(df, FUN = hcut, method = "wss")
```

Figura 6. Diagrama Elbow para obtener el número optimo de clústeres



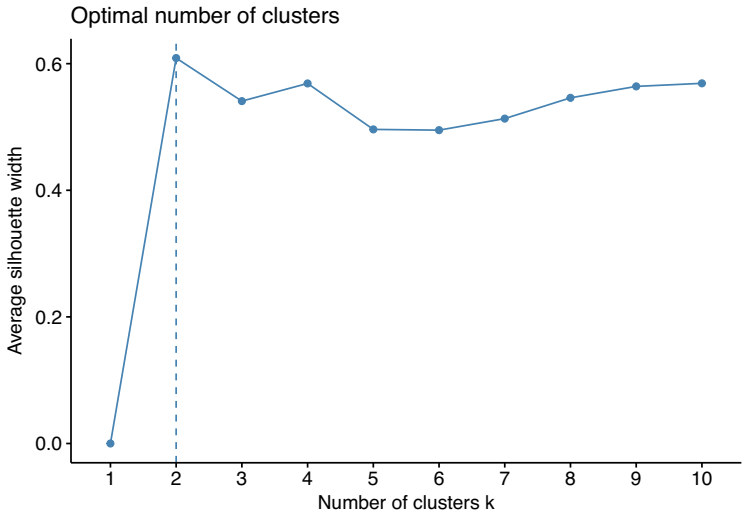
El punto óptimo estaría en 3 (es también razonable al tratarse de una aproximación gráfica elegir como número de clústeres óptimos el valor de 2).

Método de silueta

La idea subyacente a este método es calcular la calidad de los clústerrs con una medida promedio. La función *silhouette* de la librería cluster facilita este cálculo, que podemos ahorrar si utilizamos el mismo método utilizado anteriormente pero con la indicación del método igual a "silhouette".

```
fviz_nbclust(df, FUN = hcut, method = "silhouette")
```

Figura 7. Diagrama de silueta para obtener el número óptimo de clústeres



En este caso la respuesta que da el método es 2. Por tanto, el número de grupos que definir sería de 2.

Método del Gap Statistic

Este método es válido para la mayoría de los métodos de clúster y fue desarrollado por R. Tibshirani, G. Walther y T. Hastie, de la Universidad de Stanford.

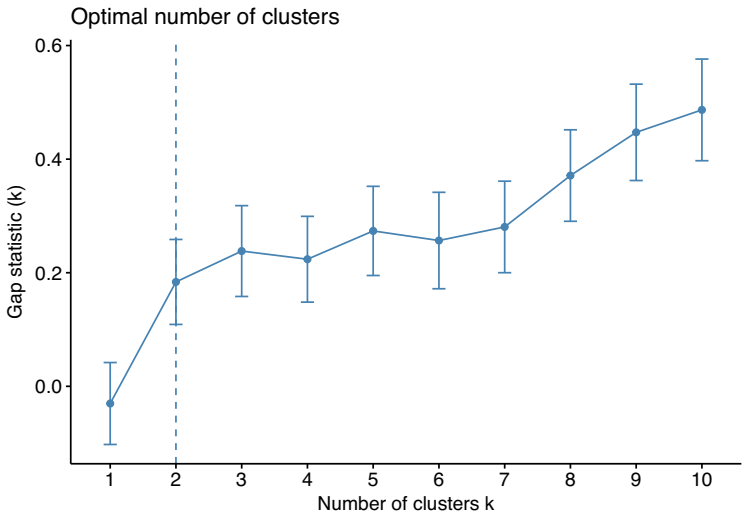
La idea que desarrollan es calcular la variación intraclúster para diferentes valores del número de clústeres. Este valor se compara con el valor esperado bajo la distribución de referencia de los datos. La fórmula utilizada es:

$$Gap_n(k) = E_n^* log(W_k) - log(W_k)$$

Procedemos como anteriormente utilizando el método *fviz_gap_stat*:

```
gap_stat <- clusGap(df, FUN = hcut, nstart = 25, K.max = 10, B = 50)
fviz_gap_stat(gap_stat)
```

Figura 8. Diagrama del salto estadístico para obtener el número óptimo de clústeres



1.4.3. Métodos divisivos

La construcción de clústeres con métodos divisivos sigue una aproximación inversa a la vista con los métodos aglomerativos. En este caso, el funcionamiento es de arriba abajo.

El método que usaremos para realizar un clúster jerárquico divisivo es *diana*.

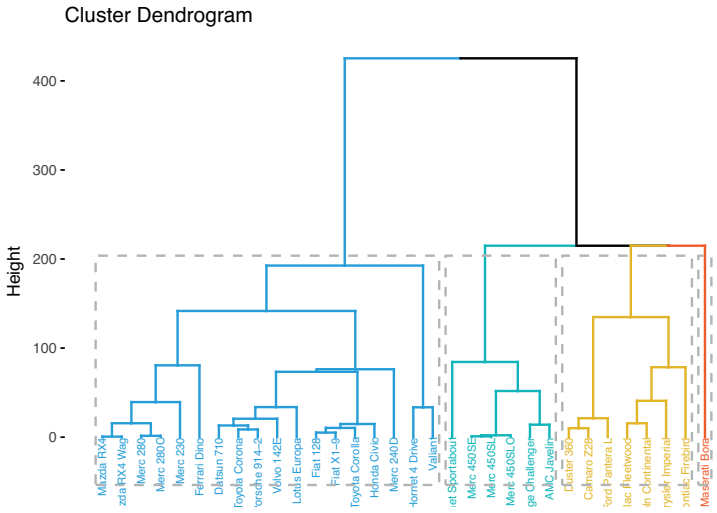
```
hc4 <- diana(df)
hc4$dc # Coeficiente relativo a la estructura clustering encontrada
# plot dendrogram
fviz_dend(hc4, k = 4, rect = TRUE)
```

El valor del coeficiente da:

```
## [1] 0.9330235
```

Y el resultado del dendograma es el siguiente:

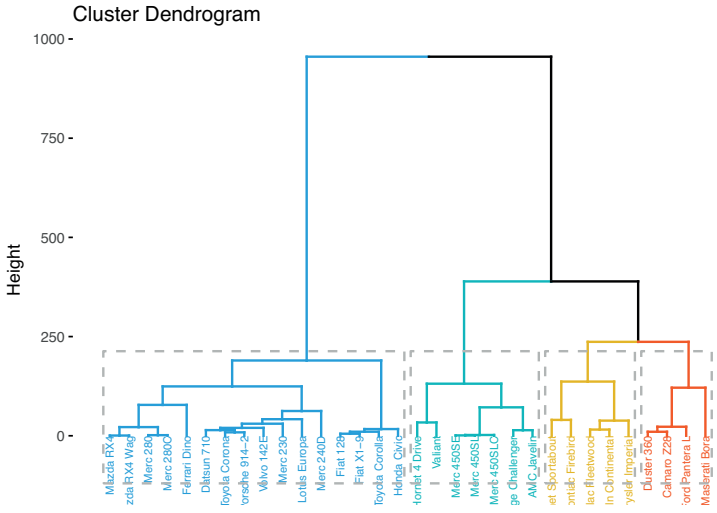
Figura 9. Diagrama de 4 clústeres por el método divisivo "diana"



Este resultado se puede comparar con el que genera la opción aglomerativa con un valor del coeficiente de estructura:

```
## [1]  0.9765103
```

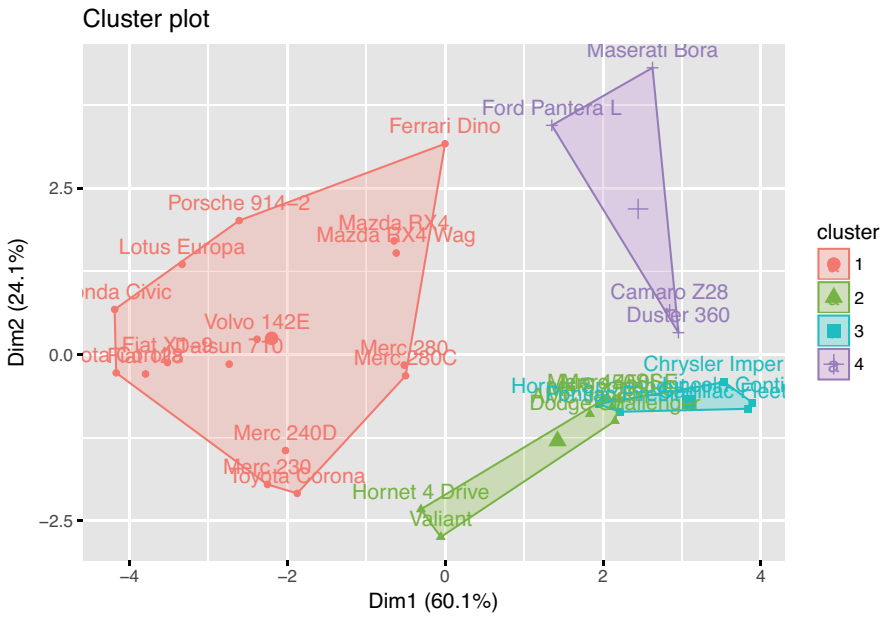
Figura 10. Diagrama de 4 clústeres por el método aglomerativo "ward"



En ocasiones puede ser interesante visualizar el conjunto de clústeres obtenido proyectados sobre los ejes de mayor variación de los datos (utilizando los vectores propios vía PCA).

