Universidad de Oviedo

Algoritmos de clustering: número de clusters

Jesús Abel García García

Dirigido por
Susana Montes Rodríguez y Noelia Rico Pachón

UNIVERSIDAD DE OVIEDO
Facultad de Ciencias
Grado en Matemáticas

Julio de 2022

Índice general

Capítulo 1

Introducción

1.1. Motivación

El análisis de datos es una rama que ha experimentado un gran desarrollo en los últimos 50 años. Ya desde 1962, estadísticos como Tuckey [27] veían el potencial del análisis de datos, el cual continua siendo actualmente un campo de investigación grande y en desarrollo (Jain et al. [13] , Chambers et al. [8], van de Vijver & Leung [28]) con métodos y técnicas adaptados a diversas aplicaciones. Además, debido al gran incremento de información disponible y al desarrollo de la tecnología en estos últimos años, se posibilita una mayor capacidad de almacenamiento y procesamiento de la información. Esto hace que las técnicas de análisis de datos sean cada vez más comunes en cualquier ámbito.

Todo análisis de datos busca responder unas preguntas previas, y la correcta formulación de las mismas es una de las tareas clave del investigador. Según el tipo de preguntas, los métodos pueden dividirse en: exploratorios, si no se conoce nada sobre los datos y se busca obtener patrones en la muestra que ayuden a comprenderlos; o confirmatorios, si las preguntas buscan comprobar ciertas hipótesis concretas de la población utilizando la muestra. El análisis confirmatorio usa técnicas clásicas como el contraste de hipótesis o modelización para resolver estas hipótesis, como en Wagenmakers et al. [29].

Por otra parte, el análisis exploratorio es un campo mucho más abierto y en el que la interpretación y el conocimiento de los límites de los métodos es fundamental, siendo muy difícil comprobar la efectividad y fiabilidad de los resultados en la mayoría de casos.

Dentro del campo del análisis exploratorio, se pueden utilizar distintos procesos en función de los datos. Una de las técnicas más utilizadas es el *data clustering*, que consiste en tratar de agrupar los datos con el fin de interpretar qué clase de objetos pertenecen a cada uno de los grupos creados. Este proceso, en general, sigue los siguientes pasos:

1. Estudio y adecuación de los datos al dominio del problema realizado de forma previa a la clasificación. Por ejemplo, entre otros: estudio de variables redundantes, selección de variables claves para el estudio o transformación de las misma para garantizar agrupaciones significativas.
2. Agrupación de los datos en distintas clases haciendo uso de uno o varios métodos de aprendizaje.
3. Interpretación de los resultados, tanto análisis de su validez como las consecuencias que se pueden extraer del análisis.

Debido a la gran variedad de datos y contextos en los que se usa el *clustering*, hay muchos tipos de técnicas diferentes. Según el conocimiento previo de los datos y los grupos, se pueden dividir en 2 grandes grupos:

- **Aprendizaje Supervisado:** Técnicas en las que se conocen todos los grupos previamente y se parte de un conjunto de datos ya clasificados. Usando estos elementos se entrena el modelo, con el fin de utilizarlo posteriormente para clasificar datos futuros que carecen de etiqueta.
- **Aprendizaje no Supervisado:** Técnicas que tratan datos donde no se tiene ningún conocimiento previo sobre la pertenencia a diferencias clases o cantidad de grupos en la muestra.

Dentro de la categoría de aprendizaje supervisado se pueden encontrar métodos como las máquinas vector soporte, las redes neuronales, o los árboles

de decisión. Estos últimos (Myles et al. [18]), tratan de segmentar iterativamente los datos utilizando la variable que mejor divida el conjunto hasta llegar a la decisión óptima. La idea es construir un grafo de tipo árbol, donde los vértices representan distintos estados y las aristas las decisiones que se toman para segmentar la muestra. En el contexto de la clasificación, la raíz del árbol es el conjunto total de los datos y se van construyendo vértices en función a subparticiones de los datos. Los árboles de decisión permiten tratar con datos de muchos tipos y dar una interpretación sencilla, pero son muy sensibles a pequeños cambios en el conjunto y cuando se tienen varias variables categóricas con distinto número de niveles, tienden a dar más peso a las de mayor número de categorías. Una alternativa robusta a los árboles de decisión son los *random forests* (Breiman [4]), o bosques aleatorios, que utilizan técnicas de muestreo como el boostrap aplicadas a grupos de árboles de decisión para lograr resultados más exactos y menos sensibles a pequeños cambios, a cambio de una mayor dificultad a la hora de interpretar los resultados.

Los métodos supervisados necesitan tener datos previamente etiquetados dentro de las distintas agrupaciones. Cuando no se tienen datos etiquetados, o las etiquetas disponibles se corresponden con una estimación previa, se recurre a métodos no supervisados. Estos algoritmos suelen ser más complejos por el hecho de no saber cuál es la clasificación final real que se pretende alcanzar, y por tanto no solo es difícil sacar conclusiones, sino también saber lo buenas que han sido las mismas. Los algoritmos no supervisados son habitualmente conocidos como algoritmos de *clustering*, ya que crean grupos (en inglés, *clusters*), de objetos acorde a su similitud. Algunos de los algoritmos más comunes son el método k-means y el *clustering* jerárquico. Este trabajo se centrará principalmente en estos métodos, que serán descritos en profundidad en el próximo capítulo.

El objetivo de este trabajo es presentar las bases del *clustering* de datos, una técnica versátil y que con un buen entendimiento de los conceptos puede ser útil para aportar información en una gran variedad de problemas. Además, estas técnicas se aplicarán a un ejemplo concreto utilizando datos sobre música, una rama donde la ciencia de datos juega un papel cada día más importante y la agrupación de canciones según distintos parámetros es una práctica muy común (Curram & Mingers [9], West & Lamere [31], Luo et al. [16]).

Más concretamente, el objetivo de este trabajo puede ser desglosado en los siguientes puntos:

- Plantear el problema del *clustering* de datos desde un punto de vista estadístico y definir los elementos clave a la hora de establecer criterios de similitud.
- Estudiar métodos de *clustering* basados en distintos criterios: particionales, jerárquicos, de densidad y difusos. Comprender su funcionamiento y el trasfondo teórico para discernir cuándo es correcto usar cada uno.
- Analizar distintas métricas para evaluar las particiones realizadas por los algoritmos estudiados. Definir las métricas y dar un razonamiento teórico de las mismas para poder interpretar los resultados de forma correcta.
- Aplicar los conceptos y técnicas comentados a un conjunto de datos reales, con el fin de observar el comportamiento de las técnicas aprendidas en la práctica.

1.2. Estructura del trabajo

Derivado de todo lo anterior, este trabajo se estructura como sigue: en el Capítulo 2 se comienza con planteamiento del problema del *clustering* de una forma matemática, para seguidamente definir 4 grandes tipos de algoritmos: particionales, jerárquicos, de densidad y difusos. En el Capítulo 3 se desarrollan distintas técnicas de validación de *clusters*, concretamente los índices CH, Silueta y Rand. De cada una de las partes desarrolladas en el Capítulo 2 se desarrollarán dos algoritmos que serán utilizados más adelante en el Capítulo 4, en el cual se aplican todos los elementos y algoritmos explicados anteriormente para realizar un estudio de un conjunto de datos real construido expresamente para este trabajo: se buscará identificar géneros musicales a partir de una serie de características musicales.

Capítulo 2

Clustering: Algoritmos y métricas

Antes de comenzar el estudio de los distintos métodos e índices, es necesario establecer una notación clara y dar las definiciones necesarias para poder desarrollar el tema con precisión. En este capítulo se van a introducir elementos y terminologías utilizados para explicar los distintos algoritmos de *clustering*.

2.1. Planteamiento del problema

El problema a tratar es obtener información de una muestra de datos $X = \{x_1, \ldots, x_n\}$, con p variables cada uno, que se quiere dividir en $\{C_1, \ldots, C_k\}$ *clusters*. Para un planteamiento más formal, se trata el espacio muestral X como un subconjunto de $\mathbb{R}^p$, donde cada elemento es un vector $x \in \mathbb{R}^p$. Esto es posible solo si todas las variables son números reales. En la realidad, es común el uso de variables categóricas o discretas que no encajan dentro de $\mathbb{R}^p$, a las cuales se les aplican transformaciones para ser convertidas en numéricas. Las p variables son las responsables de determinar como se realiza el *clustering*, y por tanto es necesario establecer una noción de similitud entre datos. Esto no es para nada trivial y es algo completamente

necesario, ya que sin imponer unas ciertas suposiciones previas cualquier par de objetos podría llegar a verse similar (se demuestra en el conocido como *Ugly Duckling Theorem* de Watanabe [30]). La mayor ventaja de considerar todas las variables numéricas, motivo por el cual se aplican habitualmente transformaciones a las variables categóricas, es que permite hacer uso de la noción matemática de distancia, un concepto claro y concreto que permite establecer un criterio para medir la similitud entre objetos a la hora de agrupar elementos.

Definición 2.1. *Sea Y un conjunto de elementos, se denomina a la aplicación $d : Y \times Y \longrightarrow \mathbb{R}$ una métrica sobre Y si cumple $\forall x, y, z \in Y$:*

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) \leq d(x, z) + d(y, z)$

A continuación, siendo $X \subset \mathbb{R}^p$ una muestra con $x_i = (x_{i_1}, \ldots, x_{i_p}), x_j = (x_{j_1}, \ldots, x_{j_p}) \in X$, se enumeran las distancias más comunes en el contexto del *data clustering*:

- **Distancia euclídea:** $d(x_i, x_j) = \sqrt{(x_i - x_j)^T (x_i - x_j)}$
  Es la distancia más estándar e intuitiva. Da unas agrupaciones de fácil visualización, lo que la hace muy popular con datos en 2 o 3 dimensiones, y detecta bien *clusters* compactos y aislados con una tendencia a darles forma circular. Sin embargo, uno de sus principales inconvenientes es que si una variable es mucho mayor que el resto, los grupos se verán dominados por esta, no permitiendo ver grupos o tendencias inducidos por otras variables menos prominentes, pero no por ello menos importantes. La solución más común y sencilla es normalizar todas las variables.
- **Distancia de Mahalanobis:** $d(x_i, x_j) = \sqrt{(x_i - x_j)^T \sum^{-1} (x_i - x_j)}$
  $\sum$ es la matriz de covarianza de los datos x_i y x_j. Esta distancia es una variación de la euclídea, haciéndola invariante respecto al escalado de los datos y por tanto suavizando el protagonismo de las tendencias

dominantes. Los *clusters* pasan a tener una tendencia elíptica en lugar de circular. El único inconveniente es que para usarla se debe suponer implícitamente que las densidades marginales de cada clase son Gausianas, cosa que no siempre es cierta en casos prácticos.

- **Distancia de Canberra:** $d(x_i, x_j) = \sum_{k=1}^{p} \frac{|x_{i_k} - x_{j_k}|}{|x_{i_k}| + |x_{j_k}|}$

  Esta distancia es la distancia conocida como L_1 pero con un factor de re-escalado $(|x_{i_k}| + |x_{j_k}|)^{-1}$. Es una buena distancia para detectar agrupaciones rectangulares y destaca por su mayor rapidez a nivel computacional respecto a la euclídea y derivados. Su mayor inconveniente es su gran sensibilidad en datos con valores cercanos a 0.

- **Distancia de Chebychev:** $d(x_i, x_j) = \max_{1 \leq k \leq p} (|x_{i_k} - x_{j_k}|)$

  También conocida como distancia del máximo por ser inducida por la norma infinito, es una distancia también muy simple computacionalmente y que agiliza la comparación de datos. El precio a pagar es la mayor dificultad detectando formas de *clusters*, y debido a que solo se fija en la variable más dispar siempre, tiene el mismo problema que la distancia euclídea a la hora de detectar tendencias menores cuando hay gran disparidad entre las variables.

La cantidad de distancias posibles es realmente extensa, para un estudio más en detalle de las distintas métricas y sus propiedades se puede recurrir a [10] y [15].

En algunos casos, en lugar de dar los datos en una muestra, se dan sus distancias relativas ya sea por mayor comodidad a la hora de trabajar o porque solo se conocen estas.

Definición 2.2. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos y d una métrica, se define la matriz de distancias de X con respecto a la métrica d como la $n \times n$ matriz de la forma $\hat{X} = \{d(x_i, x_j)\}_{i,j}$ con $i = 1, \ldots, n; \quad j = 1, \ldots, n$.*

Una vez planteado el problema y establecida una métrica, conviene definir una serie de expresiones y funciones que más adelante se usarán en los diferentes algoritmos. La función matemática en la que se base el algoritmo determinará la forma de los *clusters* y las agrupaciones de elementos, por lo que es importante entender bien los conceptos sobre los que se basa cada algoritmo.

Definición 2.3. *Sea X una muestra de datos y $\{C_s\}_{s=1}^{k}$ una partición de la misma, se define el centroide del* cluster *C_r como el p-vector:*

$$c_r = \frac{1}{|C_r|} \sum_{x_i \in C_r} x_i$$

Este centroide corresponde a la media de todos los elementos de un *cluster* respecto a una variable determinada, lo que nos hace pensar en definir también un equivalente de la varianza aplicado a los *clusters*.

Definición 2.4. *Sea X una muestra de n datos y $C_1, \ldots, C_k$ una partición. Se definen la suma de distancias intra-cluster S_w e inter-cluster S_b como:*

$$S_w = \sum_{k=1}^{q} \sum_{\substack{x,y \in C_k \\ x<y}} d(x,y) \qquad\qquad S_b = \sum_{k=1}^{q-1} \sum_{l=k+1}^{q} \sum_{\substack{y \in C_l \\ x \in C_k}} d(x,y)$$

La distancia S_w indica la separación promedio por *cluster* de los elementos del mismo, mientras que S_b simboliza la separación promedio entre distintos *clusters*. De forma intuitiva, una buena partición debería dar *clusters* de elementos similares y fácilmente diferenciables, lo que implica una S_w baja y una S_b alta. De esta intuición surgen distintos índices de validación y hasta varios tipos de algoritmos particionales que se basan en minimizar o maximizar una de estas funciones o alguna variación de las mismas. Es necesario tener en cuenta que los centroides implícitamente usan la métrica euclídea, mientras que las funciones de suma de distancias tienen una definición más general. Esto intuitivamente indica que los algoritmos que usen centroides deberían utilizar esta misma distancia para mantener una cierta consistencia.

Más adelante se tratará el problema de agrupar datos con un enfoque difuso. La base de la lógica difusa es que los argumentos tienen un grado de verdad, no son verdades o mentiras como en la lógica clásica Booleana. Traduciendo esto al *clustering*, los datos dejarán de estar limitados a pertenecer a un único *cluster*, y tendrán en su lugar un grado de pertenencia a cada grupo. Esto crea la necesidad de incorporar un nuevo elemento para caracterizar un problema: la matriz de grados de pertenencia o pesos.

Definición 2.5. *Sea X una muestra de n datos que se quieren repartir en k* clusters*. Se define la matriz de pesos U como la $k \times n$ matriz tal que $U = \{\mu_{ij}\}$ con $i = 1, \ldots, k$ y $j = 1, \ldots, n$, y cumple:*

1. $\mu_{ij} \in [0, 1] \quad \forall i = 1, \ldots, k; \quad j = 1, \ldots, n$
2. $\sum_{i=1}^{k} \mu_{i,j} = 1 \quad \forall j = 1, \ldots, n$
3. $0 < \sum_{j=1}^{n} \mu_{ij} < N \quad \forall i = 1, \ldots, k$

Cada μ_{ij} representa el grado de pertenencia del elemento j al cluster i.

Con esto ya se tiene la base sobre la que se desarrollaran varios algoritmos difusos. En la sección correspondiente se elaborará sobre estos conceptos y como se construyen algoritmos para obtener esta matriz U, que pasará a caracterizar la partición final en lugar de una representación directa de los *clusters*. Es posible obtener una partición no difusa de los datos a partir de U, pero se hablará en más detalle sobre esto en la sección correspondiente.

2.1.1. Limpieza de los datos

Habitualmente es necesario hacer una limpieza previa de los datos antes de aplicar los algoritmos de *clustering*. Una de las técnicas habituales es hacer un análisis de componentes principales, conocido comunmente como PCA por sus siglas en inglés *Principal Component Analysis*. El objetivo del PCA es hacer una reducción dimensional del conjunto de datos usado. La idea intuitiva en la que se basa el método es tratar de reducir la dimensión del problema manteniendo la máxima varianza de los datos gracias a transformaciones lineales.

Partiendo de un conjunto de datos $X \subset \mathbb{R}^p$, se busca realizar una transformación ortogonal lineal a otro sistema de coordeandas de forma que los vectores (o variables) maximicen la varianza de los datos. Matemáticamente,

el resultado deseado es un sistema de vectores de tamaño p formados por una combinación lineal de las variables originales junto a unos coeficientes de peso $w = (w_1, \ldots, w_p)$. Para encontrar los pesos deseados, se impone una condición de normalización ($||w|| = 1$) y se aprovecha la condición de maximizar la varianza. Para obtener el primer vector, se debe resolver la condición:

$$argmax(\sum_{i=1}^{n}(x_i \cdot w)^2) = argmax(||Xw||^2) = argmax(w^T X^T X w)$$

Aprovechando que los vectores de pesos se buscan normalizados, se reescribe la expresión como $argmax(\frac{w^T X^T X w}{w^T w})$, que puede resolverse con resultados de análisis matricial aprovechando que $X^T X$ es semidefinida positiva. El resultado es que w debe ser el primer autovector de la matriz $X^T X$, que es algo proporcional a la muestra empírica de la matriz de varianzas y covarianzas. De esta forma, se puede obtener un vector de coeficientes para realizar la transformación lineal al vector que mayor varianza conserva de los datos.

Se puede proceder de forma iterativa para seguir obteniendo vectores hasta el $k-$ésimo vector que más varianza explica de los datos. Para esto, se supone que se tienen ya $k-1$ vectores de coeficientes w_i calculados. Eliminando estas $k-1$ componentes principales calculadas, se recalcula el conjunto de datos y se realiza un proceso análogo para el w_k.

$$\hat{X} = X - \sum_{i=1}^{k-1} X w_i \tag{2.1}$$

$$w_k = argmax(\frac{w^T \hat{X}^T \hat{X} w}{w^T w}) \tag{2.2}$$

Se puede iterar este proceso hasta obtener p componentes principales que contendrán el 100 % de la varianza del conjunto de datos, pero reordenados de forma que la cantidad de varianza de cada vector del conjunto $\{w_i\}_{i=1}^{p}$ es creciente con i. Componentes con poca varianza no ayudarán mucho al conjunto y se pueden omitir, siendo esta la forma de reducir la dimensión del conjunto de datos inicial. Una buena norma usada experimentalmente es que alrededor del 80 % se guarda ya casi toda la información útil. Este método será usado más adelante en el conjunto de datos real para identificar una serie de características dentro de un subconjunto de datos.

2.2. Algoritmos de clustering

Una vez dada una definción formal del problema de agrupación de datos y establecidos los coneptos báiscos necesarios para los algoritmos usados en este trabajo, se procede a desarrollar métodos de *clustering*. Con el fin de introducir estos algoritmos de forma ordenada, esta sección se subdivide en 4 grandes tipos diferentes de algoritmos: particionales, jerárquicos, densidad y fuzzy.

2.2.1. Métodos particionales

Los métodos particionales comienzan considerando una división inicial aleatoria de toda la muestra, la cual van refinando de forma iterativa.

k-means

Una de las técnicas de clustering más habituales es *k-means*, un algoritmo iterativo basado en minimizar la suma de cuadrados de las distancias *intra-cluster* para un número k de *clusters*. Esta suma de cuadrados se expresa como:

$$SSE(X, c_1, \ldots, c_k) = \sum_{r=1}^{k} \sum_{x \in C_r} \frac{1}{2}(x - c_r)^2$$

Es muy importante notar que la elección de la norma euclídea es clave para garantizar la convergencia del algoritmo a un óptimo local. Sin embargo, en ciertos casos prácticos se utilizan otro tipo de distancias ya que dan resultados mejores que la mayoría de algoritmos, pero la elección del *k-means* en estos casos es puramente heurística (Singh et al. [25], Santhi & Murali [24]).

Dada una muestra de datos X y un número de *clusters* k, se eligen k elementos de X de forma aleatoria (más adelante se verá que usar determi-

nados criterios para la selección de estos elementos puede ser clave para la partición final obtenida) y se toman como centroides de la partición inicial. Para completar la partición de los datos, se reparten de forma que $x_i \in C_l$, con l tal que:

$$\min_{r=1,\ldots,k} d(x_i, c_r) = d(x_i, c_l)$$

Con esta partición, se vuelven a recalcular los centroides y se reitera el proceso de asignación de elementos a cada grupo. Como criterios de parada es común utilizar la convergencia (los centroides no cambian de una iteración a otra), comprobar si SSE se reduce menos que un cierto valor establecido previamente como umbral o añadir un número máximo de iteraciones (si se llega a este criterio es recomendable pensar en otros algoritmos ya que no se tiene garantizada la convergencia de SSE).