```
hc <- hclust(d, method = "ward.D2" ) # Ward's method
subG <- cutree(hc, k = 4) # dividimos en 4 grupos
table(subG)
## 1 2 3 4
## 16 7 5 4

df %>% mutate(cluster = subG) %>% head

fviz_cluster(list(data = df, cluster = subG))
```

Figura 11. Diagrama de 4 clústeres por el método aglomerativo "ward" sobre los ejes PCA



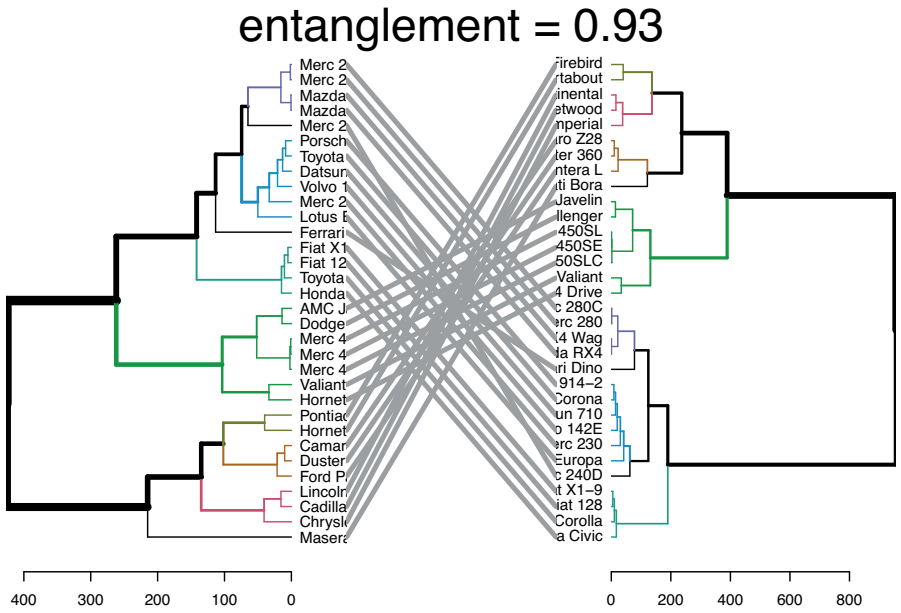
Cómo se ajustan dos dendogramas

Existe la posibilidad de poner dos propuestas de dendogramas enfrentadas para contrastar cómo se aproximan o separan de una forma visual. Se denomina *entanglement* y en ocasiones puede ser de ayuda:

```
res.dist <- dist(df, method = "euclidean")

hc1 <- hclust(res.dist, method = "complete")
hc2 <- hclust(res.dist, method = "ward.D2")

dend1 <- as.dendrogram (hc1)
dend2 <- as.dendrogram (hc2)

dend_list <- dendlist(dend1, dend2)

tanglegram(dend1, dend2,
```

```
  highlight_distinct_edges = FALSE, # Turn-off dashed lines
  common_subtrees_color_lines = FALSE, # Turn-off line colors
  common_subtrees_color_branches = TRUE, # Color common branches
  main = paste("entanglement =", round(entanglement(dend_list), 2))
  )
```

Figura 12. Tanglegram de dos métodos aglomerativos Complete y ward.D2



2. K-means

2.1. Introducción

El método de *clustering* k-means es uno de los métodos de agrupación basada en centroides más utilizados y antiguos. Es un método que funciona muy bien cuando los grupos que tienen que identificarse tienen una estructura compacta y convexa.

Figura 13. Datos generados (dat1) artificialmente con una clara identificación de grupos en los datos de manera compacta. Esta es la estructura ideal para aplicar el método de k-means

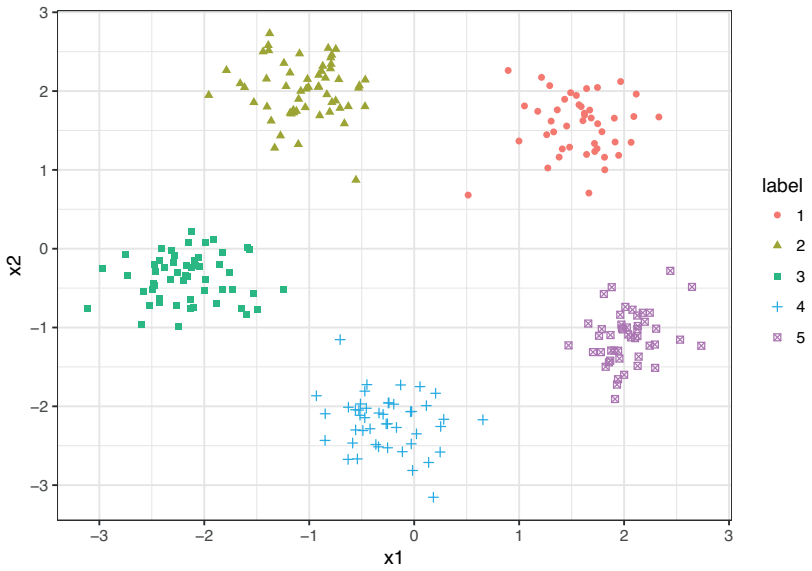
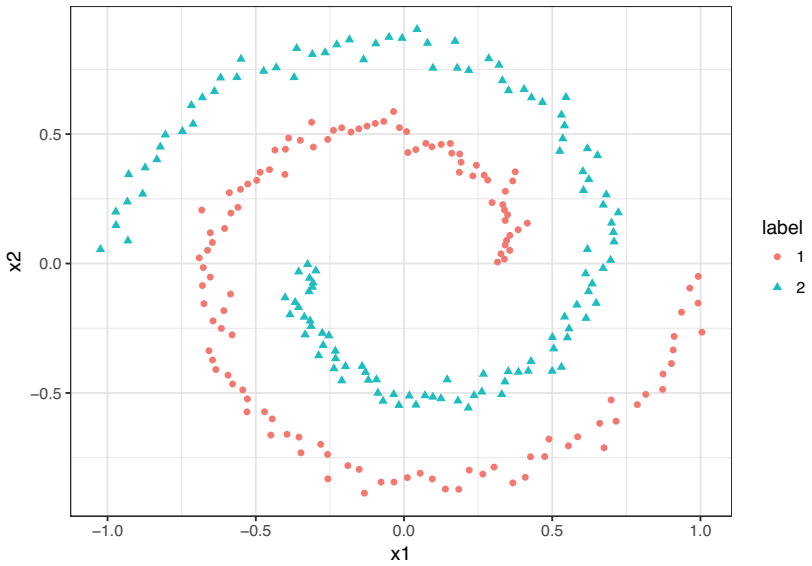


Figura 14. Datos generados (dat2) artificialmente con una clara identificación de grupos en los datos de forma conexa. Esta es la estructura ideal para que el método de k-means no funcione



Es evidente la mejor estructura de los datos para aplicar el método de k-means.

2.2. Procedimiento del k-means

La idea básica de funcionamiento de este algoritmo es muy trivial. Parte de un número por defecto de grupos. Este valor es suministrado, no se discute en este punto si el valor refleja un número correcto o no. A partir de este momento, se generan aleatoriamente tantos centroides como grupos hemos indicado. En este punto empieza un procedimiento iterativo que asigna cada observación al centroide más cercano. Una vez terminada esta operación, se recalculan los centroides y se repite el proceso de asignar las observaciones a los centroides. Como puede observarse, el proceso es muy simple y termina cuando no hay cambios en las asignaciones de las observaciones a los centroides.

Descriptivamente, el algoritmo sería:

1) definir el número de grupos o clústeres: *k*,

2) generar *k* centroides aleatoriamente,

3) asignar cada observación al centroide más próximo,

4) calcular la nueva ubicación de los centroides a partir de las observaciones asignadas y

5) repetir el proceso hasta que los centroides no varíen su posición.

Este procedimiento aplicado al caso generado artificialmente y que identificamos con el nombre *dat1* se visualiza en los siguientes diagramas:

Figura 15. Iteración 1, generando centroides aleatoriamente

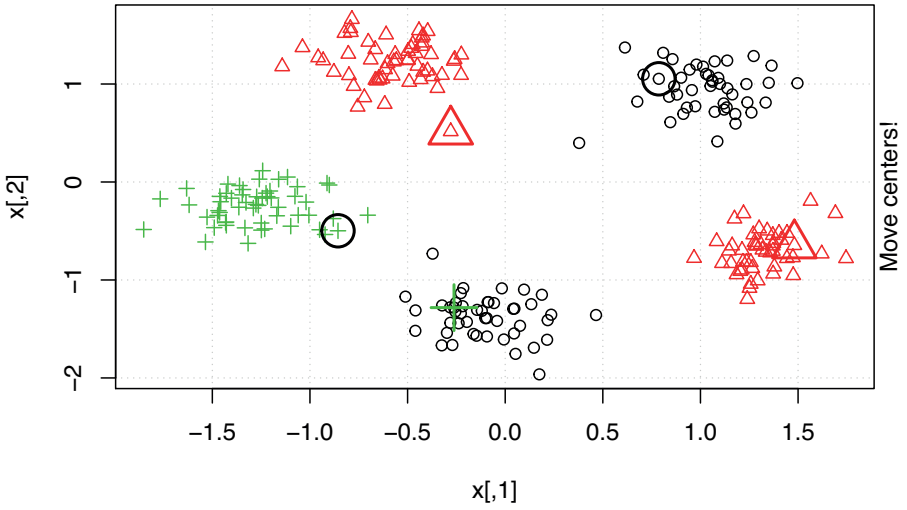


Figura 16. Iteración 1, asignando observaciones a los centroides más próximos

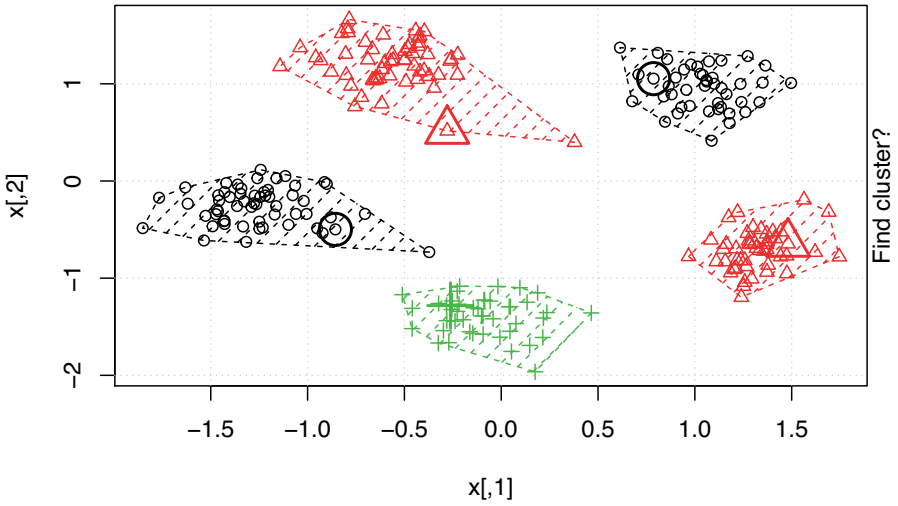


Figura 17. Iteración 2, asignando observaciones a los clústeres, los centroides se han desplazado de la situación inicial

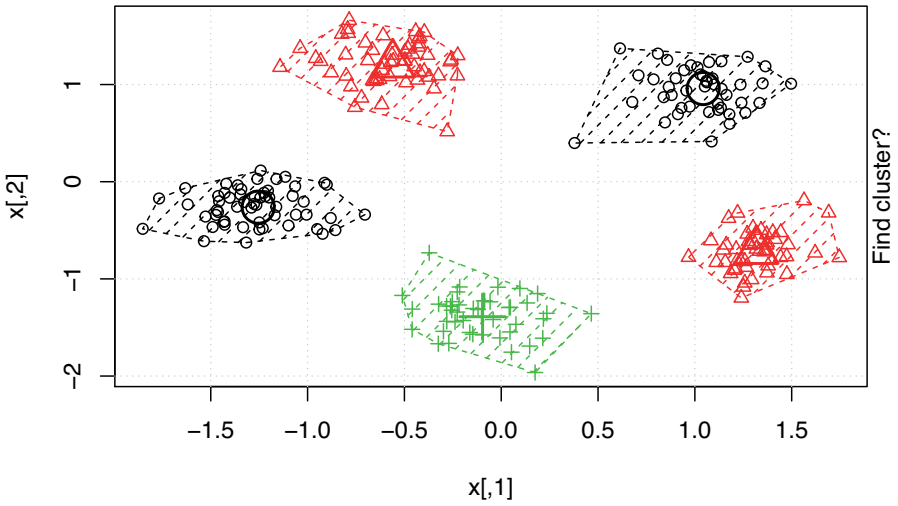
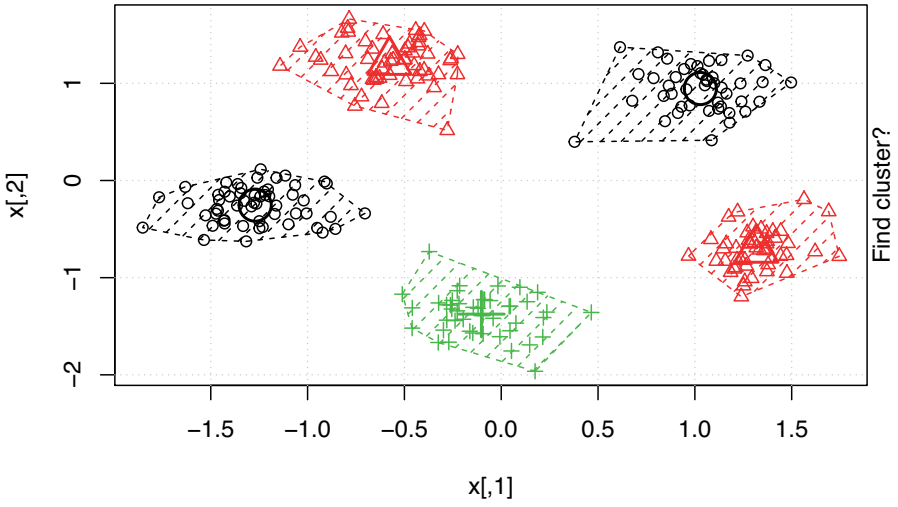


Figura 18. Iteración final, asignando observaciones a los clústeres y puntos claramente alrededor de los centroides



¿Qué sucede con el segundo caso? El procedimiento empieza de modo similar.

Figura 19. Iteración 1, generando centroides aleatoriamente

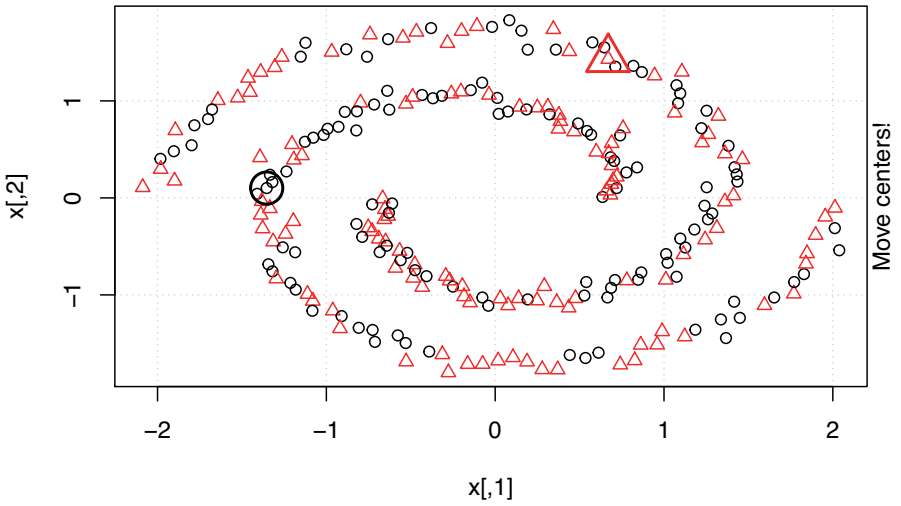


Figura 20. Iteración 1, asignando observaciones a los centroides más próximos

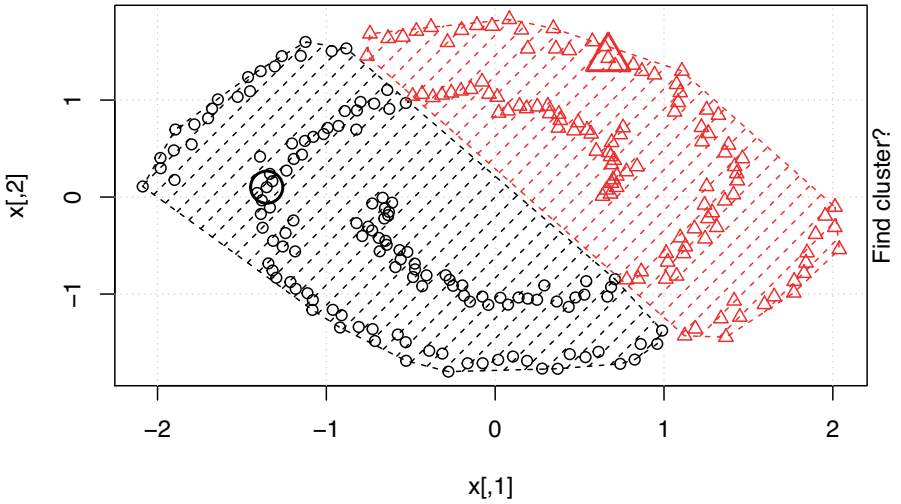


Figura 21. Iteración 12, asignando observaciones a los clústeres, los centroides se han desplazado de la situación inicial

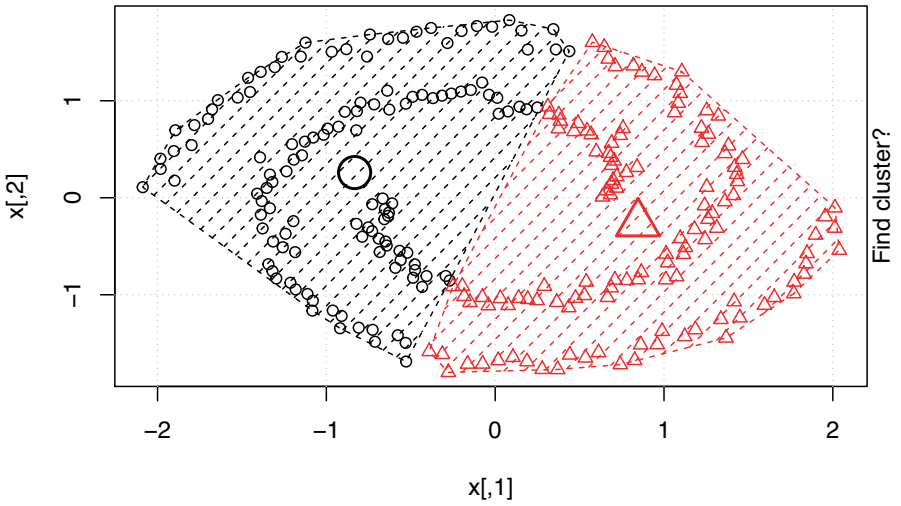
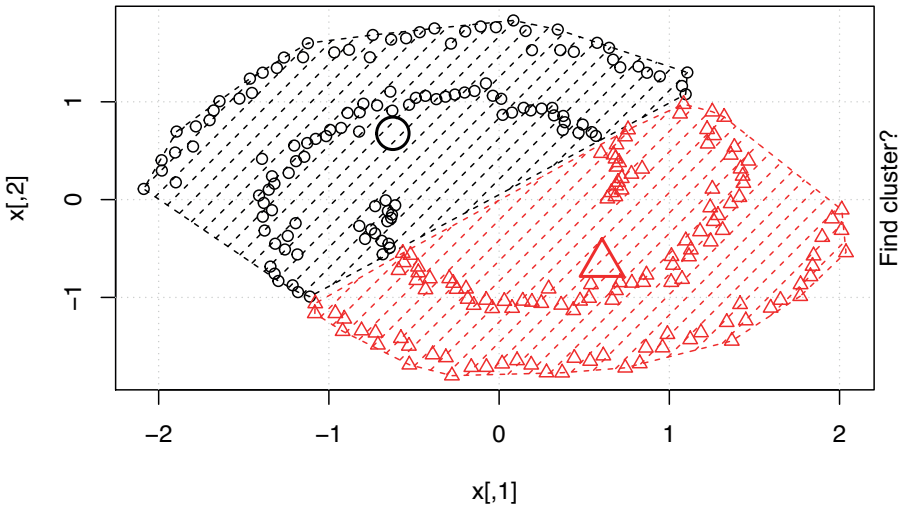


Figura 22. Iteración final, asignando observaciones a los clústeres y puntos claramente alrededor de los centroides



El resultado final no es satisfactorio para nada.

2.3. Elección del número óptimo de clústeres

De nuevo, la parte compleja en este método de clúster es la identificación del número óptimo de clústeres.

Vamos a proceder a esta identificación aplicando los mismos métodos descritos anteriormente y comparando su resultado en los casos presentados y que actúan de caso ideal y caso que evitar.

La salida del método define una estructura de datos como la mostrada a continuación:

```
rsdf1 <- dat1$x %>% scale() %>% kmeans(5)
str(rsdf1)

## List of 9
##  $ cluster     : int [1:250] 3 1 1 1 4 1 3 5 1 2 ...
##  $ centers     : num [1:5, 1:2] -0.102 -1.271 1.028 1.317 -0.577 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "1" "2" "3" "4" ...
##   .. ..$ : NULL
##  $ totss       : num 498
##  $ withinss    : num [1:5] 4.05 4.68 4.33 2.97 5.06
##  $ tot.withinss: num 21.1
##  $ betweenss   : num 477
##  $ size        : int [1:5] 46 56 45 47 56
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

En este caso el método que vamos a aplicar es *kmeans(5)*, pasando como argumento el número de clústeres que queremos (que de entrada no sabemos si es o no óptimo).

Podemos identificar las características de cada clúster visualizando las posiciones de cada centroide. Esto es:

	y	count	x	xmin	xmax
1	6	6	12.5	10	15
2	12	12	17.5	15	20
3	8	8	22.5	20	25
4	2	2	27.5	25	30
5	4	4	32.5	30	35

Podemos visualizar el número de observaciones por clúster:

```
rsdf1$size
## [1] 46 56 45 47 56
```

También hay información de las variaciones intraclústeres y extraclústeres:

```
rsdf1$withinss
## [1] 4.050092 4.684998 4.328276 2.968895 5.056929
```

```
rsdf1$betweenss
## [1] 476.9108
```

Y tenemos el elemento principal, que son las asignaciones realizadas a los grupos:

```
labels1 <- rsdf1$cluster
```

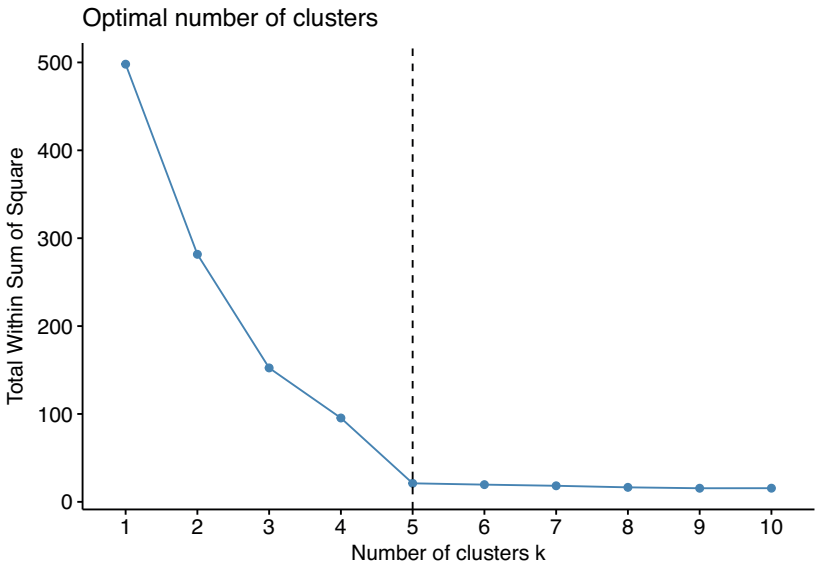
2.3.1. Método Elbow

En este caso la visualización nos da claramente un número óptimo en el valor de *k* igual a 5 (son los grupos que hemos creado artificialmente).

La generación se obtiene de este modo:

```
fviz_nbclust(dat1$x %>% scale(), kmeans, method = "wss")+
  geom_vline(xintercept = 5, linetype = 2)
```

Figura 23. Visualización de la variación de la suma total de cuadrados por diferentes valores de k para el conjunto de datos dat1. El valor óptimo queda muy marcado en la posición 5 (corresponde al codo)

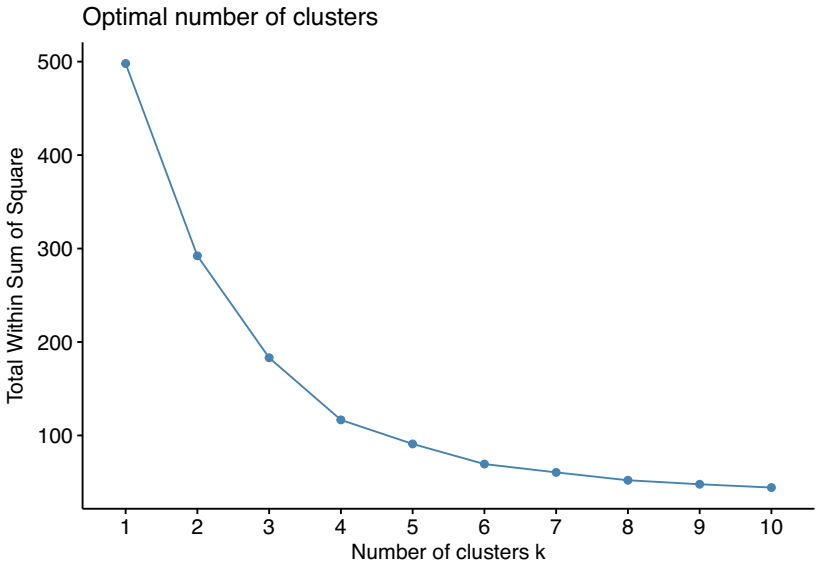


¿Qué sucede para el caso no ideal? Si visualizamos, veremos que no está muy clara la identificación del número óptimo. La sentencia utilizada es la misma cambiando el juego de datos.

```
fviz_nbclust(dat2$x %>% scale(), kmeans, method = "wss")
```

La observación del gráfico no permite extraer un valor concluyente al aplicar la regla del codo. ¿Dónde marcamos el codo?, ¿en 2, 3, 4, 5 o 6?