Proposición 2.1. *Sea $X \subset \mathbb{R}^p$ una muestra de datos, d la distancia inducida por la norma euclidea y $c_1, \ldots, c_k$ k centroides de una partición. La función $SSE(X, c_1, \ldots, c_k)$ no puede incrementar tras realizar una iteración del algoritmo* k-means*.*

Demostración. En primer lugar, se obtiene cual es la expresión de los c_r que optimiza la función:

$$\frac{\partial}{\partial c_r} SSE = \sum_{x \in C_r} (x - c_r) = 0 \Longleftrightarrow c_r = \frac{1}{|C_r|} \sum_{x \in C_r} x$$

con $|C_r|$ el número de datos asignados a C_r. Por tanto, tomar los centroides de cada *cluster* para la próxima iteración da lugar a la partición óptima. Además, derivando otra vez es fácil ver que será un mínimo de la función suma de errores cuadrados: $\frac{\partial^2}{\partial c_r^2} SSE = 1 > 0$
Si con estos nuevos centroides no cambia ninguna asignación de los x entre los grupos, eso quiere decir que se ha vuelto a llegar a la misma partición óptima y por tanto se ha alcanzado un mínimo. En otro caso, se recalculan una vez más los centroides, dando lugar a un nuevo mínimo de la SSE y se vuelve a iterar el proceso. Como siempre se va de mínimo en mínimo, no es posible que el valor de SSE aumente. □

Aunque se ha probado que el algoritmo mejora la SSE siempre, no se puede garantizar que el mínimo que alcance la función sea global, siendo de

hecho en la mayoría de ocasiones un mínimo local. El mínimo obtenido viene determinado por la elección de los centroides inciales. Esto ocurre porque el fundamento matemático del algoritmo se puede interpretar como el del método de Newton, en el que la semilla inicial irá al óptimo más cercano, sin importar si es global o local, no logrando ver el mínimo global en el caso de que existiera[3].

La simpleza del *k-means* se consigue con un precio, y es la cantidad de hipótesis previas necesarias para el buen funcionamiento del algoritmo. Una de las más limitantes es la necesidad de formas circulares (o de n-esferas cuando se tienen más de dos variables) para los *clusters*, que junto a la alta sensibilidad al ruido lo hacen poco práctico para datos sin un tratamiento previo. El ruido tiene un gran efecto porque cada punto (independientemente de si es ruido o real) en una clase tiene el mismo peso a la hora de fijar el centroide. El ruido es un problema que se soluciona relativamente fácil (más en la sección de algoritmos de densidad), sin embargo, la forma circular de los *clusters* es algo que viene de la distancia euclídea y del propio uso del centroide, ya que se está imponiendo implícitamente que los datos siguen una distribución paramétrica en la que cada *cluster* viene generado por una gausiana de media su centroide. Cuando se sabe que las formas de los *clusters* van a ser arbitrarias o muy complejas no es recomendable usar *k-means*, aunque la realidad en la práctica es que se utiliza habitualmente como inicio de cualquier análisis. Para ejemplificar las distintas características de los algoritmos, se hará uso de una colección de conjuntos de datos artificiales[1], y a partir de estos se construirán *clusterings* de ejemplo. El primer ejemplo se puede observar en 2.1, donde se muestra como el *k-means* no es capaz de ver las formas complejas.

En la Figura 2.2 se aprecia como el algoritmo da dos particiones muy diferentes de un mismo conjunto de datos aunque las formas son elipsoidales y relativamente sencillas de ver para el *k-means*. Aprovechando que el conjunto de datos es artificial y se conoce la verdadera agrupación de los datos, se puede analizar donde está el error. El problema es que, en la primera partición dos centroides se inicializan en el *cluster* circular. Al estar tan cercanos y

[1]Los datos han sido obtenidos de: https://github.com/deric/clustering-benchmark

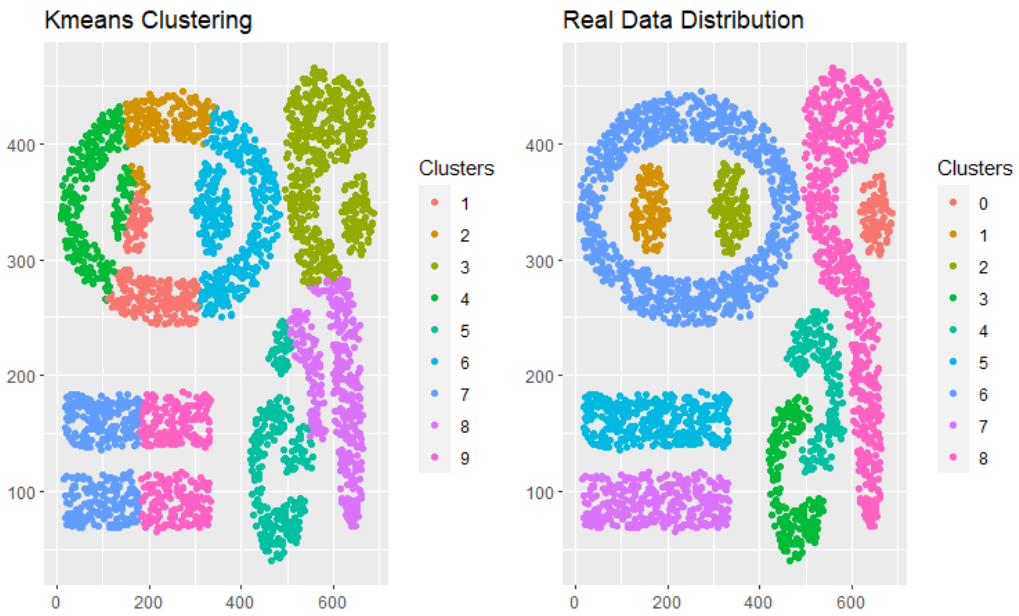


Figura 2.1: Datos artificiales generados a partir de distintas formas geométricas sin ruido. A la izquierda se observa la partición generada por *k-means*, que no es capaz de imitar la partición real que se observa a la derecha.

dentro del mismo grupo, *k-means* separa el *cluster* original en dos, lo que luego lleva a la agrupación de dos *clusters* en otro lugar para mantener el número k de grupos exigido. En definitiva, si varios centroides se sitúan muy cerca o dentro del mismo *cluster* real, no se alcanzará una partición cercana a la óptima. Permitir división de *clusters* cuando pasen un cierto umbral de variación *intra-cluster* o unirlos si la distancia de sus centros es menor a un cierto valor suaviza y corrige este tipo de errores, pudiendo llegar a la partición óptima desde cualquier punto, siempre y cuando se tomen los valores límite para estas uniones y separaciones adecuados. Algoritmos como el *isodata* contemplado en [2] usan este tipo de técnicas.

Aún con todos estos inconvenientes, la sencillez tanto matemática como computacional del *k-means* hace que se siga utilizando y se desarrollen una gran variedad de alternativas y mejoras para suplir sus limitaciones. Principalmente se ha trabajado mucho en la inicialización de los centroides para garantizar llegar a un mínimo cercano al global y la utilización de otros parámetros alternativos, para poder usar otras métricas no euclídeas sin perder la convergencia y así aprovechar las propiedades de estas otras distancias. Entre estos últimos destacan algoritmos como el *k-median* que toma

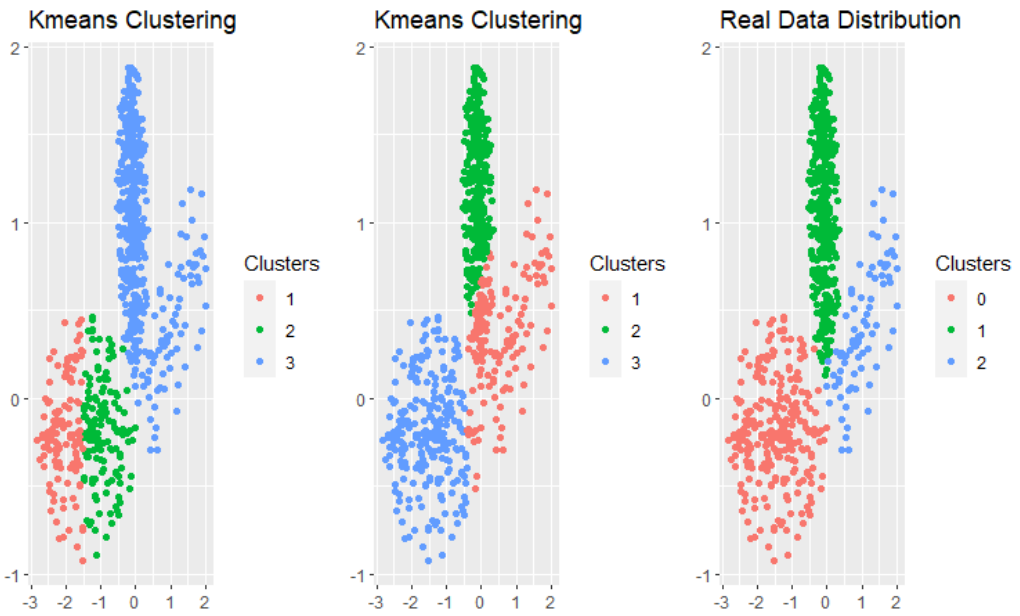


Figura 2.2: Comparación de un conjunto de datos artificial utilizando *k-means* con la clasificación real. Izquierda a derecha: *k-means* con mala inicialización, *k-means* con buena inicialización y clasificación real de los datos.

las medianas en lugar de los centroides y se suele usar con la distancia de Manhattan, ya que al buscar la partición óptima como en la Proposición 2.1, se obtienen las medianas en lugar de los centroides al optimizar la función SSE; y el *k-medoids*, que usa el concepto de *mediod* (elemento dentro del *cluster* que esta más cerca del resto simultáneamente) en lugar de centroide, aumentando la robustez del algoritmo frente a datos extremos. Entre las variaciones que afectan a la inicialización, destaca el *k-means++*.

k-means++

El *k-means++* es una variación del *k-means* muy popular que trata de alejar los centroides lo máximo posible entre sí, reduciendo la posibilidad de que dos centroides cercanos distorsionen un grupo de datos como en la Figura 2.2. Para ello, se procede de la siguiente manera:

1. Fijar el primer centroide c_1 de entre los n datos, con una distribución

uniforme. A partir de ahora, se llamará $d_{max} = \underset{x,y \in X}{max} d(x,y)^2$.

2. Para obtener el segundo centroide, tomar una función de probabilidad cuyos pesos sean las distancias al centroide c_1 al cuadrado, es decir: $P(x_j = c_2) = \frac{d(x_j,c_1)^2}{d_{max}}$

3. Iterar hasta tener todos los centroides fijados; con r centroides fijos, la dsitribución de probabilidad para el $r+1$ centroide será: $P(x_j = c_{r+1}) = \frac{1}{d_{max}^r} \prod_{k=1}^{r} d(x_j, c_k)^2$

De esta manera la elección de centroides inicial sigue siendo aleatoria pero garantizando grupos mejor distribuidos y disminuyendo la probabilidad de que dos centroides comiencen dentro del mismo *cluster*. Aun así, no desaparece totalmente el problema de la convergencia a mínimos locales, y sigue manteniendo todos los inconvenientes principales del *k-means* sobre formas y ruido.

Como se puede observar, todas estas variaciones que mejoran el *k-means* suelen acarrear una mayor complejidad tanto computacional como matemática, siendo la mejora que aportan a veces muy pequeña. Por ello, otra alternativa para asegurar una buena partición del algoritmo, y de las más usadas en la práctica, es sencillamente inicializar el algoritmo varias veces y quedarse con el que da una menor función objetivo. Con este enfoque probabilístico en el que tras suficientes intentos se acaba llegando al mínimo deseado, no se solucionan los problemas sobre forma de *clusters* o ruido en la muestra, pero sí se consigue reducir el impacto de la inicialización sin aumentar la complejidad y sin la necesidad de mucho más tiempo de ejecución gracias a la rapidez del propio *k-means*.

2.2.2. Clustering jerárquico

Mientras que *k-means* necesita una cierta partición inicial y va modificando estos *clusters* iniciales hasta converger a una solución, los algoritmos jerárquicos van reduciendo o aumentando el número de *clusters* hasta cumplir unos parámetros deseados o que no se puedan realizar más uniones o divisiones. Generalemente, se pueden clasificar en 2 tipos:

- Aglomerativos: Se empieza con tantos *clusters* como datos se tengan y se van uniendo basándose en distancias y similitud
- Divisivos: Se empieza con un único *cluster* que se va separando en función una vez más de métricas y distancias

Aunque sus principios son similares, en general se suelen considerar sobretodo algoritmos aglomerativos por una razón puramente computacional. Cuando se tiene un algoritmo aglomerativo, usando un conjunto de datos de n objetos, en el momento inicial el número total de uniones entre *clusters* posibles es $\frac{1}{2}n(n-1)$, que crece cuadráticamente con n pero aun así da un número de combinaciones computable incluso para un gran número de datos. Sin embargo, en un algoritmo divisivo todas las posibles maneras de dividir el *cluster* inicial en en una partición de 2 *clusters* es $2^{n-1}-1$; esta dependencia exponencial hace que incluso conjuntos de datos de tamaño medio sean incomputables y limita fuertemente las opciones. Por tanto, la elección en muchos casos prácticos que requieren trabajar con grandes cantidades de datos es usar métodos aglomerativos. Aunque existen manera de inicializar los métodos divisivos reduciendo su complejidad temporal y dejándolos en una posición similar a la de muchos métodos aglomerativos, esto queda fuera del área de estudio de este trabajo (se recomienda [17] para un estudio más en detalle de estas técnicas de implementación del algoritmo). Por tanto, de ahora en adelante se hablará indistintamente de *clustering* jerárquico y de *clustering* aglomerativo.

La principal característica y ventaja del *clustering* jerárquico es que permite obtener una estructura del proceso de agrupación, que puede representarse en forma de un diagrama similar a uno de árbol conocido como *dendrograma* (Figura 2.3). Éste, que debe de ser interpretado de abajo hacia arriba, representa cómo los datos se van uniendo para formar nuevos grupos, representando cada dato en el eje x y en el eje y el número de iteraciones que lleva el algoritmo (cuanto mayor altura, más iteraciones ha realizado). Por tanto, realizando cortes horizontales en determinados valores de y, podremos ver como se encuentran repartidos los datos en ese momento. Dónde cortar este dendrograma es una cuestión equivalente al conocimiento del número de centroides en el *k-means*, y si no se tienen conocimientos previos sobre los datos se recurren a métodos de *cluster validity* que se tratrán en la siguiente sección. En casos sencillos, el propio dendrograma puede aportar información sobre el número de *clusters*, observando cuando el algoritmo empieza a juntar grupos alejados entre sí, como se puede ver en la Figura 2.3.

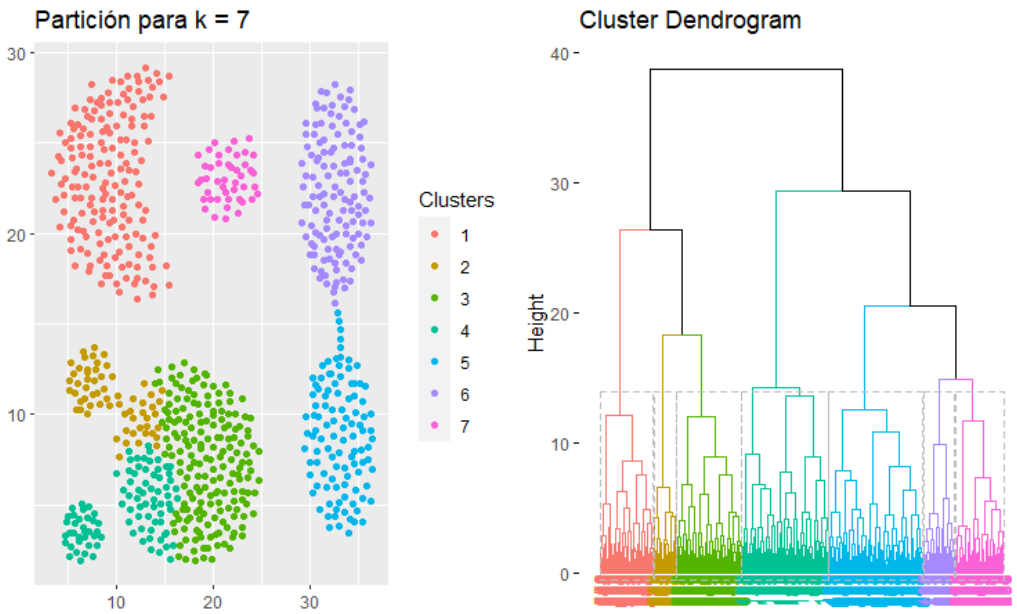


Figura 2.3: Partición de una muestra por el método *complete linkage* de unos datos artificiales, junto a su dendrograma y la clasificación real de los datos.

Una ventaja evidente frente al *k-means* es que la partición inicial siempre es la misma, cada elemento es un *cluster* en la primera iteración. Esto hace que al repetir el algoritmo en los mismos datos, el resultado sea idéntico, cosa que no se asegura en el *k-means*. También son bastante superiores a la hora de tratar datos no isotrópicos, algo en lo que los algoritmos particionales tienen muchos problemas. Una de las razones por las que trabajan mejor en este tipo de datos es por la mayor flexibilidad a la hora de seleccionar la distancia. Mientras que anteriormente se necesitaba recurrir a la distancia euclídea para asegurar la convergencia del método, ahora en los jerárquicos, salvo que se usen distancias que recurren a centroides o elementos definidos con distancia euclídea, pueden variar la métrica en función de qué datos se busca agrupar.

La desventaja del *clustering* jerárquico más clara es que tienen una complejidad temporal mayor, por lo que un conocimiento previo de los datos puede ser fundamental para determinar si es necesario recurrir a un *clustering* jerárquico. Por esto el *k-means* es más popular en la práctica, ya que los tamaños de datos suelen ser bastante considerables y con un gran número de variables. Esta cantidad de elementos a agrupar también puede dar problemas a la hora de la visualización, cosa que puede hacer el dendrograma

difícil de interpretar y reducir su utilidad.

En este tipo de *clustering* se trabaja con la matriz de distancias, que se construye computando la distancia entre cada par de elementos. Por tanto, la manera de medir la similitud o distancia entre datos es una característica importante, definiendo las tendencias de forma de los *clusters* y los tipos de datos en los que van a ser más o menos efectivos. Aunque existen numerosas distancias válidas (distancia entre centroides, medias, modas, etc), las más comunes y utilizadas son las *single-link* y *complete-link*.

En el *single-link*, la distancia entre dos *clusters* se define como

$$D_{ij}^{s} := \min_{x \in C_i, y \in C_j} d(x, y)$$

siendo d la métrica usada en el problema a tratar. En el *complete-link*, la distancia que se usa es comparando los valores más lejanos:

$$D_{ij}^{c} := \max_{x \in C_i, y \in C_j} d(x, y)$$

El *single-link* es en general mucho más versátil, por ejemplo, puede detectar *clusters* concéntricos relativamente cercanos. Aun así, hay un problema que puede existir en el *single-link*, y es conocido como el efecto de encadenamiento (*chain effect*): el algoritmo es bastante sensible ruido en los datos, pudiendo hacer que dos *clusters* cercanos se fusionen aunque no sean un único grupo (Figura 2.4). Esto da lugar a *clusters* alargados, que son encadenamientos de clusters más pequeños y le da el nombre a este efecto. Por otra parte, el *complete-link* da *clusters* muy compactos y no sufre tanto con el ruido muestral. Basándose en evidencia experimental, se observa que aunque el *single-link* da mayor versatilidad, el *complete-link* da mejores resultados en un mayor número de casos. En el ejemplo expuesto en 2.4, los datos están agrupados de forma compacta en su *cluster* romboidal y sin ruido, pero con las puntas de los rombos muy pegadas entre ellas. Este acercamiento es el que produce que el *single link* enlace grupos diferentes y no de una partición satisfactoria.

Por tanto el algoritmo sobre una muestra de datos X de tamaño n con una cierta métrica d seguiría el siguiente esquema:

1. Dividir X en n *clusters* de la forma $C_i = \{x_i\}$

2. Calcular la matriz de distancias entre *clusters*
3. Unir los dos *clusters* con la distancia mínima
4. Reiterar desde el paso 2 hasta que solo quede un único *cluster*
5. Representar en un dendrograma el proceso de agrupación y con algún criterio de validación o conocimiento previo de los datos decidir un número de *clusters* donde cortar el dendrograma

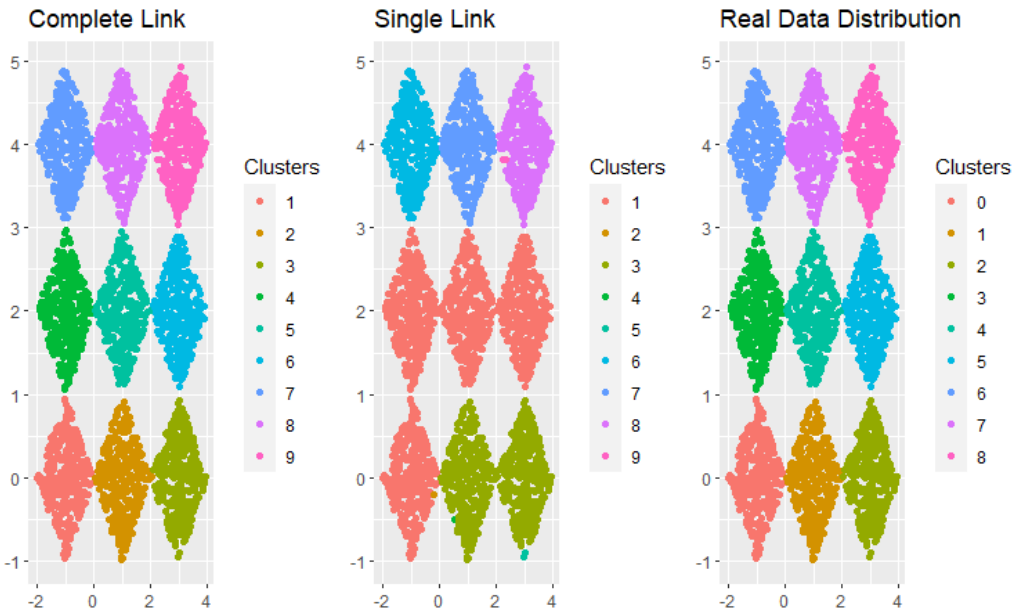


Figura 2.4: *Clustering* jerárquico de una muestra artificial en el que se observa como el *complete linkage* obtiene unos resultados fieles a la partición real, mientras que el *single linkage* tiende a juntar *clusters* por el *chaining effect*.