Figura 24. Visualización de la variación de la suma total de cuadrados por diferentes valores de k para el conjunto de datos dat1. El valor óptimo no queda muy claro



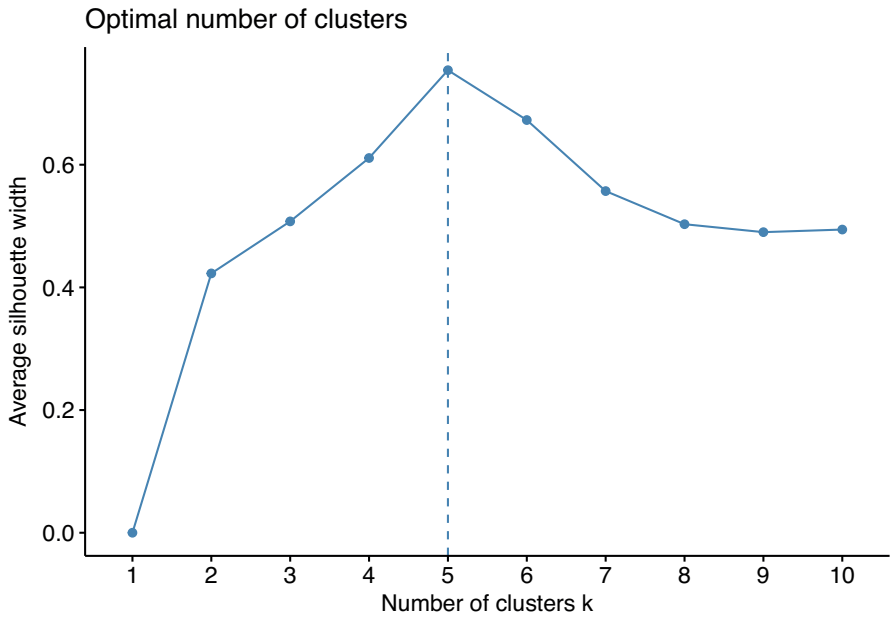
Una forma de resolver este *handicap* es por medio de otro método para contrastar.

2.3.2. Método de la silueta

La generación se obtiene haciendo un cambio en el método utilizado. Ya no utilizamos la suma de cuadrados (wss) sino el descrito como "silhouette":

```
fviz_nbclust(dat1$x %>% scale(), FUN = kmeans, method = "silhouette")
```

Figura 25. Visualización del valor óptimo por el método de la silueta para el caso 1



De nuevo la generación de la representación gráfica es concluyente en cuanto al valor óptimo de clústeres para nuestro conjunto de datos compacto. Nuevamente el valor es 5.

Para el caso más complejo tenemos la siguiente salida:

```
fviz_nbclust(dat2$x %>% scale(), FUN = kmeans, method = "silhouette")
```

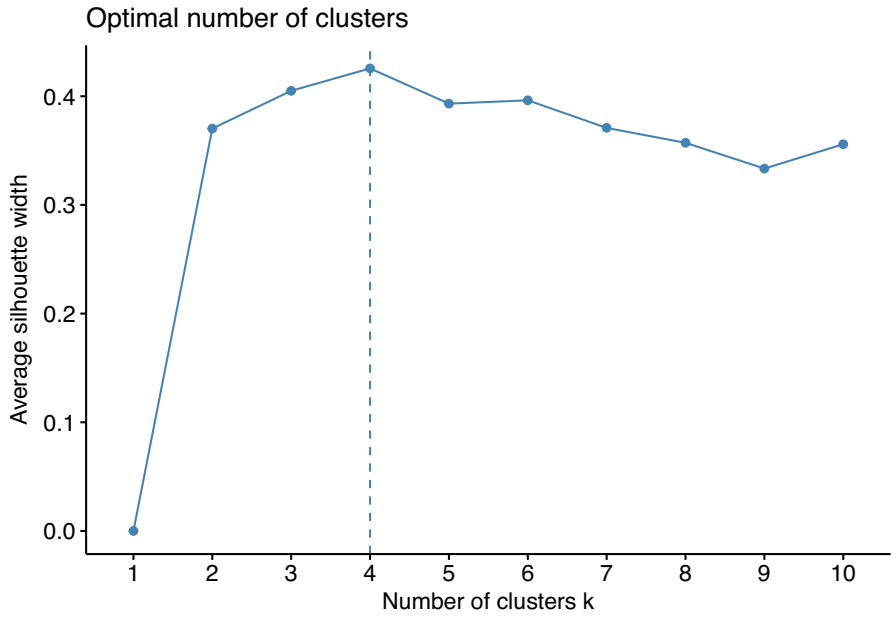
El valor no es el que esperábamos, pero al menos tenemos un valor de referencia.

2.3.3. Método del Gap Statistics

La generación se obtiene de este modo:

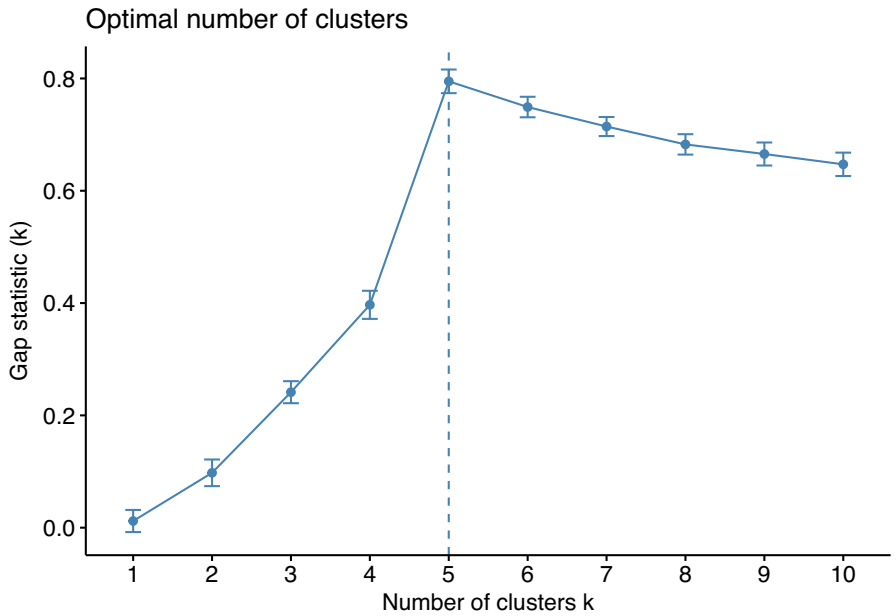
```
gap_stat <- clusGap(dat1$x %>% scale(), FUN = kmeans, nstart = 25, K.max = 10, B = 50)
```

Figura 26. Visualización del valor óptimo por el método de la silueta para el caso 2




```
fviz_gap_stat(gap_stat)
```

Figura 27. Visualización del valor óptimo por el método de Gap Statistic para el caso 1

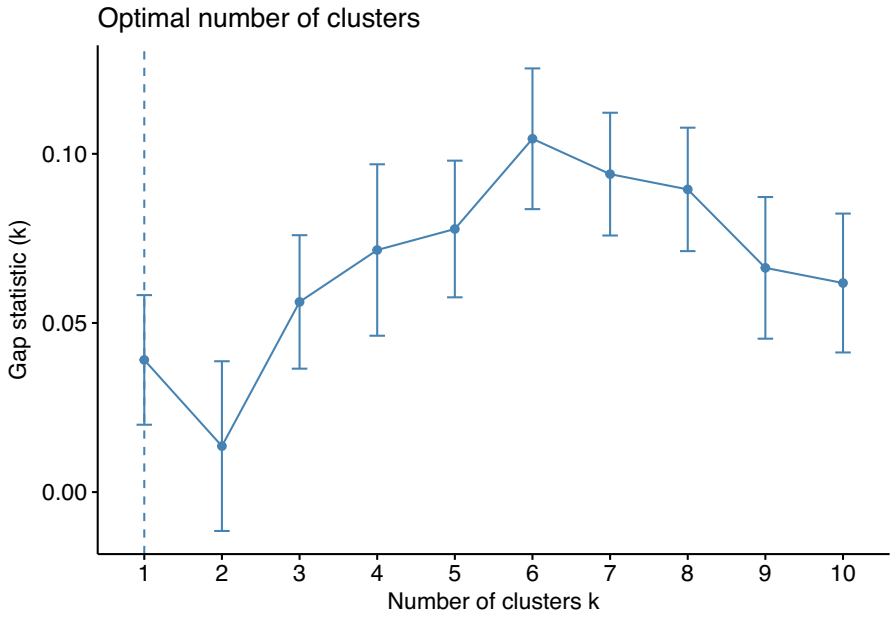


Podemos observar que en el juego de datos compacto los tres métodos dan el mismo valor óptimo, que es el de 5.

Para el caso del conjunto conexo no hay una posición compartida en cuanto al número óptimo.

```
gap_stat <- clusGap(dat2$x %>% scale(), FUN = kmeans, nstart = 25, K.max = 10, B = 50)
fviz_gap_stat(gap_stat)
```

Figura 28. Visualización del valor óptimo por el método Gap Statistic para el caso 2



2.4. Resultados finales

En los dos casos analizados podemos visualizar y proyectar las agrupaciones clúster obtenidas. Podemos ver la disparidad de elecciones en cuanto al numero de clústeres para el caso del conjunto de datos generado artificialmente que hemos llamado dat2 y los del conjunto de datos compacto llamado dat1.

En la librería clúster está la opción de analizar la estabilidad de la solución:

```
cbds1 <- clusterboot(dat1$x %>% scale(),clustermethod = kmeansCBI,
runs=100,iter.max=100,krange=5)
```

Que nos da como resultado:

```
# estability
cbds1$bootmean

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

El resultado muestra lo que *a priori* parecía evidente: gran estabilidad en las agrupaciones conseguidas. Esta solución o bien puede proyectarse en un plano

vía vectores propios (PCA), o bien añadirse al conjunto de los datos disponibles como una nueva característica de las observaciones.

```
# best paartition
label <- cbds1$result$partition
```

```
#visualiza particio
fviz_cluster(list(data = dfcompact[,1:2], cluster = label))
```

Figura 29. Distribución de puntos en los 5 grupos generados

Para el caso 2, si fijamos como número óptimo de clústeres $k = 5$, la estabilidad que obtendremos no debería ser muy alta. Si observamos los valores obtenidos, veremos que están lejos de los valores anteriores e ideales de 1.

```
cbds2 <- clusterboot(dat2$x %>% scale(),clustermethod = kmeansCBI,
runs=100,iter.max=100,krange=5)
```

Para visualizar la estabilidad de las agrupaciones:

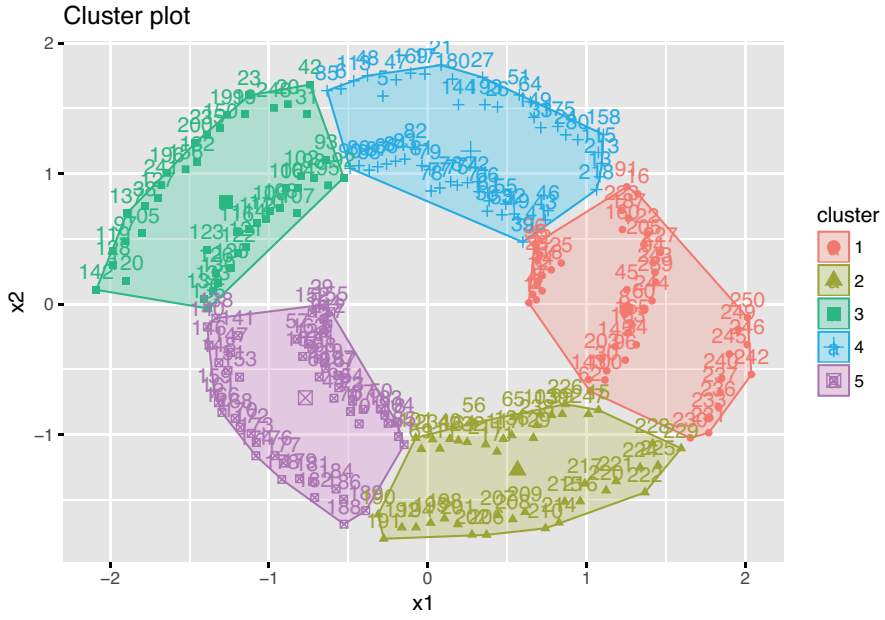
```
cbds2$bootmean
```

```
## [1] 0.5567000 0.6043370 0.6935833 0.6166994 0.7272915
```

y podemos almacenar la solución así como visualizarla.

```
label <- cbds2$result$partition
#visualiza particio
fviz_cluster(list(data = dfconnect[,1:2], cluster = label))
```

Figura 30. Visualización del valor óptimo por el método Gap Statistic para el caso 2

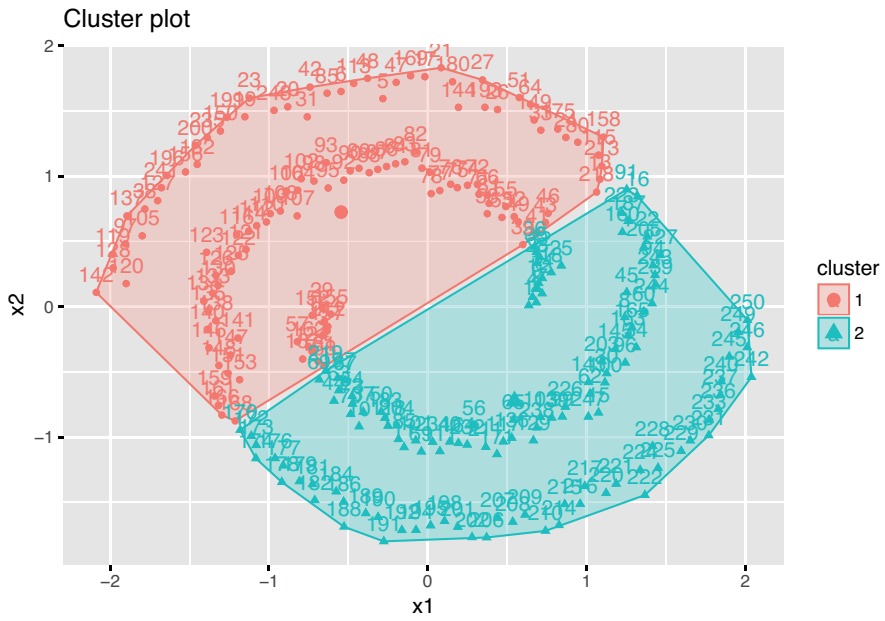


Si lo intentamos con dos clústeres, la estabilidad sube:

```
## [1] 0.8628799 0.8636076
```

Pero la solución no es ideal:

Figura 31. Visualización del valor óptimo por el método Gap Statistic para el caso 2



2.4.1. Caso mtcars

En el caso del conjunto de datos tratado en diversas ocasiones, ¿cuál es el número óptimo de grupos utilizando k-means? Si se realiza el procedimiento anterior, llegaremos a un óptimo de 2. La caracterización de estos dos grupos la podemos obtener de los centroides.

	mpg	cyl	disp	hp	drat
1	0.6440403	-0.7893528	-0.7679844	-0.7093044	0.5342642
2	-0.8280518	1.0148821	0.9874085	0.9119628	-0.6869112

	wt	qsec	vs	am	gear	carb
1	-0.6215850	0.4685997	0.6751327	0.4105508	0.4235542	-0.3310564
2	0.7991807	-0.6024854	-0.8680278	-0.5278510	-0.5445697	0.4256439

¿Y la distribución de observaciones dentro de los dos grupos?

¿Y las características sobre los datos no normalizados?

3. *Clustering* espectral

3.1. Introducción

La idea del clúster espectral es ser capaz de detectar agrupaciones que no siguen límites convexos. Estas situaciones se pueden visualizar con los ejemplos artificiales mostrados en el caso del *clustering* por k-means.

En ocasiones no podremos establecer si nuestros datos siguen estas estructuras, que en el caso de dos variables y proyectadas en un gráfico aparecen claramente caracterizadas. Este tipo de aproximaciones son aplicadas para la identificación de regiones en el procesado de imágenes.

3.2. Formulación matemática

El clúster espectral parte de una matriz de distancias o similitudes. Matemáticamente, suele usarse la terminología *matriz afín*, que mide la proximidad de cada par de puntos.

Esta matriz de afinidad usa una métrica que habitualmente es el kernel gaussiano. La idea es que cuando dos puntos estén muy próximos sean asignados al mismo clúster. Esta matriz de adyacencia representa aristas y nodos en un grafo. Con esta matriz se calcula la matriz laplaciana, que también es denominada de admitancia y que es una representación matricial de un grafo.

Los siguientes pasos son la solución de un problema general de vectores propios y la elección de un subespacio sobre los k vectores propios. En este subespacio se aplica un método de k-means para localizar los clústeres.

3.3. Aplicación práctica

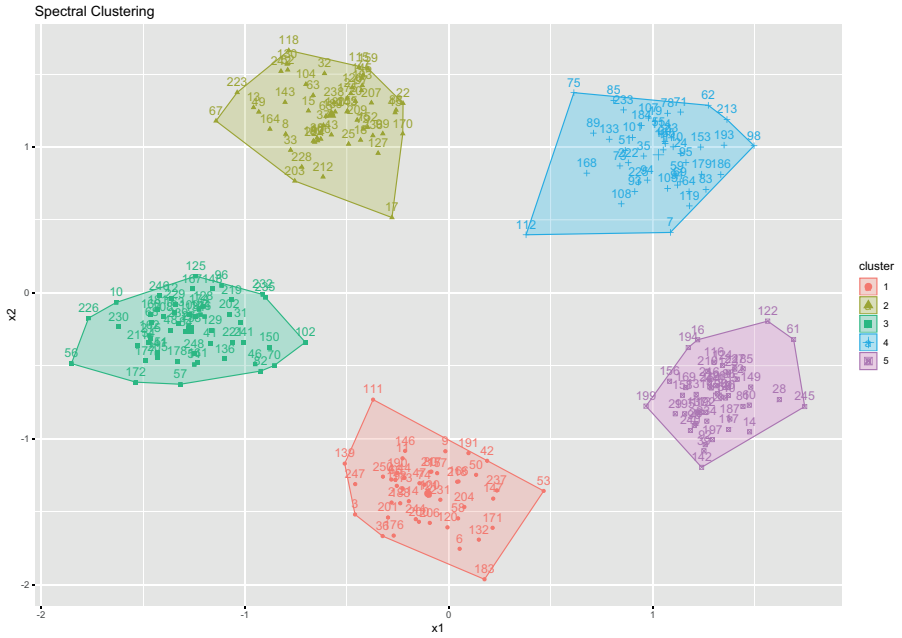
Existen diversas librerías para aplicar este método. La más utilizada es *kernlab*.

El metodo *specc* realiza el *clustering* espectral sobre la matriz indicada y con el número de grupos especificados.