2.2.3. Métodos de densidad

DBSCAN

Hasta ahora todos los algoritmos vistos necesitaban una aproximación o un conocimiento previo de la forma de los *clusters*. Además, también eran muy sensibles al ruido o *outliers*, esto se debe a que los algoritmos vistos

hasta ahora son paramétricos, es decir, suponen una cierta distribución de los datos y con ella crean una función de verosimilitud que optimizar (por ejemplo, en el caso del *k-means* el parámetro de la distribución sería el vector de centroides). Una manera de evitar esta dependencia de un parámetro y conocer la forma de los *clusters* es utilizar un enfoque de densidad para el *clustering*. Esto quiere decir que los grupos se formarán en función a la densidad de datos en cada región del espacio, creando *clusters* en las zonas más densas y dejando los *outliers* fuera.

El *DBSCAN* no requiere de un número de grupos, pero necesita dos parámetros clave: ϵ y $MinPts$. El parámetro ϵ indica el tamaño del entorno que se tendrá en cuenta para estimar la densidad en cada punto y el parámetro $MinPts$ será el mínimo de datos que debe encontrarse en una ϵ-bola para que se considere densa. Ahora con esto en mente se dividen los datos.

Definición 2.6. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos,y dados unos parámetros $\epsilon > 0$ y $MinPts \in \mathbb{N}$, un $x \in X$ se dice que es* core *si su entorno de radio ϵ, $N_\epsilon(x)$, contiene más elementos que $MinPts$, es decir, $|N_\epsilon(x)| \geq MinPts$.*

Definición 2.7. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos,y dados unos parámetros $\epsilon > 0$ y $MinPts \in \mathbb{N}$, un $x \in X$ se dice que es* border *si $x \in N_\epsilon(y)$ con y core, pero $|N_\epsilon(x)| < MinPts$.*

Nota: Si un punto no es ni *core* ni *border* se le llama *outlier*.

Se ve que los *core* son puntos en zonas de alta densidad y serán los candidatos a comenzar *clusters*, mientras que los *border* serán puntos frontera de las regiones de alta densidad, además, los *outliers* serán el ruido de la muestra y se pueden desechar. Esta buena detección de ruido hace que en algunas ocasiones se utilice el algoritmo para eliminar estos *outliers* y después realizar algún otro tipo de *clustering*. En la Figura 2.5 se puede observar como al quitar el ruido de una muestra se puede mejorar la partición. Aunque sigue teniendo problemas para detectar el pequeño grupo que forma una línea entre los dos *clusters* superiores, se observa como los clusters alejados y bien diferenciados no tienen elementos de otros clusters y se mejora considerablemente la partición. Por consiguiente, para una muestra $X \subset \mathbb{R}^p$ con n datos y unos parámetros fijos de ϵ y $MinPts$, el algoritmo funcionaría así.

1. Seleccionar un $x \in X$ aleatorio, comprobar si *core*, volver a seleccionar

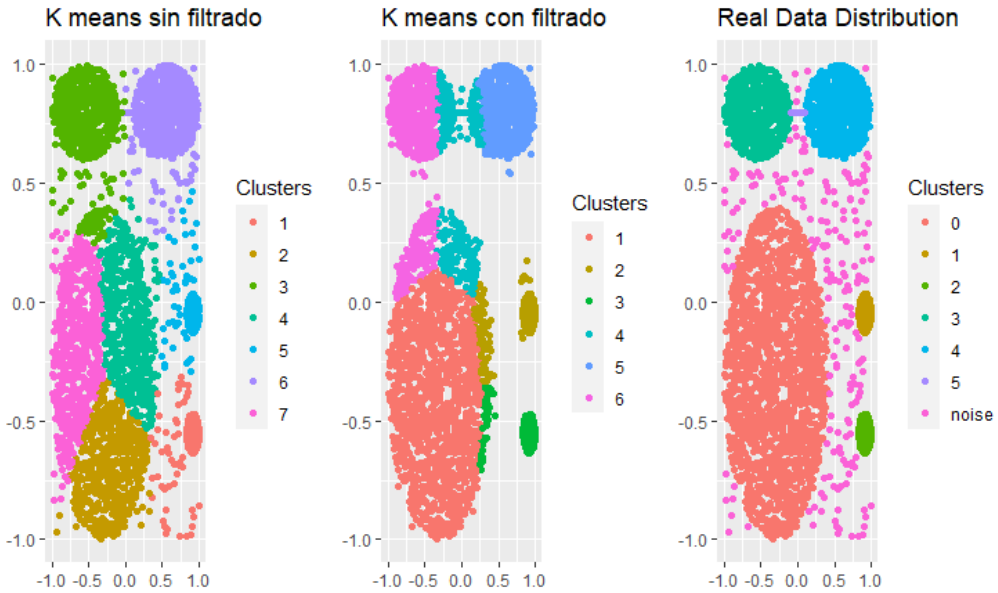


Figura 2.5: Las fiuras de izquierda a derecha muestran una partición de unos datos artificiales por *k-means*, luego una partición de *k-means* pero con un filtrado previo de los *outliers* con DBSCAN y finalmente la partición real

elementos hasta encontrar uno.

2. Se crea un *cluster* con x y todos los datos dentro de $N_\epsilon(x)$.
3. Se comprueba si hay más *cores* en el *cluster*. Si hay, añadir su $\epsilon-$entorno al *cluster* y reiterar hasta que no se encuentren nuevos *cores.*
4. Volver al paso 1 y reiterar hasta que no se encuentren más *cores.*

Como se observa en la Figura 2.6, este método consigue éxito en datos donde se observó en la Figura 2.1 que el *k-means* tenia problemas. El DBSCAN es una herramienta de detección que con pocas hipótesis sobre la muestra es capaz de ver formas muy complejas fijándose solo en zonas de alta densidad, y además, no es necesario proporcionar un número previo de *clusters*. A costa de esto, ahora se tiene que dar el valor de 2 parámetros, ϵ y $MinPts$, que determinarán la partición final y cuya estimación es igual o más problemática que el número de *clusters* en muchos casos. La interpretación de estos dos parámetros es la cantidad de datos en un cierto aréa necesarios para considerar ese area "densa", por lo que al final no necesitar un número de *clusters* previo resulta en la necesidad de otro parámetro equivalente.

Con esto se quiere remarcar que los algoritmos vistos hasta ahora siempre necesitan un criterio previo para realizar su partición y sin ese parámetro no pueden detectar *clusters*. En este caso, lo que se le dice al algoritmo es la densidad mínima para considerar una zona como una agrupación.

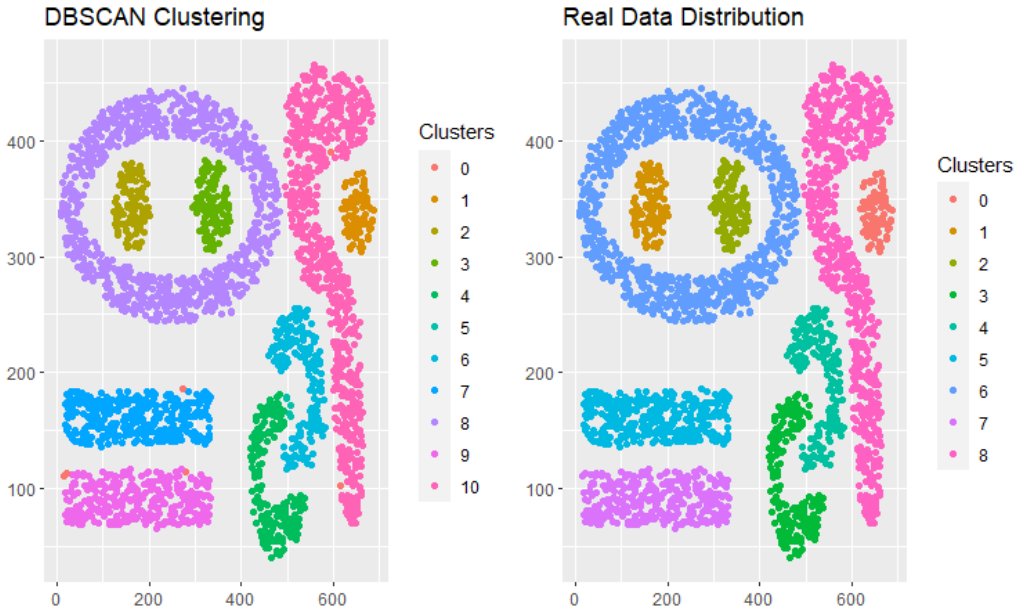


Figura 2.6: Partición obtenida usando el DBSCAN con unos valores de $\epsilon = 12$ y $MinPts = 4$ (Izquierda). Partición real de los datos (Derecha)

La estimación previa de estos parámetros acarrea otro inconveniente, y es que solo detecta *clusters* de una densidad específica, es decir, en una muestra con agrupaciones de densidades variables *DBSCAN* tendría problemas para detectar todos los *clusters*. Aun con todos estos inconvenientes, este enfoque de densidades es muy potente y por tanto se han realizado numerosas modificaciones a este método para tratar de aliviar sus limitaciones. Algunos algoritmos populares son por ejemplo $ADBSCAN$ y $OPTICS$.

Adaptive-DBSCAN ($ADBSCAN$) soluciona el problema de las densidades variables de una forma muy simple. Se empieza en un cierto valor de ϵ aleatorio, se realiza una primera agrupación como en $DBSCAN$, y seguidamente se aumenta el ϵ y se vuelven a buscar *clusters*. De esta manera se barren varias densidades posibles, con el inconveniente de que hay que añadirle un nuevo criterio de parada, y para ello se le da al algoritmo un número de *clusters* en el que detenerse.

OPTICS

Ordering Points To Identify Cluster Structure, o OPTICS, es una modificación del algoritmo DBSCAN que trata de combinar enfoques jerárquicos junto con las técnicas de densidad para lograr eliminar una limitación importante del algoritmo mencionado en la sección anterior. El DBSCAN necesita como parámetros dos valores que le permitirán identificar que zonas son suficientemente densas para considerarse *clusters*, pero esto falla si los grupos son de denisidades variables. En la Figura 2.7 se ejemplifica perfectamente el problema, ya que se observa como DBSCAN no puede detectar los *clusters* pequeños de la esquina superior derecha a la vez que los otros dos. La razón es sencilla, si se toma un ϵ como en la gráfica de la izquierda se pueden detectar los dos clusters de la zona izquierda, sin embargo, el ϵ es demasiado grande y considera una misma zona la región donde se encuentran los 3 *clusters* pequeños, creando un único grupo para ellos. La solución sería reducir el ϵ para que pueda ver cada uno de estos grupos de forma separada y considerarlos clases diferentes, pero al hacer esto, como se ve en la gráfica central de la figura, los dos *clusters* que antes se veian con claridad ahora han perdido una gran cantidad de elementos. Al darle un ϵ menor, y por tanto un umbral de densidad mayor a DBSCAN, el algoritmo considera que no hay suficiente concentración de puntos en las zonas más externas de esos grupos, considerando los datos que antes estaban dentro de los *clusters* como ruido.

OPTICS hace uso de dos nuevos conceptos, motivados por la definición de *core*, que permiten caracterizar a los datos en función de la densidad relativa de su zona: la *core distance* y la *reachability distance*.

Definición 2.8. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos,y dados unos parámetros $\epsilon > 0$ y $MinPts \in \mathbb{N}$. Sea $x \in X$, se define la* core distance *de x, o CD(x), como:*

$$CD(x) = \begin{cases} \epsilon' & si\ |N_\epsilon(x)| > MinPts \\ Indefinido & si\ |N_\epsilon(x)| < MinPts \end{cases}$$

Siendo $N_\epsilon(x)$ el entorno del punto x con radio ϵ, y ϵ' el mínimo $\epsilon > 0$ tal que $|N'_\epsilon(x)| > MinPts$.

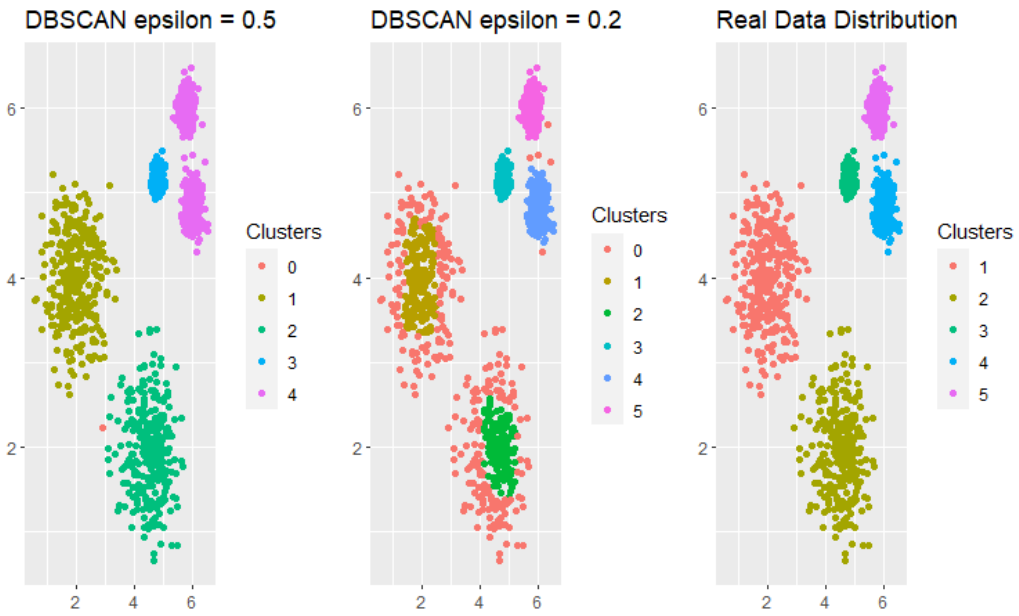


Figura 2.7: 5 distribuciones normales con varianzas diferentes. Se observa que la densidad para detectar los clusters de varianza grande no permite ver a los pequeños, mientras que un epsilon mas pequeño hace deshechar como ruido los datos externos de los *clusters* de mayor varianza.

Esta *core distance* es intuitivamente el menor valor de ϵ necesario para que el algoritmo reconozca al punto como *core*. Con esto ya se obtiene una característica adicional de cada dato que da información sobre la partición. Si un punto tiene una *core distance* muy grande relativa al resto, eso quiere decir que se encuentra muy alejado del resto de datos y por tanto será más probable que sea ruido. Con este valor, ahora se puede definir otro concepto más que actuará como un indicador adicional de las relaciones entre datos.

Definición 2.9. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos, d una métrica, y dados unos parámetros $\epsilon > 0$ y $MinPts \in \mathbb{N}$. Sean $x, y \in X$, se define la* reachability distance *de x respecto a y, o $RD_x(y)$ como:*

$$RD_x(y) = \begin{cases} \text{máx}\{CD(x), d(x,y)\} & si\ |N_\epsilon(x)| > MinPts \\ Indefinido & si\ |N_\epsilon(x)| < MinPts \end{cases}$$

Siendo $N_\epsilon(x)$ el entorno del punto x con radio ϵ.

Esta *reachability distance* es un concepto de distancia basándose en la *core distance* para determinar la cercania entre datos. Aunque no llega a ser una métrica propiamente dicha (es sencillo ver que no cumple la propiedad de simetría, ya que la distancia de un *core* a un outlier será la distancia entre ellos, mientras que la del *outlier* será la *core distance*, que no tiene porqué coincidir con la anterior), la *reachability distance* sigue siendo una herramienta muy útil en la que se apoya OPTICS para conseguir buenos resultados. La intuición detrás de este concepto es que los puntos que si dos datos pertenecen al mismo grupo, estarán cerca y además su RD será pequeña, mientras que el ruido estará en zonas de poca concentración de puntos y tendrá una RD mayor en general respecto al resto de elementos.

OPTICS pide un ϵ y un $MinPts$, aparentemente los mismos parámetros que DBSCAN. Sin embargo, en este caso el ϵ tiene una función distinta. En el DBSCAN, ϵ determinaba el tamaño de las zonas que barrería el algoritmo, pero aquí es solo una cota superior de la *reachability distance* con el único propósito de limitar el tiempo computacional y no hacer calcular al ordenador distancias demasiado grandes. En resumen, aunque OPTICS pida dos parámetros, uno de ellos es una cota para reducir el tiempo de ejecución y no afecta al proceso de agrupación. El algoritmo seguiría el siguiente esquema para una muestra de datos $X \subset \mathbb{R}^p$ y para un cierto ϵ y $MinPts$:

1. Seleccionar un punto $x \in X$ y obtener su CD y $\min\limits_{x \neq y \in X} RD_x(y)$. Si alguna de las dos no está definida o es mayor que ϵ, considerar el punto como *outlier* y seleccionar otro. En otro caso, etiquetar el punto x con la posición 1 y sus respectivos CD y $\min\limits_{x \neq y \in X} RD_x(y)$.

2. Excluir x del conjunto de datos y realizar el mismo proceso para el elemento y (como x está exlcuido tendrá otro punto que minimice su RD). Proceder como en el paso anterior, esta vez asignando la posición 2 al elemento y.

3. Repetir el paso 2 hasta recorrer todos los puntos.

Al igual que en los métodos jerárquicos, el resultado de OPTICS no es una partición concreta, si no algo que equivaldría al dendrograma, denominado *reachability plot*. En esta gráfica se colocan en el eje x los datos en el orden que los ha seleccionado el algoritmo y en el eje y su RD respecto al punto

que la minimiza. Se observará que las zonas con valores pequeños serán regiones de puntos muy cercanos, mientras que picos pueden determinar varias cosas: si son muy finos el paso de un cluster a otro, mientras que si son gruesos representan esta misma transición pero con ruido de por medio. Así, se puede hacer un estudio cualitativo de los valles de la gráfica para identificar los *clusters* y luego establecer u8n criterio en base a la *reachablity plot*. En la Figura 2.8 se observa como en los mismos datos que el DBSCAN tenía problemas, ahora es clara la existencia de 5 valles, además se puede ver que los dos primeros *clusters* que ha encontrado tienen una menor densidad ya que su RD mínima es mayor. Para obtener una partición de ahí, se puede seleccionar un corte en el eje y, que sería el equivalente de fijar un cierto ϵ y realizar un DBSCAN. Otra manera de hacerlo, en la que no se pierde la información sobre la diferencia de densidades que aporta el algoritmo es tras un análisis de los máximos, realizar cortes en el eje x para determinar los *clusters*, y luego fijar un RD máximo a partir del cual los datos serán considerados ruido.

En conclusión, el algoritmo ha introducido el concepto de *reachability distance* para lograr observar las diferencias de densidad entre *clusters* antes de realizar una partición, garantizando una mejor agrupación. Todas estas mejoras vienen con un coste, pero en este caso no demasiado grande. Hay una mayor complejidad temporal respecto al DBSCAN, pero no lo suficientemente significativa como para limitar el algoritmo a conjuntos de datos reducidos. El otro problema y actual objeto de estudio de varios grupos (Ankerst et al. [1], Patwary et al. [21]), es tratar de optimizar el algoritmo para datos de alta dimensionalidad, donde la *reachability plot* empieza a presentar problemas y es necesario recurrir a modificaciones en el algoritmo o a métodos sofisticados de visualización.