```
# Utilizamos los datos del apartado anterior, datos compactos:
sc1 <- specc(dat1$x,centers=5)
```

```
# str(sc1) para apreciar estructura de los datos que genera

labelsc1 = as.data.frame(factor(sc1)) # Obtenemos grupos (no en formato ok)

#str(labelsc1)

label <- as.integer(labelsc1$`factor(sc1)`)

table(label)

#visualiza particio
fviz_cluster(list(data = dfcompact[,1:2], cluster = label))+ggtitle("Spectral Clustering")
```

Figura 32. Visualización del valor óptimo por el método Gap Statistic para el caso 2

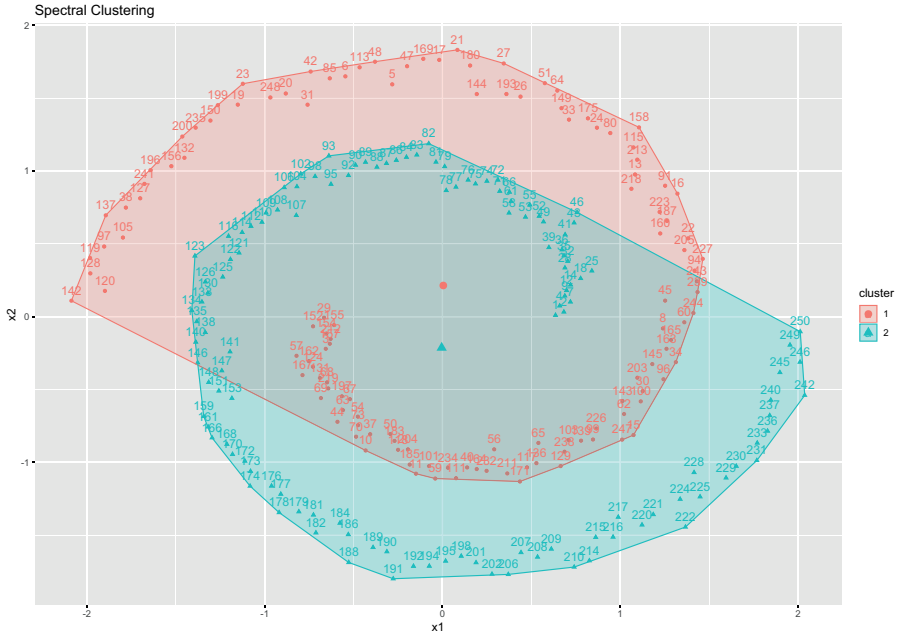


Con lo que podemos apreciar que el método funciona correctamente sobre el conjunto de datos compacto. Vamos a realizar un *clustering* para el conjunto de datos conectado sobre el que los métodos anteriores no funcionaban.

```
# Utilizamos los datos del apartado anterior, datos conectados:
sc2 <- specc(dat2$x,centers=2)

# str(sc2) para apreciar estructura de los datos que genera

labelsc2 = as.data.frame(factor(sc2)) # Obtenemos grupos (no en formato ok)

#str(labelsc2)

label <- as.integer(labelsc2$`factor(sc2)`)
```

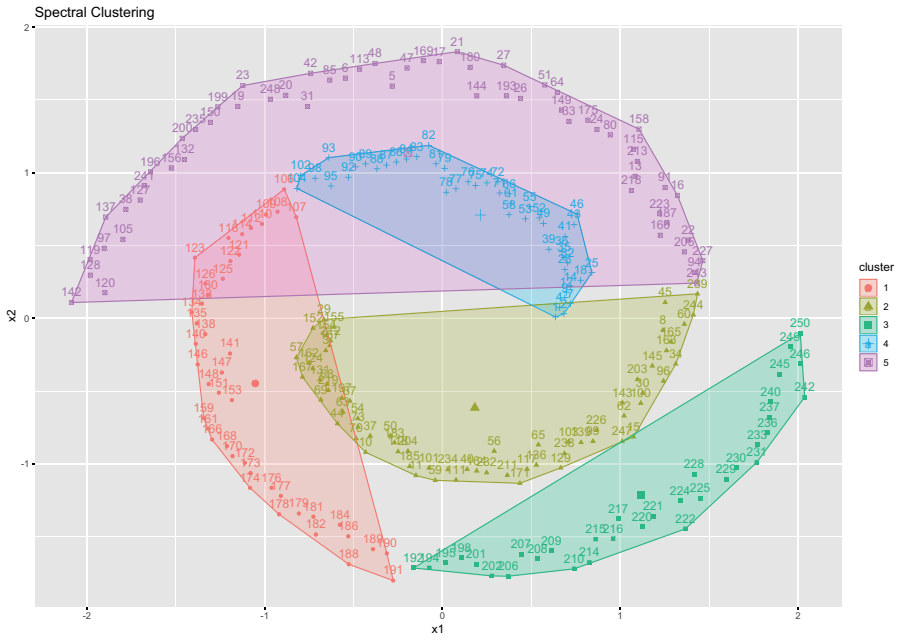
```
table(label)

#visualiza particio
fviz_cluster(list(data = dfconnect[,1:2], cluster = label))+
ggtitle("Spectral Clustering")
```

Figura 33. Visualización del valor óptimo por el método Gap Statistic para el caso 2



Una pregunta interesante es cómo responde el método si fijamos 5 grupos y no los 2 que sabemos que hay:

Figura 34. Visualización del valor óptimo por el método Gap Statistic para el caso 2



3.4. Clúster espacial basado en densidad (DBSCAN)

Este método permite obtener soluciones de interés cuando los datos están distribuidos de forma densa y el método los asigna al mismo clúster.

La librería dbscan permite realizar este tipo de *clustering*.

En estos casos no se especifica el número de clústeres, pero sí un parámetro *eps* que especifica la medida de la epsilos de la vecindad. Si es más pequeña, el procedimiento identificará más agrupaciones, mientras que si la eps es mayor, no tantas.

```
dbs1 = dbscan(dat2$x, eps = 0.12)
str(dbs1)
label <- dbs1$cluster

table(label)

fviz_cluster(list(data = dfconnect[,1:2], cluster = label))+ggtitle("Spacial Clustering")
```

Podemos testear para diferentes valores de eps. En este caso, hay eps = 0,12 y podemos verificar el efecto con eps = 0,1.

Para el caso de los datos artificiales se obtendrían los siguientes resultados:

Figura 35. DBSCAN eps = 0,12 para el caso 2

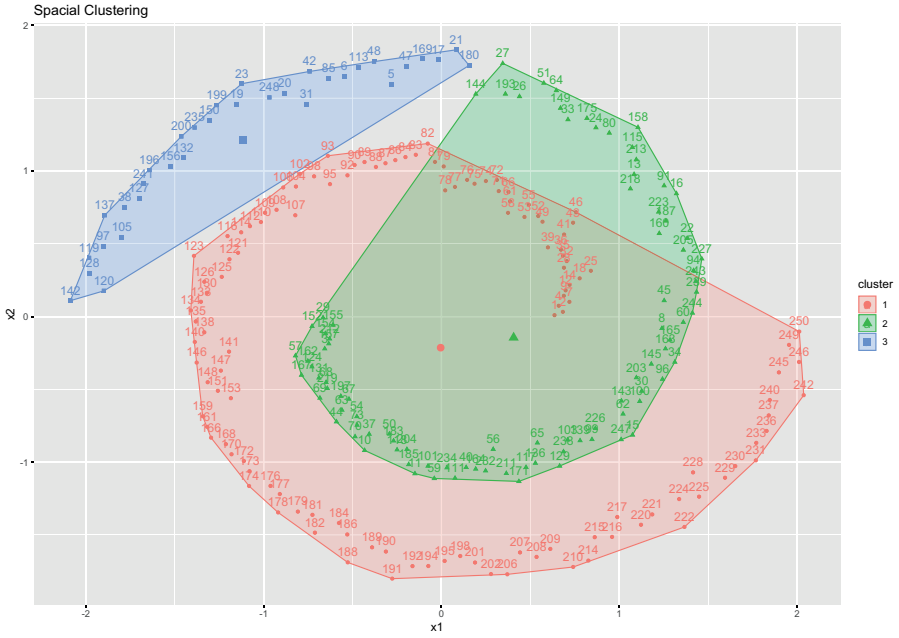
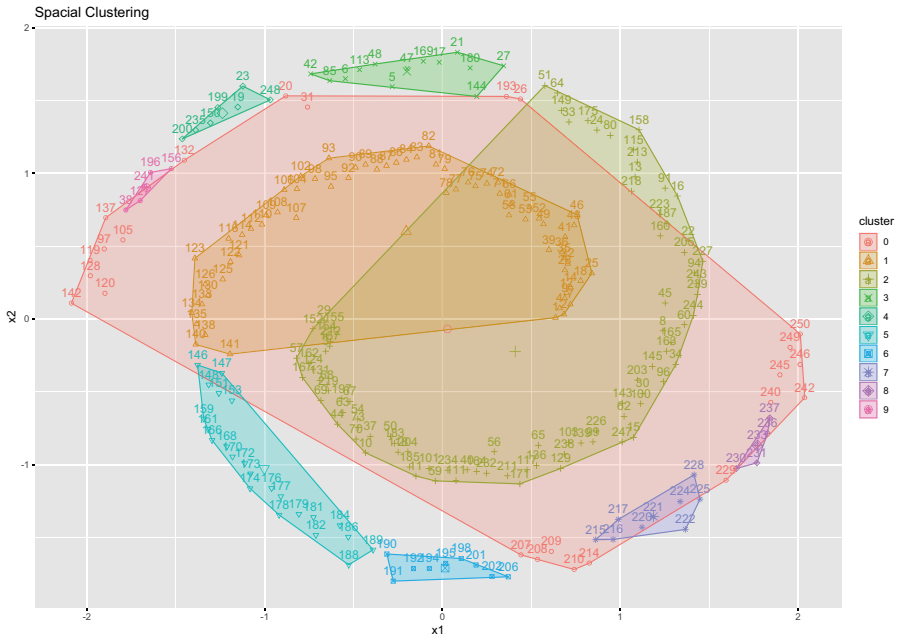
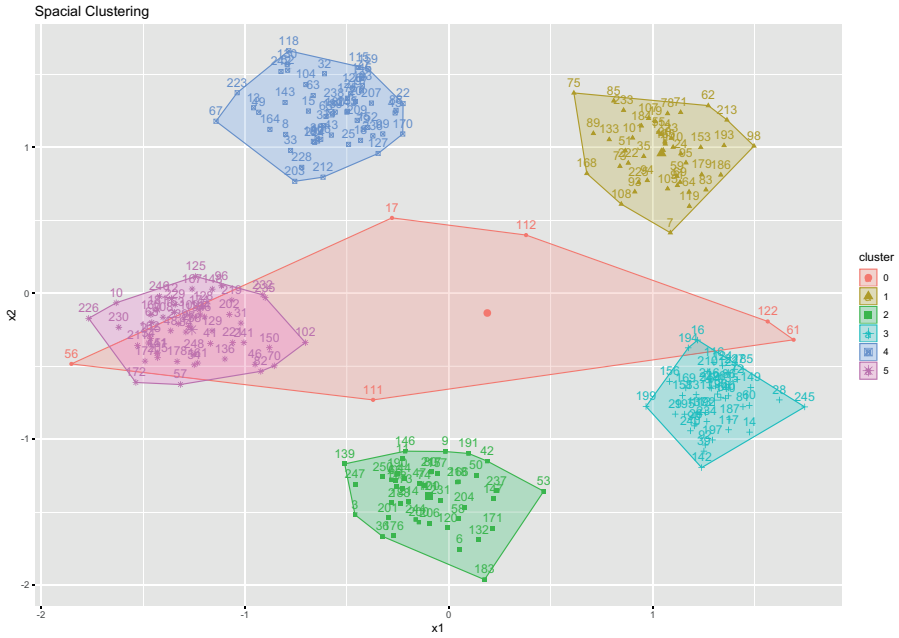


Figura 36. DBSCAN eps = 0,01 para el caso 2



Y para la situación compacta:

Figura 37. Visualización del valor óptimo por el método Gap Statistic para el caso 2



4. Caso práctico

4.1. Descripción

La idea de este apartado es, a partir de un juego de datos más complejo, aplicar las diferentes herramientas de clusterización descritas. En principio, se proponen los siguientes juegos de datos. Para cada uno de ellos se debe responder a una pregunta sencilla: ¿Cómo agrupamos las observaciones de una forma eficiente y que mejore nuestro conocimiento de los datos?

4.1.1. *Clustering* sobre los datos car

En la librería caret hay un conjunto de datos llamado cars, con 804 observaciones y un total de 18 variables. Estas variables recogen información de la reventa de vehículos de la empresa GM (General Motors) sobre modelos de 2005.

```
# Cargar la librería
library(caret)

# Cargar los datos
data(cars)

# visualizar la estructura de estos
str(cars)
'data.frame': 804 obs. of  18 variables:
 $ Price      : num  22661 21725 29143 30732 33359 ...
 $ Mileage    : int  20105 13457 31655 22479 17590 23635 17381 27558 25049 17319 ...
 $ Cylinder   : int  6 6 4 4 4 4 4 4 4 4 ...
 $ Doors      : int  4 2 2 2 2 2 2 2 2 4 ...
 $ Cruise     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Sound      : int  0 1 1 0 1 0 1 0 0 0 ...
 $ Leather    : int  0 0 1 0 1 0 1 1 0 1 ...
 $ Buick      : int  1 0 0 0 0 0 0 0 0 0 ...
 $ Cadillac   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Chevy      : int  0 1 0 0 0 0 0 0 0 0 ...
 $ Pontiac    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Saab       : int  0 0 1 1 1 1 1 1 1 1 ...
 $ Saturn     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ convertible: int  0 0 1 1 1 1 1 1 1 0 ...
```

```
 $ coupe      : int  0 1 0 0 0 0 0 0 0 0 ...
 $ hatchback  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ sedan      : int  1 0 0 0 0 0 0 0 0 1 ...
 $ wagon      : int  0 0 0 0 0 0 0 0 0 0 ...
```

Este fichero está principalmente pensado para explicar la variable Price (precio). En esta ocasión queremos plantear cuestiones en el ámbito de *clustering*.

Estas son las cuestiones que nos formulamos:

1) ¿Tenemos datos atípicos en el fichero?

2) En caso de localizar observaciones atípicas, ¿cuáles serían?

3) Utilizando algún método aglomerativo, ¿qué grupo es el más compacto (en el sentido de estabilidad)? ¿Cuántos vehículos están en este grupo? ¿Qué características de precio, kilometraje, número de cilindros y puertas tiene este grupo?

4) Utilizad un método de k-means para localizar el número óptimo de grupos. Mostrad las características de cada grupo. ¿Podríais decidir dónde va una nueva observación que no está en el conjunto de datos (es una observación nueva)?

5) Contrastad con un método diferente a los anteriores. ¿Qué resultados son más razonables?

4.1.2. *Clustering* sobre los datos Arrests

En la librería carData hay un conjunto de datos llamado Arrests, con 5.226 observaciones y un total de 8 variables. Estas variables recogen información de arrestos producidos por la policía de Toronto por posesión de pequeñas cantidades de marihuana.

```
# Cargar la librería
library(carData)

# Cargar los datos
data(Arrests)

# visualizar la estructura de estos
'data.frame': 5226 obs. of  8 variables:
 $ released: Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 2 2 2 ...
 $ colour  : Factor w/ 2 levels "Black","White": 2 1 2 1 1 1 2 2 1 2 ...
 $ year    : int  2002 1999 2000 2000 1999 1998 1999 1998 2000 2001 ...
 $ age     : int  21 17 24 46 27 16 40 34 23 30 ...
```

```
$ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 2 1 2 2 ...
$ employed: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 1 2 2 2 ...
$ citizen : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
$ checks  : int  3 3 3 1 1 0 0 1 4 3 ...
```

Este es un fichero que permite la aplicación de métodos de *clustering*. Observad que todas las variables podrían ser interpretadas como factores.

Preguntas que resolver en este caso:

1) ¿Qué métodos de los vistos son aplicables?

2) ¿Podemos agrupar variables y no observaciones?

3) ¿Qué observaciones presentan comportamiento extraño?

4) Generad una agrupación y facilitad la caracterización de las observaciones de este grupo.

5) ¿Podéis computar los tiempos de ejecución de los métodos aplicados? (*Benchmarking*).

Resumen

En general, los procedimientos de *clustering* son una buena herramienta para incrementar el conocimiento de los datos a nuestra disposición. Si bien existe una gran cantidad de opciones disponibles que hacen pensar en la dificultad para elegir la metodología óptima, esta nos permite ajustarnos a diferentes necesidades que debemos ser capaces de analizar y evaluar correctamente.

La utilización de métodos de *clustering* junto con métodos de clasificación es una aproximación interesante y utilizada en aquellas ocasiones en las que queramos dar explicación a la caracterización de estos grupos generados por los métodos de *clustering*.

Métodos clustering analizados:

- Jerárquicos
- K-means
- Clúster espectral (kernlab)
- Clúster espacial basado en densidad (DBSCAN)

Cuestiones que deben también considerarse tienen que ver con la facilidad de calcular los clústeres, si la dimensión de los datos se escala sustancialmente. Con conjuntos de datos de más de un millón de observaciones, ¿qué sentido tiene visualizar un dendograma? Quizá la parte importante es que el método de respuesta con un tiempo razonable.

Algunas librerías desarrolladas íntegramente para entornos *big data* como la H2O tienen únicamente la implementación del método de k-means: https://www.h2o.ai/.

H2O ofrece soluciones bajo una gran variedad de plataformas y entornos de desarrollo. Entre ellos, R, python, Spark, etc.