2.2.4. Métodos difusos

Fuzzy c-means (FCM)

El algoritmo difuso más popular y sencillo es una adaptación del conocido *k-means*, buscando esta vez clasificar una serie de datos X en k *clusters*. Al igual que en su versión exacta, se trabaja minimizando una función objetivo,

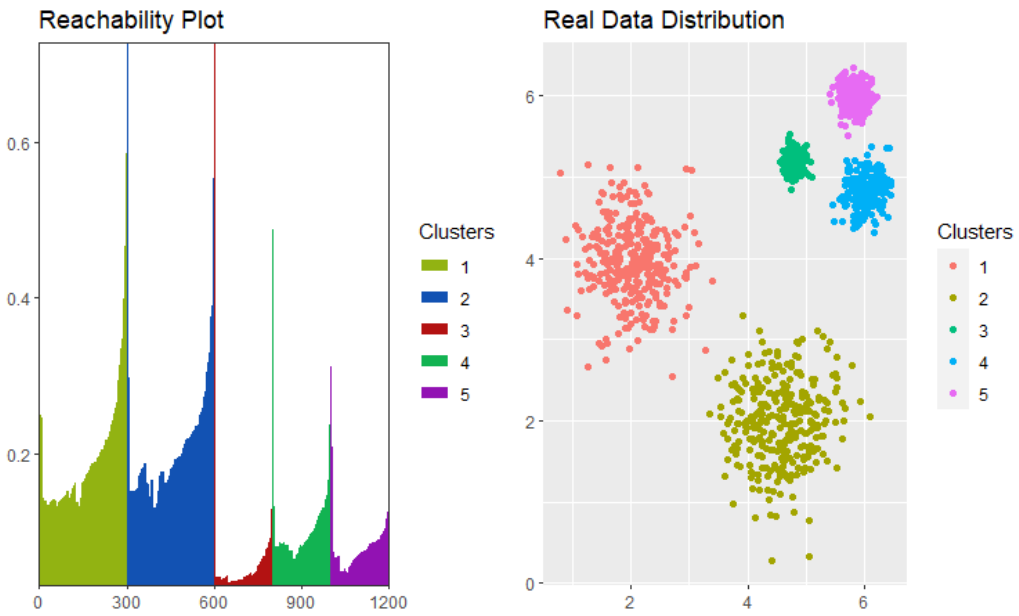


Figura 2.8: Resultados del algoritmo OPTICS para los datos de la derecha. En la *reachability plot* de la izquierda se han coloreado los puntos en función a su clase de origen para ejemplificar los valles y la efectividad de OPTICS.

que irá relacionada con las distancias al cuadrado. Sin embargo, al estar en un método difuso, todos los elementos pertenecen en cierto grado a todos los grupos. Esto ya indica que cada distancia entre centroides y *clusters* deberá ir ponderada en función del grado de pertenencia del elemento a ese *cluster*. Por tanto, se define la función objetivo de la siguiente forma:

Definición 2.10. *Sea X una muestra de n datos y $V = \{v_1, \ldots, v_c\}$ el vector de centroides asociados a una cierta partición en k* clusters*. Sea U la matriz de pesos, se define la función objetivo del FCM como:*

$$J(X; U, V) = \sum_{i=1}^{k} \sum_{j=1}^{n} (\mu_{ij})^m ||x_j - v_i||^2 \tag{2.3}$$

Este $m \in [1, \infty)$ recibe el nombre de *fuzzifier* o difusor, y determina la separabilidad de los *clusters* resultantes. Es decir, cuanto mayor sea m más difusa se volverá la partición, ya que incluso grados de pertenencia relativamente altos tenderán a 0. Por tanto, un conocimiento previo de la estructura de los datos puede ayudar a determinar este exponente, siendo buen criterio tomarlo cercano a 1 si los *clusters* están bien separados y sin muchos puntos

aislados, mientras que se tomará un m cercano a 6 si la partición es caótica y con muchos grupos intersecados (con $m = 6$ una pertenenecia de 0.8 pasaría a ser de 0.26, haciendo los *clusters* muy difusos). En general, si no se conoce nada de la partición a priori, se toma el valor $m = 2$.

La función J puede ser vista como la medida ponderada de la variación *intra-cluster* en función a los grados de pertenencia. Por tanto, el objetivo principal es alcanzar un mínimo, preferiblemente uno global. Una vez más, al usar centroides surge la necesidad de utilizar la norma euclídea, aunque se puede realizar el algoritmo con una norma más general, afectando a la forma de los *clusters*. A

A diferencia del *k-means*, en este caso la función a minimizar depende de los centroides y además de los pesos, haciendo algo más complejo obtener los valores óptimos para la siguiente iteración del algoritmo. Este es el motivo por el que se introdujeron ligaduras a la hora de definir U, ya que ahora se pueden obtener estos valores óptimos de (U, V) usando $\sum_{i=1}^{c} \mu_{i,j} = 1 \quad \forall j = 1, \ldots, n$ y el método de los multiplicadores de Lagrange para formar:

$$\bar{J}(X; U, V, \lambda) = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mu_{ij})^m ||x_j - v_i||^2 + \sum_{j=1}^{n} \lambda_k \left(\sum_{i=1}^{c} \mu_{ij} - 1 \right) \tag{2.4}$$

Al igualar los gradientes de $\bar{J}$ respecto a U, V, λ a 0, se obtienen las expresiones óptimas de la matriz de pesos y los centroides:

$$\mu_{ij} = \frac{1}{\sum_{l=1}^{k} \left(||x_j - v_i|| / ||x_j - v_l|| \right)^{2/(m-1)}} \qquad v_i = \frac{\sum_{j=1}^{n} (\mu_{ij})^m x_j}{\sum_{j=1}^{n} (\mu_{ij})^m}$$

De forma análoga al *k-means*, al ir de partición óptima en partición óptima, se garantiza que la función objetivo se irá acercando a un mínimo. Esto resulta en un algoritmo que mantiene la simpleza del *k-means*, dando una nueva perspectiva de los datos. Añadiendo un parámetro más a la función objetivo, la matriz de pesos U, obtenemos todos los grados de pertenencia de cada dato. Sin embargo, también va acompañado de los mismos inconvenientes: Es sensible a la partición inicial y además hay que fijar un número

previo de grupos. También se ve limitado al restringirse a la norma euclídea, ya que esta tiene una tendencia para detectar mejor *clusters* compactos, circulares y sin demasiados puntos aislados. Se ha trabajado en varias formas de corregir este último problema, algunas usando distancias adaptativas y otras tratando de buscar una alternativa para la función objetivo.

Algoritmo Gustafson-Kessel (GK)

Gustason-Kessel es una adaptación del algoritmo anterior para detectar distintas formas de *clusters*. Se deja de utilizar la norma euclídea, pasando a ser la matriz de distancias una variables de la función a minimizar, de forma que pasa a tener la siguiente expresión:

$$J(X;U,V) = \sum_{i=1}^{k}\sum_{j=1}^{n}(\mu_{ij})^m(x_j - v_i)'A_i(x_j - v_i)$$

siendo U la matriz de pesos, V el vector de centroides y A la k-tupla de matrices semi definidas positivas que inducen la norma para cada *cluster*. La dependencia lineal entre las A_i y la función objetivo plantean un inconveniente a la hora de hallar el mínimo: si se hacen las A_i menos semi definidas positivas, se puede reducir de forma arbitraria el valor de $J(X;U,V,A)$, por lo que es necesario añadir alguna restricción a estas matrices. La solución es imponer que aunque la forma de los *cluster* cambie en cada iteración, se mantiene su volumen constante. Esto equivale a dejar el determinante de cada A_i invariante:

$$det(A_i) = \rho_i \quad \rho_i > 0 \qquad \rho = (\rho_1, \ldots, \rho_c)$$

Con esto se puede volver a realizar el método de Lagrange, dando una expresión análoga de los centroides y los grados de pertenencia además de una expresión del A óptimo. Para una notación más clara a partir de ahora se usará $D^2_{i,j,A}$ para referirse a $(x_j - v_i)A_i(x_j - v_i)$. Además, es conveniente introducir F_i, de matriz de covarianza difusa del *cluster* i-ésimo, que se define

de la forma:

$$F_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m (x_j - v_i)(x_j - v_i)'}{\sum_{j=1}^{N} (\mu_{ij})^m} \tag{2.5}$$

Con esto obtenemos una expresión de los parámetros que optimizan la función objetivo:

$$\mu_{ij} = \frac{1}{\sum_{l=1}^{k} \left(D_{i,j,A_i}(x_j, v_i) / D_{l,j,A_j}(x_j, v_l) \right)^{2/(m-1)}} \tag{2.6}$$

$$v_i = \frac{\sum_{j=1}^{n} (\mu_{ij})^m x_j}{\sum_{j=1}^{n} (\mu_{ij})^m} \tag{2.7}$$

$$A_i = [\rho_i det(F_i)]^{1/n} F_i^{-1} \tag{2.8}$$

Con esta adaptación, el algoritmo procedería de la siguiente manera:

1. Inicializar la matriz U de forma aleatoria, siempre manteniendo las restricciones para que sea una matriz de pesos
2. Calcular los v_i con 2.7
3. Obtener las F_i de cada *cluster* y las A_i con 2.8
4. Calcular las $D^2_{i,j,A}$ y usarlas para obtener los pesos optimizados μ_{ij} de acuerdo a 2.6
5. Repetir hasta que se llegue a uno de los criterios de parada (se usan los mismos que en el *k-means*)

Aunque sigue siendo necesario que la densidad de los grupos sea los más constante posible, GK logra mejorar la clasificación cuando las formas de los *clusters* pierden la tendencia circular y compacta, como se observa en la Figura 2.9. En las gráficas se ve como el algoritmo *k-means* difuso tiene problemas con los extremos de los *clusters* rectangulares, mientras que GK reduce mucho el error a la hora de clasificar.

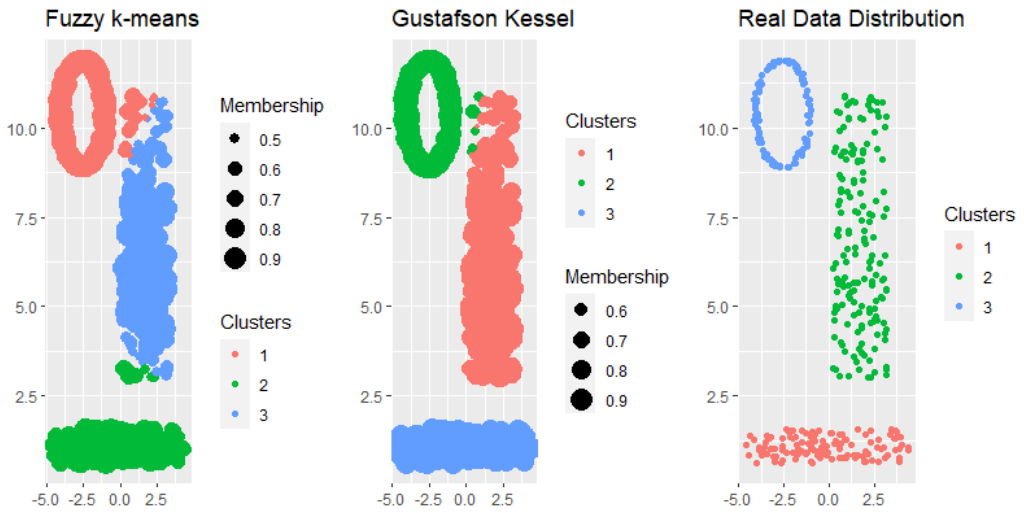


Figura 2.9: Partición de una muestra artificial usando FCM y GK, comparados con la clasificación real. Se observa como GK comete menos errores de clasificación y además en general da grados de pertenencia mayores, representado por el tamaño de los puntos.

2.3. Validación de Cluster

Hasta el momento, se ha estudiado una variedad de algoritmos con enfoques y procedimientos muy diversos, pero todos tenían algo en común: parámetros de partida. Todos los algoritmos funcionan en base a algún tipo de modelo o asunción previa, que es lo que permite realizar una distinción entre datos. Estas suposiciones previas realizadas por el modelo pueden ser la forma de los *clusters*, como en el *k-means*, o la densidad de los mismos como en el DBSCAN. Además, en todos los algoritmos no jerárquicos se requiere de un cierto parámetro inicial (en general el número de *clusters*, menos en los algoritmos de densidad que piden ϵ y *MinPts*) que determinará la partición. En todas las figuras y ejemplos mostrados anteriormente, las muestras eran artificiales y se sabía a que grupo pertenece cada dato y la cantidad de clases existentes en la muestra. En la realidad, no siempre se tendrá acceso a estos parámetros y es preciso buscar métodos para estimarlos.

Estos inconvenientes hacen surgir el estudio de la valided de las particiones y del cálculo del número óptimo de *clusters*: el *Cluster Validation*. Con el uso de distintas herramientas matemáticas se construirán diferentes índices que

indicarán cuál es el número de *clusters* óptimo para una cierta partición. A la hora de aplicarlos, entender su funcinamiento y los elementos que utilizan es fundamental, ya que una vez más se debe ser consistente con las distancias. Si se planea hacer una partición con un algoritmo no euclídeo, utilizar un índice de validación que use elementos como centroides puede dar lugar a error. En general, se suelen contrastar varios índices y elegir el número de *clusters* más común entre ellos, pero a veces tener un conocimiento previo del problema y un entendimiento de las distintas técnicas de validación permite limitar la cantidad de validaciones a realizar, eligiendo las técnicas que mejor se ajusten al problema.

2.3.1. CH Index

Este índice fue propuesto en 1974 por T. Caliński & J Harabasz [6]. Busca aprovechar propiedades intrínsecas de la muestra, comparando las distancias entre puntos de un mismo *cluster* con las distancias al resto de datos. El concepto de relacionar distancias a puntos del mismo grupo y las de puntos de otros grupos es una herramienta recurrente en muchos métodos de validación. En este caso, se utilizan conceptos de varianza y covarianza entre distintos puntos, y para ello se han de definir dos matrices.

Definición 2.11. *Sea una muestra de n datos $X \subset \mathbb{R}^p$, una partición $\{C_i\}_{i=1}^k$ con sus respectivos centroides $\{c_i\}_{i=1}^k$. Se definen la matriz de dispersión inter-cluster, W_k, y la matriz de dispersión intra-cluster, B_k, como:*

$$W_k = \sum_{j=1}^{k} \sum_{x_i \in C_j} (x_i - c_j)(x_i - c_j)^T \qquad B_k = \sum_{j=1}^{k} |C_k|(c_k - \bar{x})(x_k - \bar{x})^T$$

Dada una partición concreta, la matriz de dispersión *inter-cluster* determina la cercanía de los datos a su centro en cada *cluster*, mientras que la *intra-cluster* indica la separación relativa de los grupos formados. Con esto en mente, se ve que una partición con la métrica euclidea que debe ser bastante homogenea y sin ruido, con *clusters* bien diferencias, se debe corresponder con un W_k bajo y con un B_k alto.

Es importante notar que se está dando por conocida ya una partición

del problema, es decir, aunque no se use explícitamente, se está usando un algoritmo de *clustering* para dar esa partición y poder obtener los valores de B_k y W_k. En general, no hay mucha diferencia de un algoritmo a otro a la hora de dar el número de *clusters*, pero es importante tener en cuenta que estos índices darán resultados ligeramente diferentes dependiendo del método usado para obtener la partición inicial $\{C_i\}_{i=1}^{k}$.

El índice CH, o *Variance Ratio Criterion* (VRC), utiliza estas dos matrices de manera análoga al SCE y al SCF en el ANOVA, y construye un estadístico análogo al usado para el F-test para utilizar como índice.

$$CH(k) = \frac{\frac{tr(B_k)}{k-1}}{\frac{tr(W_k)}{n-k}}$$

Aunque se puede ver de donde viene la intuición (el estimador del ANOVA es $\frac{SCF/(I-1)}{SCE/(n-I)}$, que sigue una distribución $F_{I-1,n-I}$), hasta aquí llega el parecido; no hay una demostración probabilística rigurosa que relacione el índice CH con una distribución. En su lugar, se realiza una justificación más cualitativa, ya que una partición deseable se mencionó que debe tener W_k pequeño y B_k grande. Esto es equivalente a maximizar $CH(k)$. A diferencia de los métodos como el *k-means* que buscan optimizar una función y tienen problemas por no llegar al óptimo global, en el caso del $CH(k)$ es correcto parar en el primer máximo que se encuentre, ya que sea local o global, al ser el que da el menor número de grupos será el menos costoso computacionalmente de todos los máximos.

A continuación, en la Figura 2.10 se expone una muestra de datos usada anteriormente para ejemplificar el *clustering* jerárquico. En este caso, se ha utilizado el índice CH para determinar el número óptimo de *clusters* como 9, que coincide con el máximo de la función CH(k). Con este número se puede luego realizar una partición con algoritmos como el *k-means*.

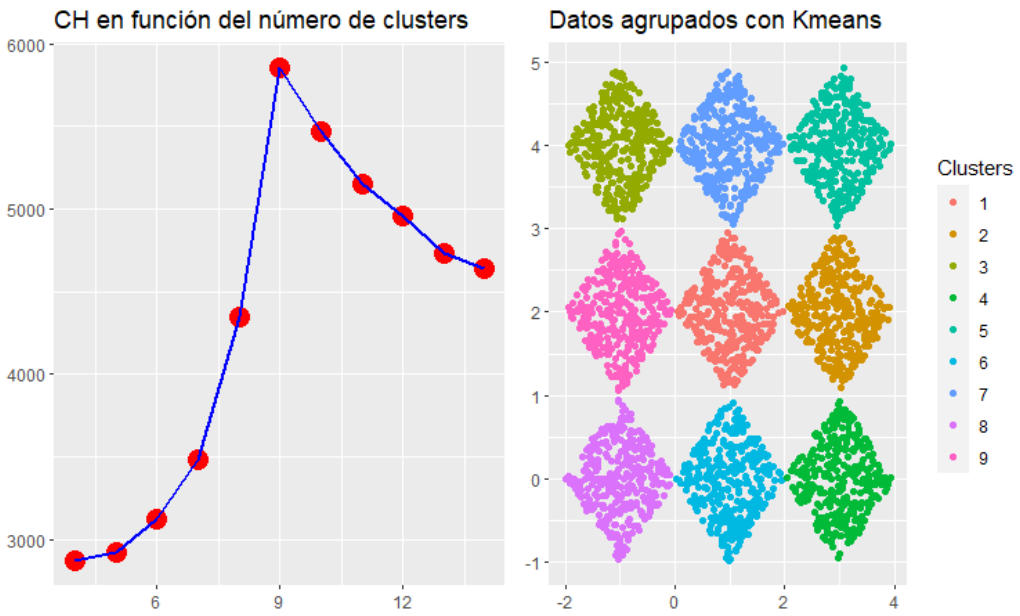


Figura 2.10: A la izquierda se observa la evolución de la función CH en función del número de clusters. A la derecha se muestran los datos usados para estos valores concretos de CH.

2.3.2. Silhouette Index

El índice Silhouette o Silueta, introducido por Rousseeuw [23], realiza un análisis elemento a elemento para dar una evaluación de la colocación del mismo y así logra establecer un criterio de validación para una partición. La idea detrás de esta métrica es dada una partición, comparar lo cerca que está cada elemento de su *cluster* con los elementos del *cluster* más cercano. Así, se definen los siguientes conceptos.

Definición 2.12. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos, d una distancia y sea $\{C_i\}_{i=1}^k$ una partición de la misma. Para un elemento $x \in C_r \subset X$, se definen la disimilaridad* intra cluster *e* inter cluster *como:*

$$a(x) = \frac{1}{|C_r|} \sum_{y \in C_r} d(x, y)$$

$$b(x) = \max_{r \neq i = 1, \ldots, k} \frac{1}{|C_i|} \sum_{y \in C_i} d(y, x)$$

Una vez más, se necesita de una partición previa, y por tanto, el valor del índice dependerá del algoritmo utilizado para obtenerla. El valor $a(i)$ da la distancia promedio de un elemento al resto de su clase, mientras que $b(i)$ indica la distancia promedio al segundo mejor *cluster* para el elemento, es decir, el que sin ser su *cluster* está más cerca del elemento. Estos conceptos ya indican como debe verse un elemento bien clasificado: un $a(x)$ muy pequeño en relación al $b(x)$, ya que esto quiere decir que los elementos de un grupo están todos muy cerca y diferenciados del resto. Para que esto funcione, es necesario asumir que las variables con las que se está trabajando están en una *ratio scale*, una de los 4 tipos de variables que constituyen la primera clasificación de los tipo de variables elaborada en 1946 por Steven [26]. Esto exige variables positivas en las que se puedan realizar operaciones aritméticas básicas y además se puedan establecer relaciones de proporción entre dos medidas. No son unas exigencias muy altas, ya que con métricas como la euclídea y medidas o transformaciones adecuadas la gran mayoría de problemas pueden expresarse así, por lo que no se entrará más en detalle en el tema de la escala ya que es un área extensa y fuera del objeto de este trabajo (para un resumen del estado actual de las escalas en un contexto matemático [20], [19]).

Definición 2.13. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos, d una distancia y sea $\{C_i\}_{i=1}^k$ una partición de la misma. Para un elemento $x \in C_r \subset X$, se define el índice de silueta del elemento x como:*

$$s(x) = \begin{cases} 1 - \frac{a(x)}{b(x)} & si\ a(x) < b(x) \\ 0 & si\ a(x) = b(x) \\ \frac{b(x)}{a(x)} - 1 & si\ a(x) > b(x) \end{cases}$$

o de forma equivalente:

$$s(x) = \frac{b(x) - a(x)}{\text{máx}\{a(x), b(x)\}}$$

El rango del índice de silueta es $-1 < s(x) < 1$, por eso se ha definido el caso más "neutro" en el que $a(x) = b(x)$ como el 0. Para entender el índice se puede pensar en situaciones extremas y estudiar el comportamiento de s. En el caso de un elemento x en un *cluster* compacto y bien diferenciado del resto, lo que ocurre es que $a(x) << b(x)$, es decir, $s(x) = \frac{b(x)-a(x)}{b(x)} \approx \frac{b(x)}{b(x)} = 1$. Mientras que en el caso opuesto, un elemento x está mucho más lejos de

su grupo que de elementos de otras agrupaciones ($b(x) << a(x)$), el índice toma el siguiente valor $s(x) = \frac{b(x)-a(x)}{a(x)} \approx \frac{-a(x)}{a(x)} = -1$. Por tanto, los ideal es tener valores de s lo más cercanos a 1 posible, siendo cercanos a 0 neutros (casos en los que no se tiene muy claro donde pertenece el elemento) y valores negativos reflejan una mala agrupación del elemento.

Para pasar a conclusiones globales de la partición, basta con asignar a cada *cluster* la suma de los índices de silueta de sus elementos, y tomar la media de las mismas para dar un índice asociado a una partición concreta.

Definición 2.14. *Sea $X \subset \mathbb{R}^p$ una muestra de n datos, d una métrica y sea $\{C_i\}_{i=1}^k$ una partición de la misma. Para un elemento $x \in C_r \subset X$, se define el índice de silhueta asignado a la partición $\{C_i\}_{i=1}^k$ como:*

$$Silh(k) = \frac{1}{n}\sum_{i=1}^{k} S(i) \qquad con\ S(i) = \sum_{x \in C_i} s(x)$$

Esto permite seleccionar un número óptimo de *clusters* buscando el valor máximo de esta función $Silh(k)$. En la Figura 2.11, se realiza un estudio sobre unos datos generados en 5 clases diferentes, donde el número óptimo de *clusters* es difícil de ver debido a la densidad variable, y a que una de las agrupaciones está muy dispersa. Al observar la gráfica, se ve como aunque el máximo está en 5 ($Silh(5) = 0.6185$), el valor para 4 agrupaciones está muy próximo ($Silh(4) = 0.6177$). Con esto se podría hacer un estudio con los dos posibles números de *cluster* y luego quedarse con el resultado que más se ajuste a algún conocimiento previo de la muestra. Como las particiones que dan un número más cercano a 1 son las que tienen *clusters* diferenciados y separados, este índice es sensible al ruido y también pueden ocasionar problemas muestras con formas complejas debido a la escala de ratio necesaria.

Una de las mayores ventajas de este enfoque es que gracias a su construcción, permite un análisis específico de la partición, es decir, permite identificar las debilidades de cada agrupación. Aunque en conjuntos relativamente grandes es prácticamente imposible analizar elemento a elemento, un estudio de los $S(i)$ permite identificar las agrupaciones más fiables (mayor $S(i)$) y cuales son más problemáticas.

Utilizando una vez más el ejemplo de la Figura 2.11, se pueden obtener

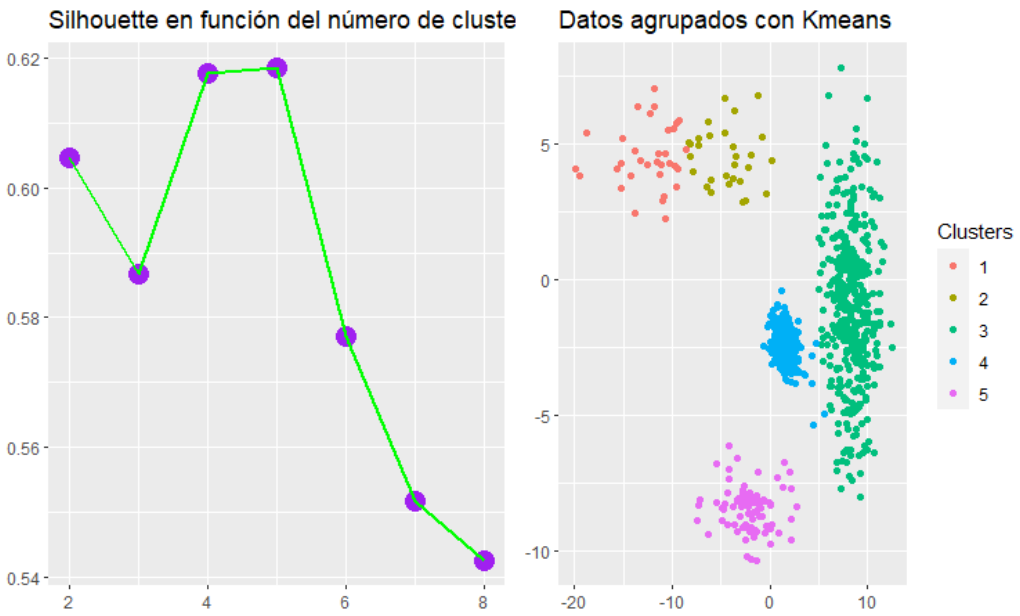


Figura 2.11: A la izquierda se observan distintos valores del índice de silueta asociados a un número distinto de *clusters*. A la derecha se muestran los datos usados coloreados con la partición dada por el algoritmo *k-means* con el número de clases que maximiza $Silh(k)$.

los datos de silueta de los 5 *clusters* que da el algoritmo:

$$(16.99591, 14.28917, 203.10059, 246.03006, 53.39111).$$

Cada posición del vector anterior corresponde con el *cluster* asociado a su respectivo número en la imagen, y por tanto tiene sentido que los *clusters* 1 y 2 muestran índices relativamente bajos, siendo la zona donde ha fallado el algoritmo en detectar un único *cluster*. También se ve como agrupaciones más compactas obtienen grandes valores de silueta, siendo los índices de 3 y 4 sumamente mayores que el de 5, aun estando muy pegados y por tanto teniendo una b relativamente baja.

2.3.3. Rand Index

Las dos métricas vistas hasta ahora se usan para un análisis exploratorio de los datos. Sin embargo, hay casos en los que se quiere contrastar el

clustering obtenido con una clasificación previa. Es en este contexto donde el Rand Index es de utilidad, propuesto en 1971 por William M. Rand [22] .

Para definirlo, se tienen en cuenta 2 suposiciones previas: cada elemento solo puede pertenecer a un *cluster* al mismo tiempo y todos tienen el mismo peso, es decir, el ruido repercutirá en esta métrica. La idea central es pensar en que hace a 2 particiones ser similares. De forma intuitiva se puede pensar en cuando un elemento pertenece al mismo grupo en ambas, pero esto da un problema sobre como identificar *clusters* entre particiones y limita la comparación a particiones con el mismo número de *clusters*. La solución es tratar con pares de elementos en vez de datos individuales. Sea $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^p$ una muestra de datos y $\mathcal{C} = \{C_1, \ldots, C_K\}$ y $\mathcal{V} = \{V_1, \ldots, V_r\}$ dos particiones de X, para $x_i, x_j \in X$ pueden ocurrir 4 situaciones:

(a) x_i y x_j están en el mismo grupo tanto en $\mathcal{C}$ como en $\mathcal{V}$

(b) x_i y x_j no están en el mismo grupo ni en $\mathcal{C}$ ni en $\mathcal{V}$

(c) x_i y x_j están en el mismo grupo en $\mathcal{C}$ pero distinto en $\mathcal{V}$

(d) x_i y x_j están en distinto grupo en $\mathcal{C}$ pero en el mismo en $\mathcal{V}$

Con estas consideraciones, si dos particiones son similares entre sí, se espera una mayor cantidad de casos (a) y (b), ya que de forma intuitiva la suma de ambos casos se pueden ver como el número de coincidencias entre las particiones. Ahora se puede definir el índice Rand.

Definición 2.15. *Sea $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^p$ una muestra de datos, $\mathcal{C} = \{C_1, \ldots, C_K\}$ y $\mathcal{V} = \{V_1, \ldots, V_r\}$ dos particiones de X. Se define el índice Rand como:*

$$Rand(X, \mathcal{C}, \mathcal{V}) = \frac{a+b}{\binom{N}{2}}$$

siendo a el número de veces que se da el caso (a) y b el caso (b) mencionados anteriormente.

Este número es la cantidad de coincidencias, dividido entre todos los posibles casos (es claro que $\binom{N}{2} = a + b + c + d$). Este número va de 0 a 1

y cuanto mayor es, mayor coincidencia hay entre particiones. Se puede ver este número como la probabilidad de encontrar una coincidencia entre las dos particiones.

El índice Rand puede dar resultados buenos, pero tiene un problema: no tiene ninguna correción de azar. Esto quiere decir que no se está teniendo en cuenta si se considerará una etiquetación aleatoria de los datos, y por tanto puede dar valores altos cuando en realidad los grupos se han elegido aleatoriamente. Un ejemplo claro sería el siguiente:

Sea una muestra de datos X con 100 elementos divididos en 2 grupos: A y B. Se sabe que 10 de los datos están en el grupo A y 90 en el B. Ahora, se crea otra clasificación en la que los 10 elementos del grupo A son del B, y 10 elementos al azar del grupo B son clasificados como A. Al obtener el valor del índice Rand computacionalmente, se observa que toma un valor de 0.67. Es decir, incluso en el peor caso posible en el que ningún elemento de A ha sido correctamente identificado, el índice sigue dando valores buenos. Este era el peor caso posible, por lo que cualquier clasificación aleatoria de los datos daría valores mayores o iguales del Rand.

Por tanto, se observa como el índice Rand no siempre garantiza que valores altos se correspondan a similitudes entre particiones y además es sensible a la descompensación entre grupos. Para solucionar estos problemas y obtener medidas más precisas y conservadoras, se desarrolló el índice Rand ajustado o *Adjusted Rand Index*.

Adjusted Rand Index

Una vez vistos los problemas del índice Rand, se propusieron y estudiaron varias alternativas (Hubert & Arabie [11]). De entre ellas, se va a exponer el índice Rand ajustado, o *Adjusted Rand Index* (ARI). La idea de este índice es hacer una corrección al Rand para que pase a mostrar lo cerca que están las particiones de haber sido elegidas aleatoriamente.

Para esto se supone que las particiones vienen de distribuciones hipergeométricas (es la hipótesis de este índice), y a partir de ahí se calcula el

valor esperado del índice Rand. Se puede deducir (Brenan & Light [5]) que el índice Rand es una transformación lineal de $\sum_{i,j} \binom{n_{ij}}{2}$, siendo n_{ij} el número de elementos comunes entre el *cluster* i de una partición y el *cluster* j de la otra. Con esto, se puede obtener la esperanza del índice:

$$E[\sum_{ij} \binom{n_{ij}}{2}] = \sum_{i} \binom{n_i}{2} \sum_{j} \binom{n_j}{2} / \binom{n}{2} \tag{2.9}$$

Donde $\sum_i \binom{n_i}{2}$ representa la cantidad de coincidencias posibles de cada *cluster* i con todos los *clusters* de la otra partición, y $\binom{n}{2}$ es el número total de parejas.

Una vez obtenida la esperanza de esta expresión, es sencillo obtener la esperanza del índice Rand por ser transformación lineal de la misma. Con esto se puede reajustar el índice y definir el ARI.

Definición 2.16. *Sea* $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^p$ *una muestra de datos,* $\mathcal{C} = \{C_1, \ldots, C_K\}$ *y* $\mathcal{V} = \{V_1, \ldots, V_r\}$ *dos particiones de* X*. Se define el ARI como:*

$$ARI(X, \mathcal{C}, \mathcal{V}) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2} / \binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2}] - \sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2} / \binom{n}{2}}$$

Esta expresión no es más que el valor del índice menos su valor esperado bajo al hipótesis de distribuciones hipergeométricas a la hora de crear particiones, dividido entre el valor máximo del índice menos el valor esperado, es decir, se podría reescribir como:

$$ARI = \frac{Rand - E[Rand]}{max(Rand) - E[Rand]} \tag{2.10}$$

Con esto se obtiene un índice que puede tomar valores entre -1 y 1. Valores negativos no tienen una interpretación útil y van siempre asociados a particiones malas, y cuanto más cercano a 0 el ARI, más cercanas las particiones a asignaciones aleatorias de etiquetas. Con esto se ha logrado obtener un criterio de similitud entre particiones que tiene en cuenta la posibilidad de etiquetados aleatorios, y además es más conservador que el índice Rand.

Capítulo 3

Análisis de un conjunto de datos musical

Con el fin de aplicar los conceptos aprendidos a un problema real, en este capítulo se realiza el estudio de un conjunto de datos. Se tratan concretamente de datos musicales, que han sido obtenidos exclusivamente para este análisis. La muestra sobre la que se realizará el análisis ha sido creada gracias a los datos ofrecidos de forma pública por Spotify, un servicio de música digital que da acceso a millones de canciones. Los usuarios de Spotify pueden acceder de forma *online* al catálogo y escuchar las canciones en cualquier dispositivo, siendo posible también interactuar con ellas con acciones como seleccionar sus favoritas, seguir artistas o crear listas de reproducción. Entre los datos a los que Spotify da acceso, además de variables descriptivas sobre las canciones y artistas, se proporcionan etiquetas relativas al género de los artistas. El reconocimiento del género de los artistas es un concepto complejo y volátil, afectado por muchos agentes como por ejemplo las tendencias de producción vigentes o los factores culturales. En muchos casos es difícil acotar un género concreto para un artista, e incluso en ocasiones para una sola canción, que podría corresponderse con un género mejor que con otro en distintas partes de la pieza. Sobre la estimación de géneros que realiza Spotify podemos encontrar la siguiente información descrita en la plataforma:

"We define genres based on info from listener playlists (title, des-

cription, etc.) and our music curation teams. We don't use info from metadata or the playlist pitching tool in Spotify for Artists. We recognize genres constantly evolve and songs can cross different genres. The way we assign songs to genres may change over time, and we may add new genres too."

Es decir, se obtienen de forma estimatoria principalmente a partir de los datos generados por los usuarios de Spotify, observando cómo interactúan con distintas canciones y las agrupaciones que tienden a crear en las mismas en forma de listas de reproducción. Por tanto, al no representar estas etiquetas un valor único real absoluto, el fin de este análisis es estudiar si los algoritmos de clustering pueden obtener particiones de canciones que se correspondan con esta estimación de géneros.

Respecto a la aplicación de las técnicas previamente descritas, se hará uso de todos los algoritmos mencionados en el capítulo anterior, sobretodo el *k-means* y los métodos jerárquicos, ya que son los más sencillos y flexibles. Los métodos de densidad y difusos serán utilizados para situaciones más específicas cuando se crea que los resultados proporcionados por los otros dos algoritmos son insuficientes o poco claros. Las técnicas de evaluación descritas serán utilizas para determinar el número de grupos con el que se obtiene la división óptima y juzgar la bondad de los resultados.

3.1. Creación del conjunto de datos

El conjunto de datos se ha creado exclusivamente para este proyecto, usando la interfaz de programación de aplicaciones (*Application Programming Interface*), conocida como API de Spotify. Una API es conjunto de protocolos informáticos que permite a dos aplicaciones conectar. En este caso, la API de Spotify facilita la conexión entre la base de datos de Spotify y distintos lenguajes de programación. El lenguaje de programación utilizado ha sido R, mediante el entorno de programación RStudio, que tras conectar con Spotify mediante la API permite acceder a su base de datos pública y obtener información de canciones, artistas, álbumes... Se ha creado una playlist de 1000 canciones en Spotify para tener unos datos con los que trabajar.

Las canciones descargadas son elementos con 2 variables identificadoras: el `track.id` y el `artist.id`, que son códigos únicos para cada canción y artista respectivamente. Más concretamente, `track.id` es una variable no numérica que contiene caracteres o combinaciones de letras y números, conocida habitualmente como 'de tipo *string*', que guarda la id única de cada canción, y el *artist.id* es también una variable *string* con el identificador único asociado al artista. Utilizando el `track.id`, se pueden obtener una serie de variables que contienen características musicales de cada canción:

- ***Acústica*** (`acousticness`): Variable numérica entre 0 y 1 que mide la probabilidad de que una canción sea acústica. Valores cercanos a 1 implican una gran confianza en que la canción es acústica.
- ***Bailabilidad*** (`danceability`): Variable numérica entre 0 y 1 que mide lo bailable que es una canción. Se usa una combinación de tempo, ritmo, intensidad y regularidad. Valores cercanos a 1 indican que es una canción fácilmente bailable.
- ***Energía*** (`energy`): Variable numérica entre 0 y 1 que haciendo uso del rango dinámico, timbre y ruido percibido determina la energía de una canción. Canciones rápidas, ruidosas y animadas tendrán valores cercanos a 1 mientras que canciones tranquilas recibirán valores bajos.
- ***Instrumentalidad*** (`instrumentalness`): Variable numérica entre 0 y 1 que predice la cantidad de voz en una canción. Coros como .ºohz .ªah"son tratados como instrumentos. Valores por encima de 0.5 indican una alta confianza en que es instrumental.
- ***Clave*** (`key`): Variable entera de 0 a 11, que indica la tonalidad de una canción. El 0 corresponde al Do, 1 a Do #...
- ***Vivacidad*** (`liveliness`): Variable numérica entre 0 y 1 que detecta la presencia de una audiencia en la grabación. Valores mayores de 0.8 indican con gran confianza que la canción es en directo.
- ***Volumen*** (`loudness`): Variable numérica que contiene el nivel de intensidad auditiva promedio de la canción en decibelios. No tiene cotas exactas pero el rango común de valores suele estar entre -60 y 0 *db*.
- ***Escala*** (`mode`): Variable binaria que toma valor 0 cuando una canción está en escala menor y 1 cuando está en escala mayor.

- ***Habladuría*** (`speechiness`): Variable numérica entre 0 y 1 que mide la cantidad de palabras habladas de cada canción. Valores por encima de 0.66 indican alta probabilidad de ser un podcast y no una canción.
- ***Tempo*** (`tempo`): Variable numérica que mide el tempo promedio de una canción. El tempo indica la velocidad o el ritmo de la canción.
- ***Valencia*** (`valence`): Variable numérica entre 0 y 1 que muestra la positividad dentro de cada canción. Canciones con valores cercanos a 1 muestran tonos más felices y animados mientras que valores bajos representan canciones tristes o más agresivas.

Este es el conjunto de variables iniciales con el que se comienza el análisis. Respecto al `artist.id`, aunque esta última puede parecer innecesaria para el análisis, se necesita para extraer el género de cada canción. Esto ocurre porque, como se ha mencionado previamente, Spotify guarda los géneros por artista, no por canción, por tanto con el identificador del artista se accede al género o géneros del mismo. Tras obtener el género de cada artista, asociamos dicho género a todas las canciones de este artista y la variable `artist.id` es eliminada, quedando finalmente dos variables *string*: una con el género o géneros y la id del track, junto a las variables de análisis.

Antes de pasar al estudio en detalle del conjunto, es necesario realizar un arreglo final. Como cada canción normalmente tiene varios artistas, y cada artista a su vez varios géneros asociados, es necesario separar estos géneros para poder realizar el análisis. La mejor manera de hacer esto es crear un dato para cada par distinto de *track.id* y género, es decir, cada canción tendrá asociados tantos datos como géneros diferentes tenga asociados.

La lista de géneros obtenidos es los siguientes: *classical, edm, house, metal, permanent wave, pop, reggaeton, rock* y *trap*.

3.2. Análisis de las variables

Una vez obtenidos los datos, es necesario realizar un análisis previo de los mismos. Como los métodos de *clustering* vistos son para un análisis ex-

ploratorio, es muy importante entender la estructura del conjunto y realizar una limpieza previa. Como se mencionó en la introducción, este proceso de selección de variables puede llevar, como se verá más adelante, a tener que realizar modificaciones previas al *clustering*. Lo primero de lo que hay que preocuparse es de la distribución de la muestra: cómo se busca una clasificación en géneros a partir de unos datos proporcionados ayuda a la obtención de resultados más fiables. En la Figura 3.1, se ve que hay una cantidad mucho mayor de *rock* y *metal*, un desequilibro que que podría tener efecto en los resultados producidos por los algoritmos.

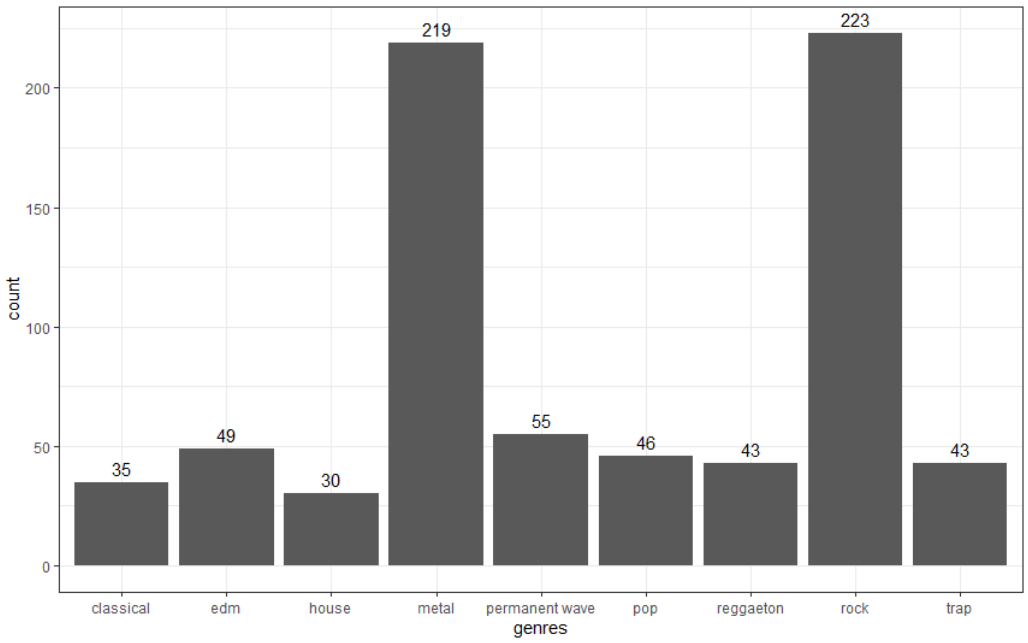


Figura 3.1: Distribución de las canciones en función del género en el conjunto de datos inicial.

Para ver el efecto de una mayor cantidad de canciones de *rock* y *metal*, se va a dividir el conjunto en 3 clases: *rock*, *metal* y *otro género*. Con esta subdivisión, se va a estudiar como se distribuyen las distintas variables segmentando por estos grupos. En la Figura 3.2, se visualiza el comportamiento de las variables `speechiness` vs. `danceability` y `acousticness` vs. `energy`. Se muestran los resultados en forma de un diagrama de dispersión, que coloca una variable en cada eje de coordenadas y establece una posición para cada dato, asignando a su vez un color según si el género es *rock*, *metal* u *otro*.

Lo que se observa en estas gráficas es que tanto el *rock* como el *metal*

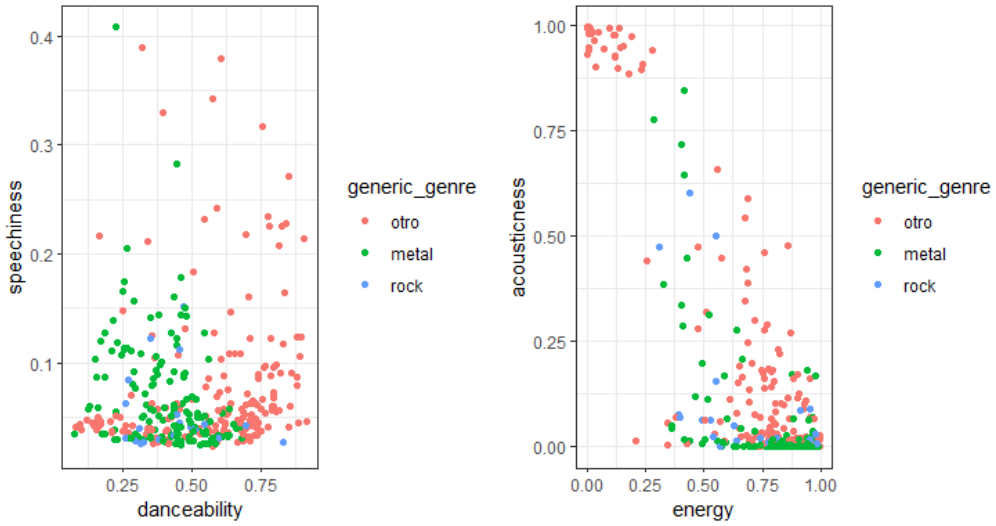


Figura 3.2: Diagramas de dispersión de los datos con subdivisión en colores según si son *rock*, *metal* u *otro género*. La gráfica de la izquierda representa `speechiness` vs. `danceability` y la de la derecha `acousticness` vs. `energy`.

parecen estar muy agrupados y con comportamientos similares, lo cual puede dificultar su separación en distintos grupos cuando se utilicen algoritmos. Además, se observa una gran concentración de datos tanto en zonas de baja `acousticness` como baja `speechiness`. Esto, de momento, no puede determinarse si es o no importante, pero será tenido en cuenta tras concluir con el análisis de géneros. Para entender mejor este comportamiento, se van a realizar diagramas de densidad en varias de las variables, subdividiendo los datos de nuevo en *rock*, *metal* u *otro género*. Estos diagramas colocan la variable a estudiar en el eje x, creando una función de densidad que a cada valor del rango de la variable le asocia un valor proporcional a la cantidad de elementos en esa zona. Como se está pasando de un conjunto discreto a una función continua, se construye una estimación. Los resultados se muestran la Figura 3.3. Se observa como tanto *rock* como *metal* tienen densidades casi idénticas, lo que hace sospechar que algo falla, ya que por muy similares que sean los dos géneros, no deberían tener un comportamiento extremadamente tan similar.

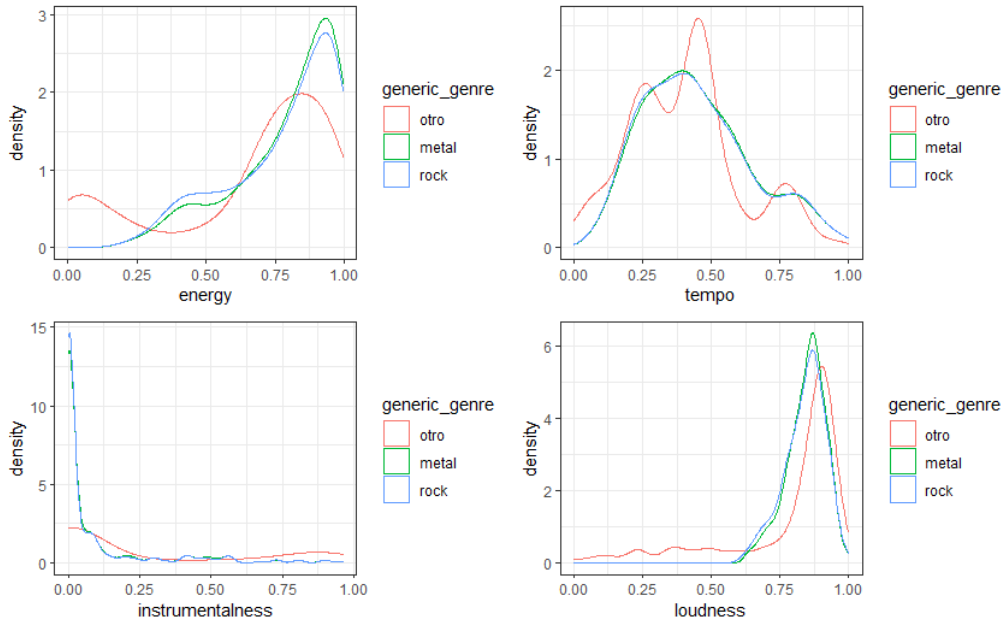


Figura 3.3: Densidad del conjunto de datos respecto a 4 variables, con una división por colores en función del género. De izquierda a derecha y de arriba abajo: energía, tempo, instrumentalidad y volumen.

Para comprobar lo que ocurre en *rock* y *metal* e identificar el porqué de este comportamiento tan similar, se procede a mirar la cantidad de canciones en el conjunto que están clasificadas con ambos géneros a la vez. Como se ve en la Figura 3.4, la gran mayoría de canciones de *rock* son también consideradas *metal*. Esto causa una gran sobre representación e imposibilita diferenciar ambos géneros.

La solución para este estudio es excluir el género *metal* y realizar el análisis con un género menos. La elección entre excluir *metal* o *rock* se ha realizado en función al conocimiento del analista sobre las canciones etiquetadas con estos géneros, ya que hay una mayor cantidad de canciones de grupos tradicionalmente asociados con el *rock*. Además, se va a tratar también el problema de la sobrerrepresentación del género *rock* respecto a los géneros restantes, tomando una muestra aleatoria de 50 canciones dentro del género rock y utilizando este subgrupo para las particiones. Con esto se consigue una partición bien distribuida, como se observa más adelante en la Figura 3.10, en la que se muestra el conjunto de datos final considerado para la aplicación de los algoritmos.

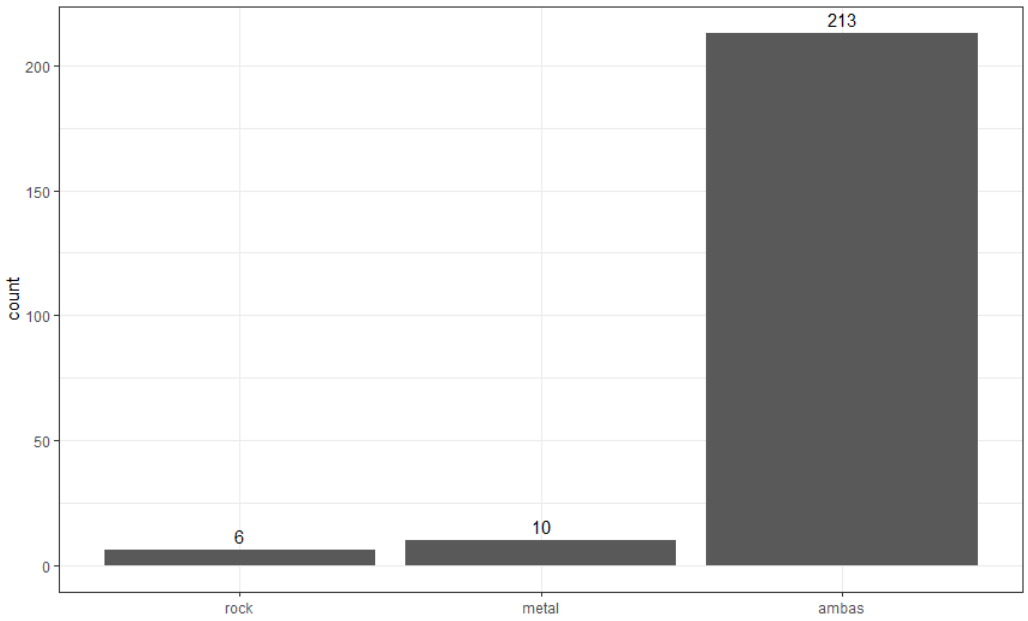


Figura 3.4: Número de canciones que son únicamente *rock*, únicamente *metal*, y ambos géneros a la vez.

Para ampliar el conocimiento del conjunto de datos y explorar en más detalle los géneros agrupados previamente como *otros*, se va realizar un estudio del comportamiento de ellos acorde a 4 variables. En la Figura 3.5 se pueden observar los resultados en forma de un diagrama de cajas. En general se observan cajas dispersas y con distintos comportamientos, especialmente la música clásica parece diferenciarse del resto en acústica y volumen. Sin embargo, parece que el *edm* y el *house* siguen un comportamiento demasiado similar. Realizando una vez el mismo análisis que con el *rock* y el *metal*, se descubre que todas las canciones *house* son además *edm*. Por tanto, se considera que no se cuenta con una muestra representativa del *house* y se elimina este género, quedando todas sus canciones asociadas solo al *edm*. También se ve que el *pop* y el *edm* tienen más del 90 % de canciones comunes, eliminando así un género más. De la misma manera, el género *reggaeton* y el *trap* están completamente solapados y por tanto se elimina el *reggaeton* y se analizará con el trap (una vez más, en base a las canciones en el conjunto).

Ahora, es necesario redefinir las variables `tempo` y `volumen` para que estén entre 0 y 1, ya que, al estar basados en distancias, una gran diferencia en escalas puede distorsionar como de cerca ve los datos un algoritmo. Para ello

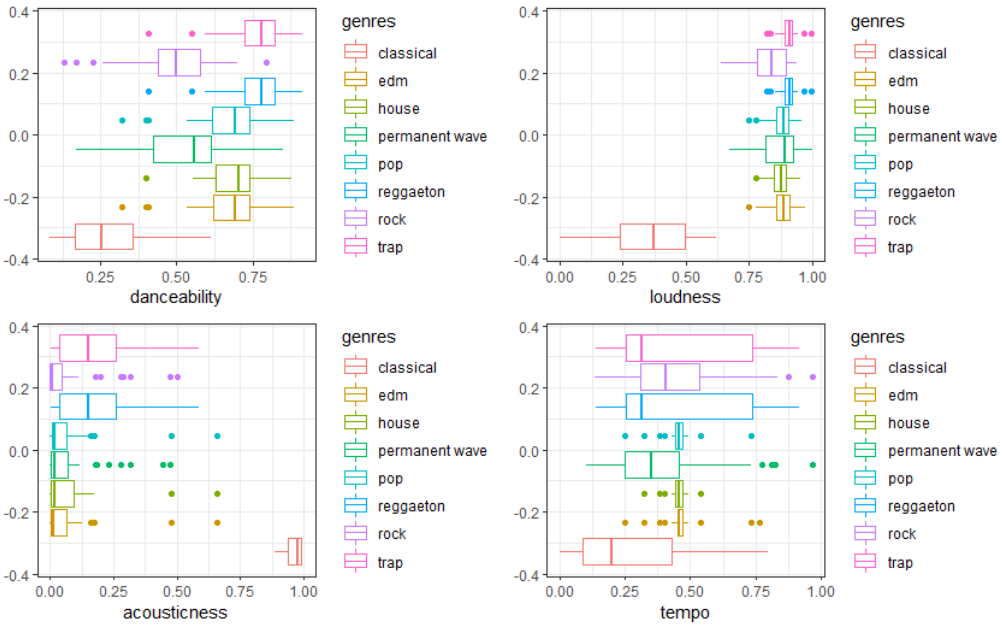


Figura 3.5: Diagrama de cajas subdividido en los distintos géneros. De izquierda a derecha y de arriba abajo: bailabilidad, volumen, acústica y tempo.

se definen los nuevos valores de cada variable de la forma:

$$tempo = \frac{tempo - max(tempo)}{max(tempo) - min(tempo)}$$

$$loudness = \frac{loudness - max(loudness)}{max(loudness) - min(loudness)}$$

Con esto, se tienen una serie de variables numéricas entre 0 y 1 que tienen comportamientos diversos y variados, como se muestra en el diagrama de densidades en la Figura 3.6.

Para concluir el análisis previo a los algoritmos, es necesario tratar con las dos variables que no son directamente numéricas: `key` y `mode`. Aunque no siguen el patrón del resto de variables a estudiar, encierran mucha información sobre las canciones. La variable `mode` el modo *mayor* o *menor* de la canción, lo cual está tradicionalmente asociado con que la canción suene más alegre (mayor) o más triste (menor). Por otro lado, la variable `key` indica la tonalidad utilizada en la composición, la cual se observa que también varía mucho entre géneros. La combinación de ambas variables es conocida en la

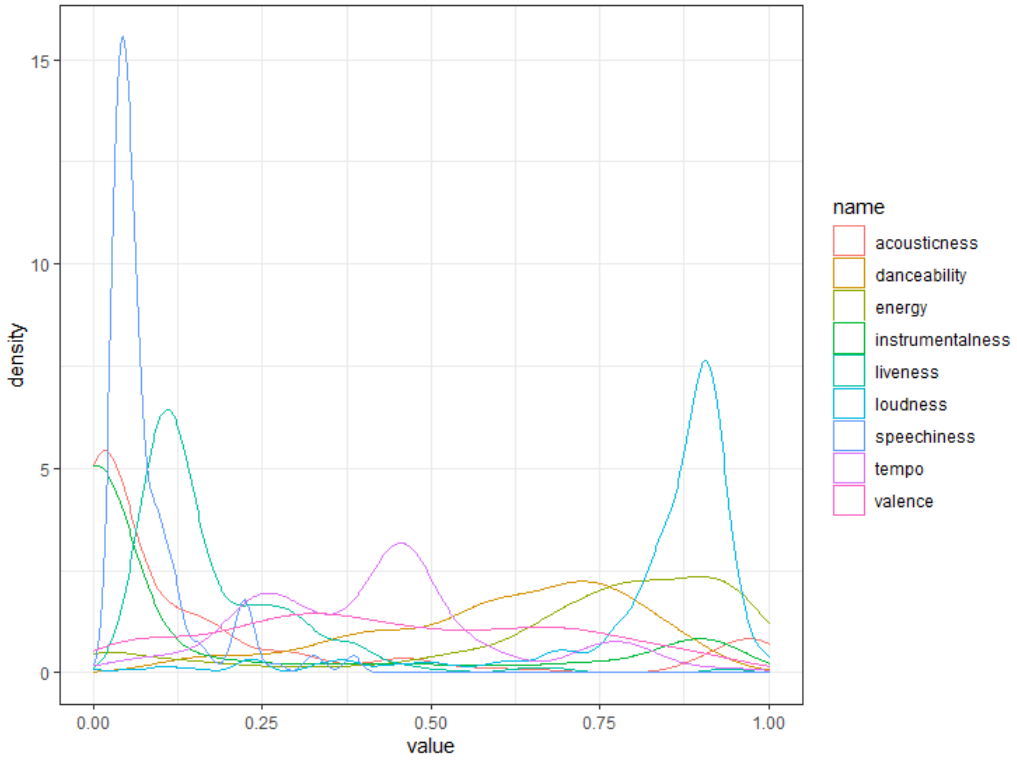


Figura 3.6: Distribución de los datos en todas las variables a utilizar para la clasificación.

teoría musical como *escala*. En primer lugar, se muestra en el Figura 3.7 cuántas canciones hay de cada escala y cómo se reparten entre mayor y menor. Se observa un comportamiento claramente no uniforme y con bastantes desequilibrios entre mayor y menor según el tono. También se ven unas escalas altamente pobladas mientras que otras cuentan con muy pocas canciones. Toda esta información puede ser significativa a la hora de establecer diferencias entre géneros y por tanto se va a buscar codificar la mayor cantidad de información en un formato que no distorsione la clasificación.

La manera de incorporar las variables `key` y `mode` es hacer uso de un concepto de teoría de la música conocido como *ciclo de quintas* para definir una nueva variable. El ciclo de quintas relaciona las escalas en función de las notas que tienen en común. Esto no solo permite establecer la noción de cómo de cerca están dos tonalidades, sino que permite relacionar las mayores con las menores. No se va a entrar en detalle sobre este concepto, pero se incluye en la Figura 3.8 una visualización del ciclo de quintas, y con él en

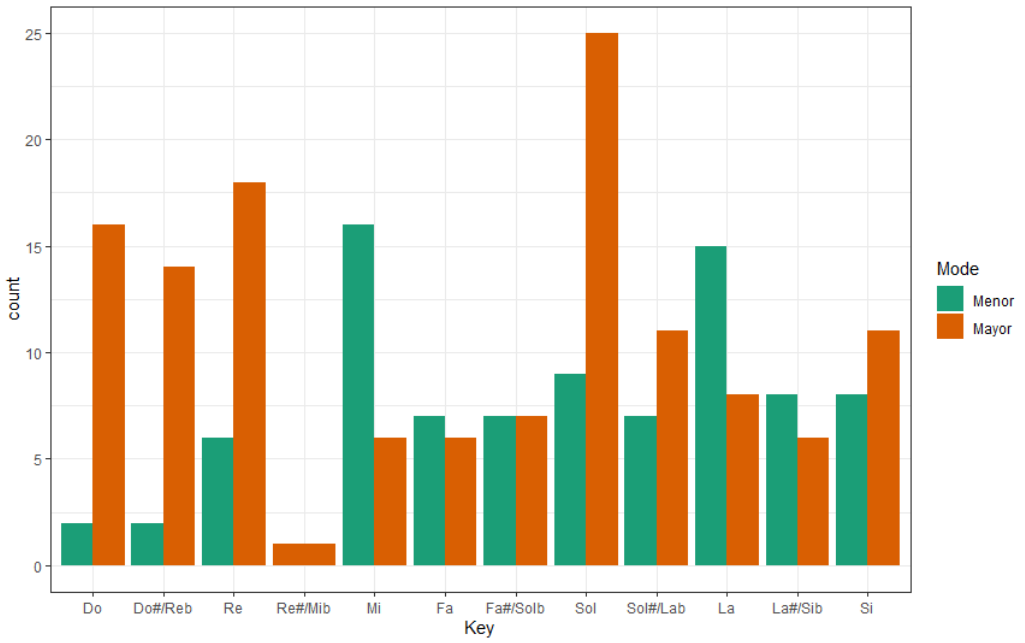


Figura 3.7: Distribución de las canciones por escala, y dentro de cada escala la división en mayor o menor.

mente se puede definir una nueva variable `fifths` de la siguiente forma:

$$fifths = (1 - mode)\frac{key}{11} + mode\frac{key + 3}{11} \mod 1$$

Esta nueva variable está escalada para que tome valores entre 0 y 1. Además considera dos escalas que usan las mismas notas pero con distinto `mode` muy cercanas y a partir de esa relación establecer la noción de distancia entre el resto. Se puede observar en la Figura 3.9 como la variable está bien distribuida y parecen observarse dos agrupaciones principales.

3.2.1. Conjunto de datos final

Con esta exploración del conjunto de datos, se ha conseguido crear un conjunto de datos con géneros bien distribuidos y variables significativas que aprovechan al máximo la información suministrada por la API de Spotify. Los datos sobre los que se aplicará el algoritmo quedan en 321 combinaciones de

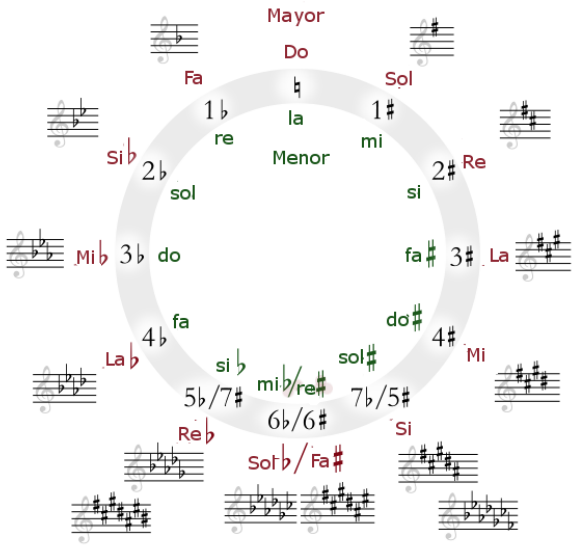


Figura 3.8: Representación visual del circulo de quintas.

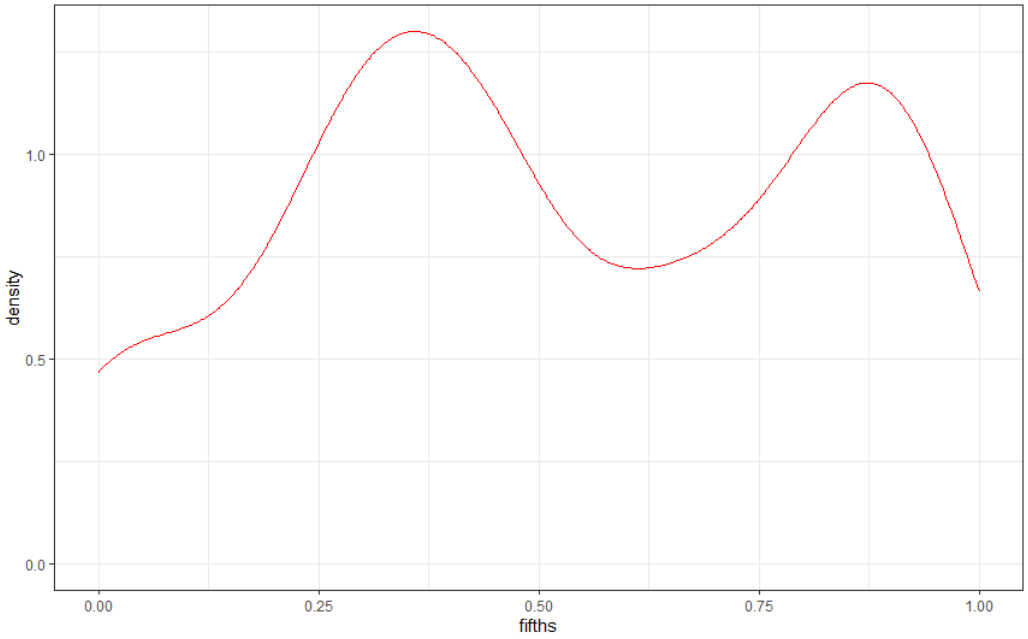


Figura 3.9: Distribución de los datos en la variable fifths.

`track.id` y género únicas, existiendo 5 géneros diferentes en el conjunto de datos: *edm*, *rock*, *trap*, *clásica* y *permanent wave*, con 10 variables numéricas

entre 0 y 1 para analiza: `danceability`, `energy`, `loudness`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, `valence`, `tempo`, y `fifths`.

En la Figura 3.10 se muestra la distribución en función del género final y la correlación entre variables.

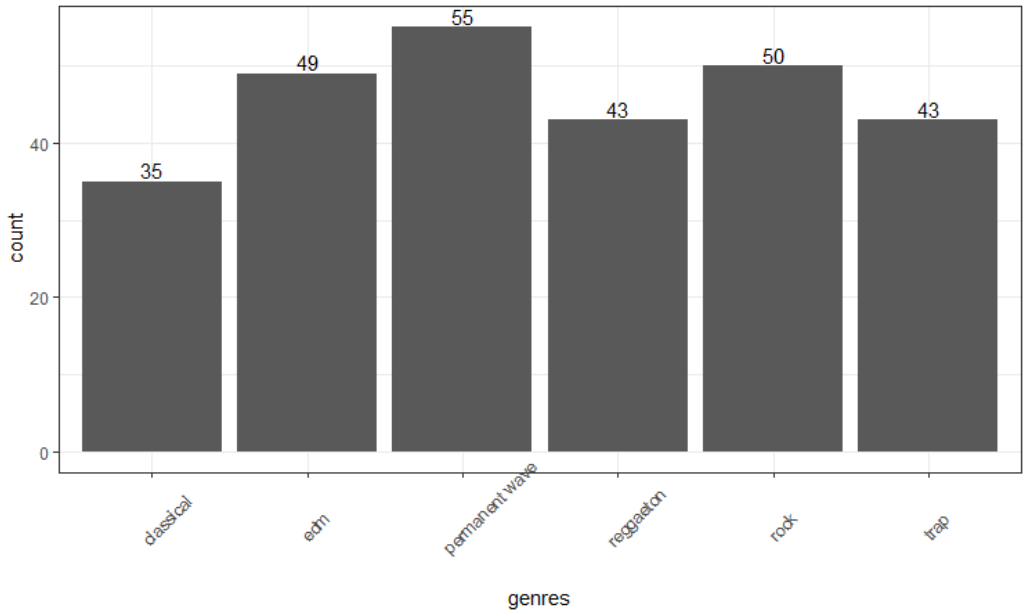


Figura 3.10: Distribución de los géneros tras el filtrado.

3.3. Aplicación de algoritmos de clustering

Una vez tratados los datos y obtenido el formato adecuado, se puede pasar a la aplicación de los algoritmos de *clustering*. En primer lugar, se va a realizar un análisis del número de grupos con los índices CH y de silueta. Se han usado 3 algoritmos diferentes (*k-means*, *single* y *complete*) para obtener las particiones de los índices, dando los 3 tendencias similares. En la Figura 3.11, se observa cómo ambos índices dan valores muy altos en dos grupos, y van descendiendo cuando aumenta el número de *clusters*.

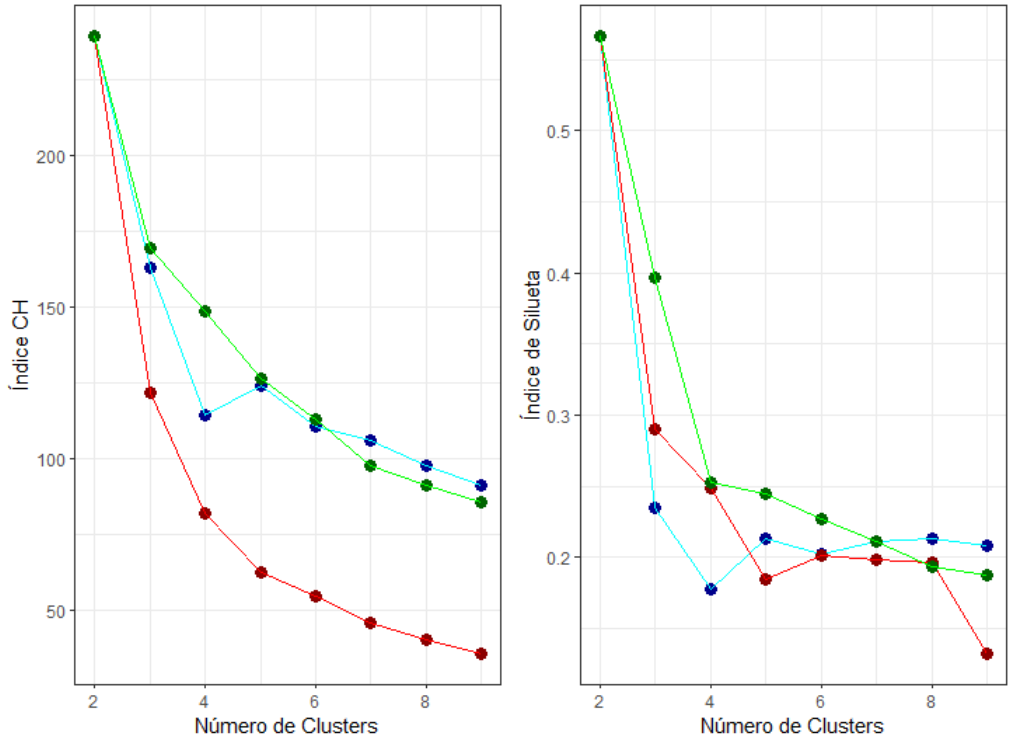


Figura 3.11: Valores del índice CH (izquierda) y de Silueta (derecha) en función del número de *clusters* para 3 algoritmos diferentes: *complete* (verde), *single* (rojo) y *k-means* (azul)

.

Este comportamiento puede indicar que las variables disponibles para el análisis no van a permitir diferenciar los géneros de partida, aunque los valores del índice cuando se usa *k-means* para obtener las particiones tiene un pequeño máximo local en $n = 5$. Ahora se podrían tomar dos direcciones

para el estudio: utilizar el número óptimo determinado por los índices de la figura, o utilizar la información conocida de los datos y fijar el número de *clusters* en 5. Se va a seguir por este ultimo camino, ya que aprovecha al máximo la información previa conocida sobre los datos y sigue el objetivo inicial del experimento: identificar lo que caracteriza y diferencia unos géneros musicales de otros.

Al realizar un *clustering* con el *single linkage* ocurre el previamente mencionado *chain effect*. Por este motivo, se descarta el método jerárquico con *single linkage* como útil para este estudio. Este comportamiento puede ser observado en la Figura 3.12. Diagramas de este tipo serán los utilizados para mostrar los resultados obtenidos por los algoritmos de clustering, ya que muestran para cada género su distribución en los grupos creados por el algoritmo de forma visual y sencilla de entender.

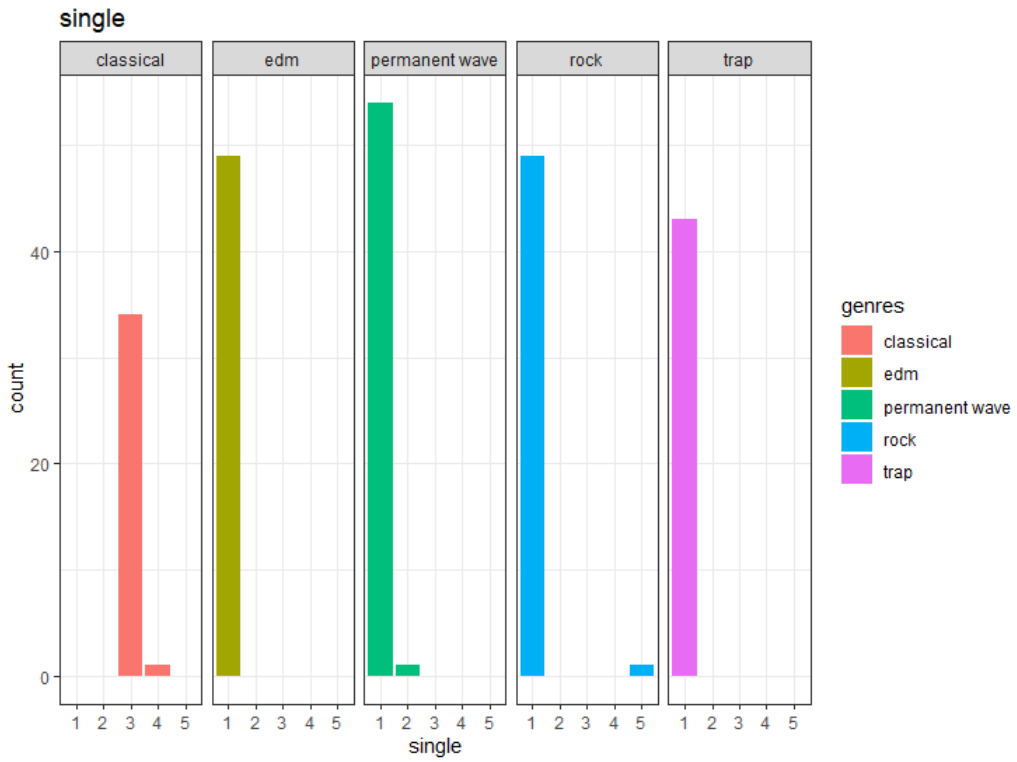


Figura 3.12: Resultados de una clasificación usando el método jerárquico *single linkage*. Se observa la distribución de cada género entre los grupos creados, con un claro efecto cadena.

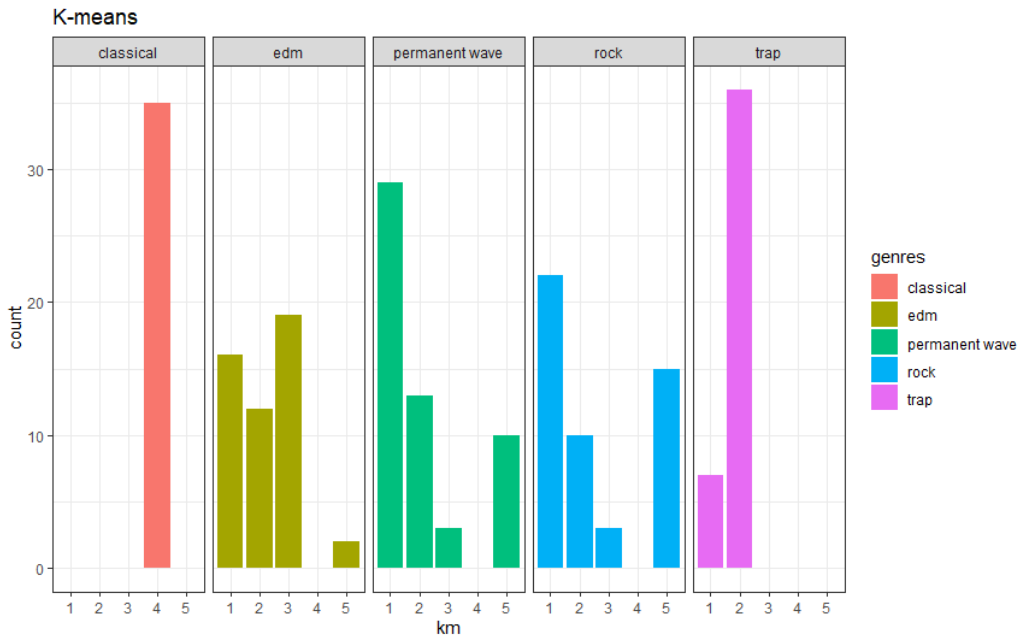
Una vez explicada la razón por la que el *single* no es apto para este estudio, se van a analizar los resultados del *k-means* y el algoritmo de clustering jerárquico con el *complete linkage*. Para garantizar que la partición obtenida por el algoritmo *k-means* no se ve influenciada por la inicialización aleatoria de los centros, se ha utilizado un método práctico, que consiste en repetir el algoritmo n veces (en este caso $n = 20$), y se toma como buena la partición que más minimiza la función coste.

En la Figura 3.13, se observan las particiones de ambos algoritmos cuando se fija el número de grupos en 5. Concretamente, se muestra en cada caja, para cada género del conjunto de datos el número de objetos que tiene en cada uno de los grupos identificados por el algoritmo. Ambos algoritmos obtienen resultados similares, pudiendo diferenciar completamente la música *clásica* del resto, y consiguiendo una buena acumulación de canciones de *trap* en un mismo grupo. Los otros 3 géneros están repartidos entre los grupos restantes, y en esos grupos es donde se observa más diferencia entre los algoritmos. Mientras que el *k-means* logra concentrar las canciones de *permanent wave* bastante bien en un sola clase, el *complete* las reparte más uniformemente. Sin embargo, el *complete* logra que el grupo 3 sea casi exclusivamente *rock*, mientras que *k-means* no consigue aislar ninguno de los 3 géneros restantes en un solo grupo.

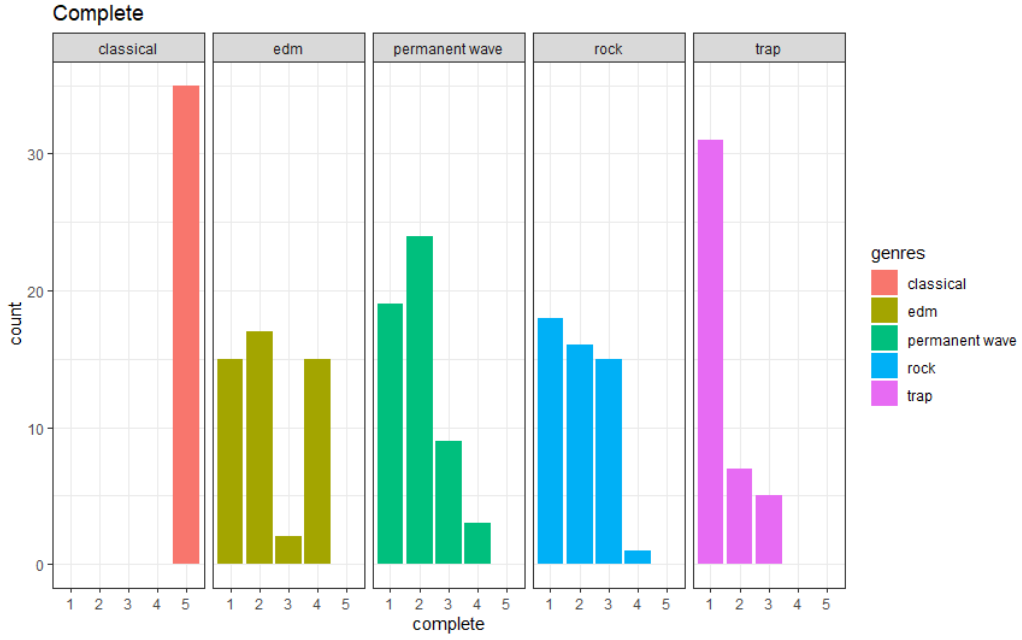
Lo primero que se deduce es que la música *clásica* es la que presenta unas características musicales más diferenciadas del resto de géneros, seguida del *trap*. Aunque era esperable que el *rock* y el *permanent wave* sean similares, es sorprendente la dificultad de separarlos del *edm*.

El siguiente paso es tratar de buscar qué variables están haciendo parecer similares al *edm*, *rock* y *permanent wave*. Para ello, se comenzará estudiando el comportamiento de 2 variables con una varianza menor de 0.005 (es decir, los datos no cambian mucho dentro de estas variables): `loudness` y `speechiness`. Como la música *clásica* ha sido completamente diferenciada, también es interesante buscar las variables críticas que la diferencian de las demás. Por tanto, se va a estudiar la densidad de las canciones en función de si son del género *classical* o no.

Los resultados se muestran en la Figura 3.14, donde se observa como todos los datos se acumulan en valores bajos de `speechiness`. En lo que respecta a `loudness`, todas las canciones que no son de música *clásica* presentan valores



(a) *k-means*



(b) *complete linkage*

Figura 3.13: Grupos del conjunto de datos para 5 *clusters* con dos algoritmos diferentes.

altos, mientras que la música *clásica* se reparte por la zona más baja. De estas figuras se sacan 2 conclusiones:

1. La variable `speechiness` no ayuda a diferenciar canciones ya que todas se acumulan en torno a valores bajos. Esto puede deberse a que esta variable solo es útil para diferenciar canciones de *tracks* hablados.

2. La variable `loudness` solo ayuda a diferenciar a la música *clásica* del resto de géneros.

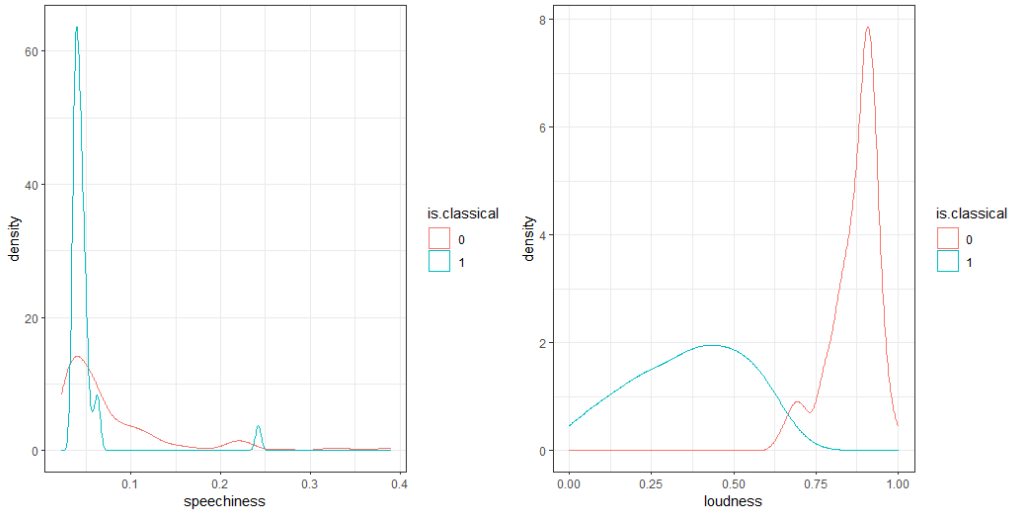
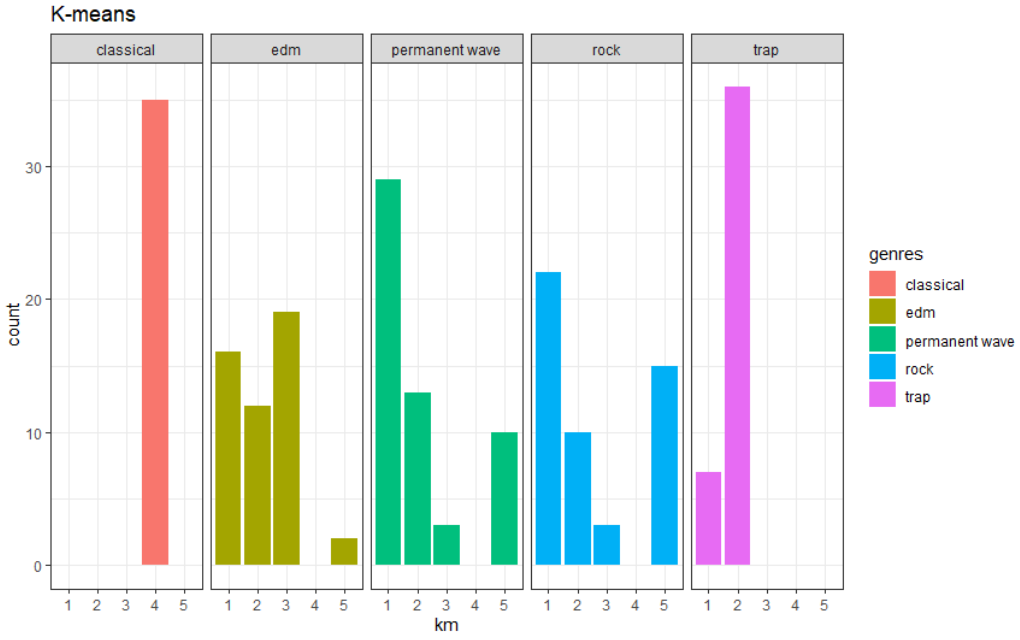


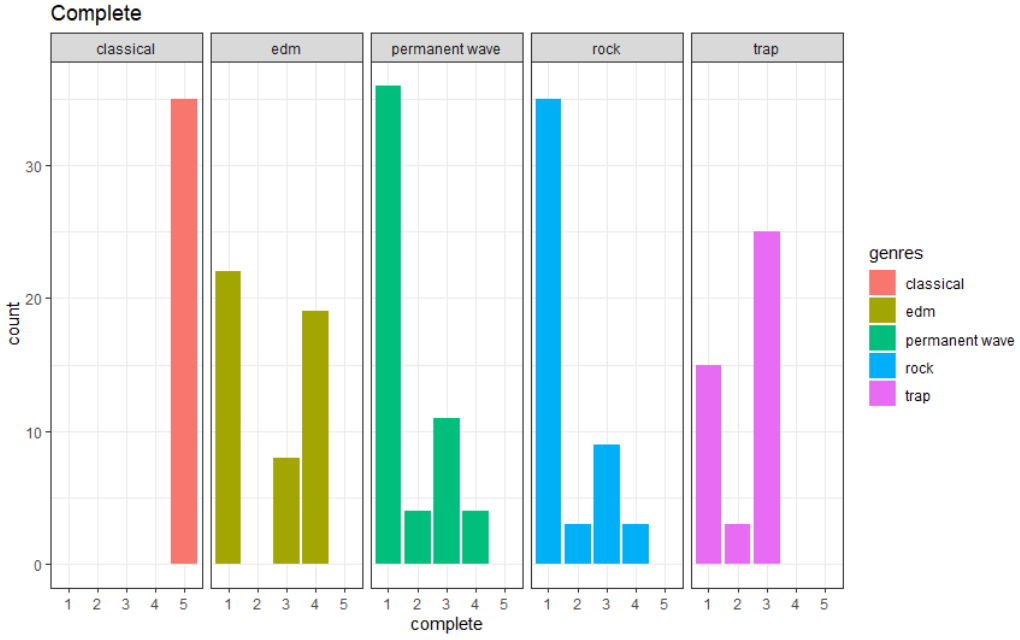
Figura 3.14: Distribución de los datos en las variables `loudness` y `speechiness` subdivididos en si pertenecen a `classical` (`is.classical` = 1) o no (`is.classical` = 0).

Para comprobar si se logra mejorar la partición sin perder los grupos de música *clásica* y *trap*, se va a realizar de nuevo *clustering*, pero esta vez eliminando las variables `loudness` y `speechiness`. Los resultados mostrados en la Figura 3.15.

Lo primero que se observa es como ambos algoritmos vuelven a diferenciar perfectamente la música clásica, indicando que probablemente hay varias variables donde la música clásica se comporta de forma muy diferente. Sin embargo, con el trap los resultados son diferentes: mientras que *k-means* mantiene el trap bastante agrupado, *complete* lo divide en 2 grupos esta vez. A su vez, *complete* logra agrupar rock y permanent wave muy bien, quedando



(a) *k-means*



(b) *complete linkage*

Figura 3.15: Clasificación del conjunto de datos, eliminando las variables *loudness* y *speechiness*, para 5 *clusters* con dos algoritmos diferentes.

bastante más repartidas en el *k-means*. Además, el edm sigue distribuyéndose de manera muy uniforme.

Tratar de seguir con este enfoque de eliminar variables que hacen que todos los datos se concentren, puede llevar a que se pierda tanta información que los resultados dejen de ser fiables. Por tanto, para concluir el *clustering*, se va a realizar un estudio más particular de los géneros que no se han conseguido clasificar con precisión.

El *edm* ha sido el género que peor se ha conseguido aislar, presentando la misma distribución entre los grupos en todas las clasificaciones realizadas. Por tanto, se va a tratar el subconjunto de datos de las canciones cuyo género es *edm*. En primer lugar, surge la pregunta de si es un conjunto homogéneo, es decir, si dentro del conjunto se pueden ver *clusters*. Haciendo uso una vez más de las métricas CH y silueta, con los algoritmos *k-means* y *complete* para realizar las particiones, se obtienen en ambos casos valores significativamente grandes cuando el número de *clusters* es 2, por lo que se puede intuir que el género *edm* muestra dos tendencias. Para encontrar esas tendencias, se recurre en primer lugar al PCA con el fin de identificar qué aspectos las causan. En la Tablas 3.1, se presentan los resultados obtenidos por un análisis de componentes principales dentro del subconjunto de canciones del género *edm*.

Componente principal	PC1	PC2	PC3	PC4	PC5
Desviación típica	0.3715	0.2558	0.1781	0.14036	0.13237
Varianza	0.4698	0.2229	0.1080	0.06708	0.05966
Varianza acumulada	0.4698	0.6927	0.8007	0.86781	0.92747

Componente principal	PC6	PC7	PC8	PC9	PC10
Desviación típica	0.09668	0.07375	0.05660	0.05142	0.02590
Varianza	0.03182	0.01852	0.01091	0.00900	0.00228
Varianza acumulada	96	0.97781	0.98871	0.99772	1

Tabla 3.1: Resultados de un análisis de componentes principales sobre los datos dentro del género edm. Se muestra la varianza y la desviación típica explicadas por cada componente y la varianza acumulada hasta cada una de ellas.

Se puede ver como las 3 primeras variables explican el 80 % de la varianza, y ya solo las dos primeras dan una cantidad de información muy grande. Si se observan los vectores de coeficientes de estas variables, se observa que la PCA1 le da un peso de 0.97 a la variable instrumentalidad; la PCA2 un peso de 0.95 a `valence` y la PCA3 un 0.87 a `liveness`. Esto es un buen indicio de que estas son las variables que causan la heterogeneidad en el *edm*. Tras un análisis de las canciones dentro del género *edm*, se concluye que ninguna es con una audiencia en directo, y que los valores grandes de esta variable (que son menos de 10) son demasiado pocos en relación al tamaño muestral (49 canciones de edm). Por tanto, el análisis se centrará solo en las dos primera componentes principales, que se puede suponer que coinciden con la `instrumentalness` y `valence` para una mejor visualización de los resultados.

En este caso, se está trabajando con unos datos de los que se sabe que están fuertemente relacionados (todos son considerados *edm*), pero se quiere ver si se identifica alguna tendencia. Por tanto, el *clustering* difuso es una herramienta útil, ya que puede aportar no solo agrupaciones, también grados de pertenencia. En la Figura 3.16, se observa como el algoritmo GK difuso divide los elementos del *edm* en dos basándose en sus dos primeras componentes principales. Las datos se ven claramente separados en dos grupos, con unas pertenencias mayores de las que podrían ser esperadas antes de aplicar el algoritmo a juzgar por el análisis previo.

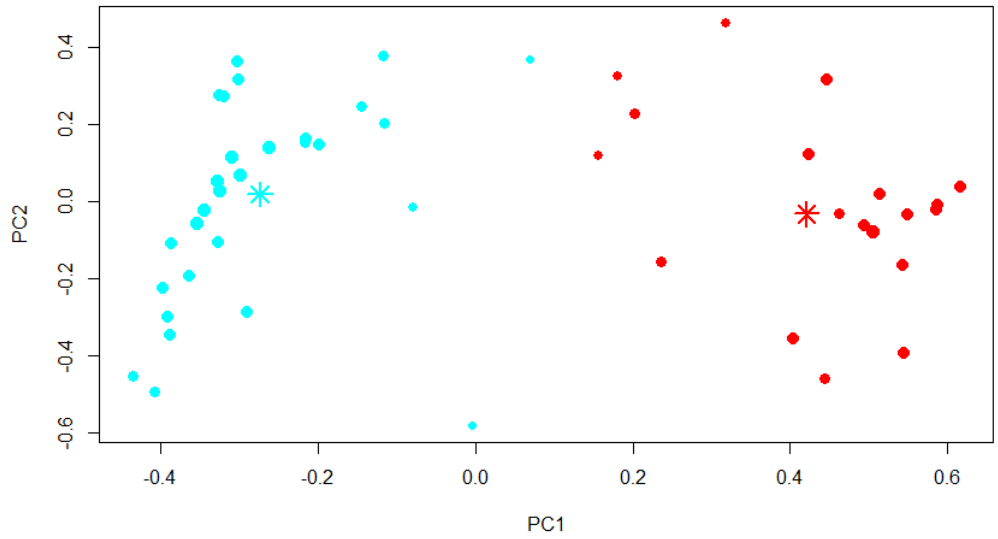


Figura 3.16: Clasificación de las canciones del género *edm* en función de sus dos primeras componentes principales. Cada color representa una agrupación formada por el método GK, y el tamaño de cada punto es proporcional a su grado de pertenencia más alto.

Con estos resultados, se concluye que el género *edm* en este conjunto de datos estaba formado principalmente por dos subgéneros muy diferenciados. Tras un análisis de las canciones y de los valores promedio de cada grupo de `valence` e `instrumentalness`, se concluye que uno de los grupos está compuesto por canciones con mucha letra, probablemente alegres y cercanas a un estilo más pop comercial hecho para poner en la radio, mientras que el otro es música con poca voz y mucho instrumento, quizás más apropiadas para festivales y no tanto para la radio. Con esto se ve como el *edm* contiene muchos subgéneros muy variados, lo que dificulta su clasificación frente a géneros como la música *clásica* con una serie de características comunes más marcadas.

El otro punto a estudiar es que en todas las clasificaciones realizadas, el *rock* y el *permanent wave* se han distribuido de maneras muy similares. En este caso, ya se ha descartado en la sección anterior la posibilidad de que las canciones sean *rock* y *permanent wave* simultáneamente, es más, este número es menor de 5. Por tanto, es necesario estudiar estos dos géneros de forma

aislada.

Con este fin, ya que los métodos más simples no logran diferencias estos géneros, se va a utilizar el *clustering* de densidad para tratar de entender mejor este subconjunto. Concretamente, se buscará obtener la *reachability plot* de las canciones que son *rock* o *permanent wave*. En la Figura 3.17 se puede observar como efectivamente se obtiene una gráfica relativamente plana con algún dato extremo. Además, los colores indican una clasificación realizada con DBSCAN, utilizando como criterio para fijar el ϵ el promedio de la *core distance* (0.3 en este caso). Se observa como la gran parte de los datos se van al mismo grupo, quedando dos pequeños grupos aislados y casi un 40 % de los datos considerados ruido. Todo esto refuerza la idea que el *rock* y el *permanent wave* están muy cercanos como géneros musicales, o que sencillamente sus diferencias características no están siendo analizadas en este estudio.

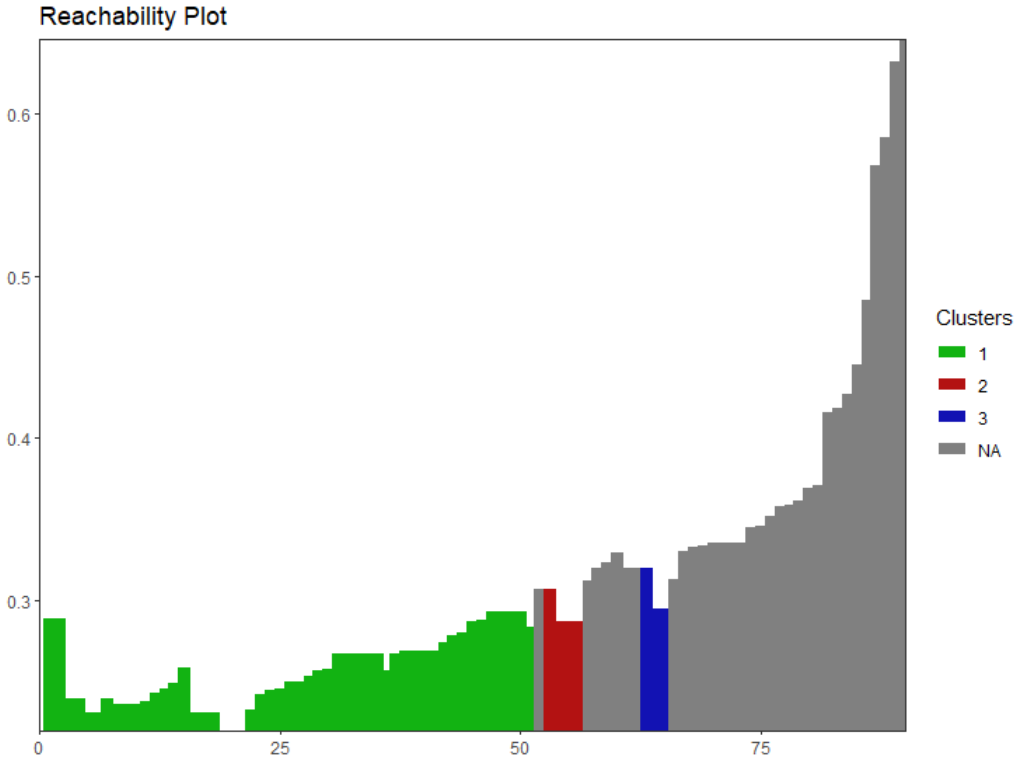


Figura 3.17: *Reachability plot* del subconjunto de canciones pertenecientes a *rock* o *permanent wave*. Los colores son el resultado de una agrupación usando DBSCAN, siendo el color gris el asociado al ruido.

Finalmente, para contrastar lo obtenido utilizando los métodos de densidad se va a recurrir de nuevo al *clustering* difuso. En este caso las componentes principales tienen repartida su varianza de forma más uniforme, por lo que una visualización directa de los datos es de poca utilidad. Sin embargo, se va a realizar un estudio de la matriz de grados de pertenencia U. Se buscará dividir el conjunto en 2 grupos, ya que es lo que indican las métricas CH y de silueta. En la Figura 3.18 se puede observar una gráfica con la diferencia en valor absoluto entre los grados de pertenencia a cada uno de los 2 grupos de cada elemento del conjunto, ordenados de forma descendiente. Se observa como ningún elemento tiene una diferencia mayor de 0.5 entre coeficientes, y además la mayoría de los datos obtienen valores muy similares. Esto quiere decir que las canciones están todas muy cerca unas de otras y de pertenecer tanto a un grupo como a otro. Esto junto a la *reachability plot* anterior son fuertes apoyos a la idea de que el *rock* y el *permanent wave* son indistinguibles con las características musicales estudiadas.

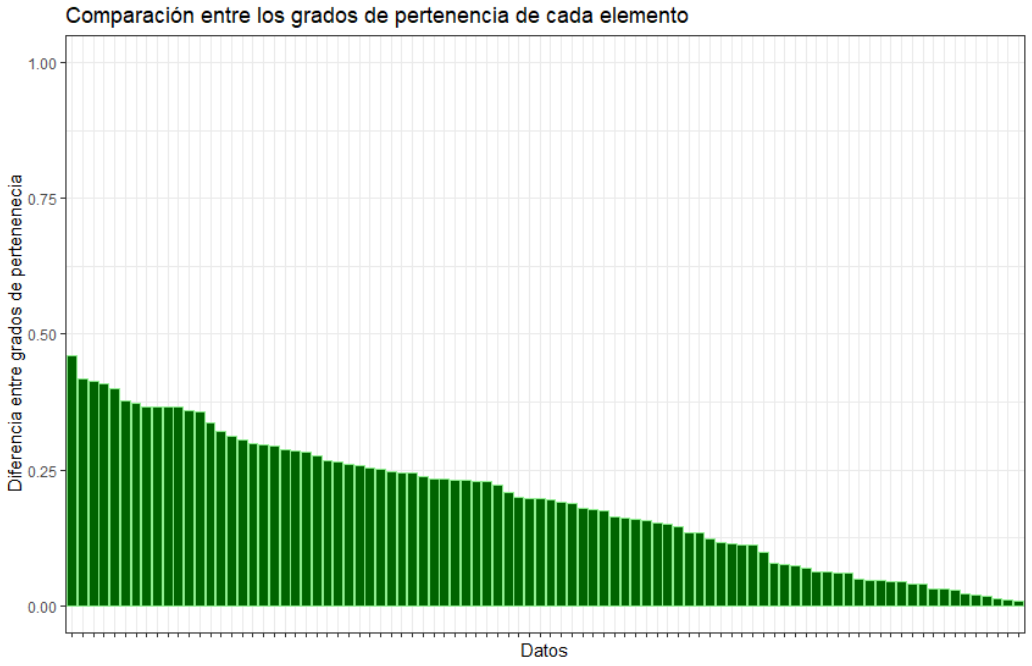


Figura 3.18: Diferencia en valor absoluto entre los dos grados de pertenencia asociados a cada dato al realizar un *clustering* difuso con el algoritmo GK.

	Partición 1	Partición 2	Sin edm	Sin edm ni permanent wave
k-means	0,266	0,266	0,342	0,531
complete	0,206	0,163	0,268	0,421

Tabla 3.2: Valores del índice ARI para las distintas particiones.

3.3.1. Resultados del clustering

Finalmente, tras realizar varios *clusterings*, es necesario buscar obtener resultados numéricos en los que apoyarse para evaluar el resultado de la partición. Para ello se hará uso del ARI mencionado en la sección de *cluster validation*, ya que es la medida más fiable para ver lo cerca que han estado los algoritmos de identificar los géneros musicales. Se obtendrá el ARI de cada una de las distintas particiones obtenidas frente a la clasificación por géneros proporcionada por Spotify.

Al conjunto de datos total se le han realizado 2 *clusterings*: el primero sin alterar nada, y un segundo eliminando `speechness` y `loudness`. Además, se han realizado *clusterings* con el *k-means* y con el agrupamiento jerárquico utilizando *complete linkage*. Se ha comprobado como la música *edm* no se puede identificar bien debido a que la muestra tomada está compuesta únicamente de dos subgéneros muy distintos, y que el *rock* y el *permanent wave* no son diferenciables con las variables de estudio. Por esto, también es interesante estudiar la partición 1 sin el género *edm*, y también sin *edm* ni *permanent wave*. Los índices ARI de cada caso se muestran en la Tabla 3.2.

Se observa como eliminar variables no ha tenido efecto en el *k-means*, y en el *complete* ha hecho que disminuya el índice Rand ajustado. Con esto se concluye que eliminar variables, aunque parezca mejorar la clasificación, en realidad no tiene un efecto positivo en ningún caso de los trabajados. Sin embargo, al no considerar el género *edm* en la evaluación se consiguen mejoras en ambos algoritmos, reafirmando que es este género el que mete ruido en los grupos creados. Esta mejora se vuelve aun más considerable al eliminar también el *permanent wave*. Además, parece que el algoritmo *k-means* logra resultados ligeramente mejores de forma consistente.

Los resultados de la partición completa son bastante bajos, incluso los que se obtienen eliminando el *edm* están muy cerca del 0, que sería una

clasificación aleatoria. Es necesario eliminar también el *permanent wave* para aumentar el valor del índice hasta valores cercanos a 0.5.

En conclusión, inicialmente los *clusterings* realizados en un primer estudio daban unos resultados relativamente pobres a la hora de identificar los 5 géneros musicales. Esto era de esperar porque ya al estimar un número de *clusters* con los índices CH y silueta los máximos se encontraban cercanos a 2 *clusters*. Además, eliminar variables que no aportaban nada al conjunto y que solo hacían parecer los datos más cercanos no afectó de forma positiva a la partición. Sin embargo, tras un análisis en detalle se detectó que el *edm* no estaba bien representado en el conjunto y que el *rock* y el *permanent wave* no son distinguibles con las variables utilizadas. Eliminando el *edm* y el *permanent wave* del análisis se observan ya valores aceptables del Rand ajustado, confirmando las sospechas de que estos son los géneros peor identificados. Por tanto, se ha logrado una buena clasificación de 3 de los 5 géneros deseados: *rock*, *trap* y música *clásica*.

Capítulo 4

Conclusiones

En este trabajo, se han estudiado los fundamentos teóricos del *clustering* de datos, estableciendo la noción de similitud y planteando herramientas matemáticas necesarias para los algoritmos discutidos más adelante. Para ello, se han desarrollado en detalle las motivaciones, ventajas y puntos débiles de los algoritmos más comunes como el (*k-means* o *complete linkage*). También se han estudiado algoritmos de clustering relacionados con la densidad de los datos, como son los algoritmos no paramétricos DBSCAN y OPTICS. Para concluir con los algoritmos, se ha estudiado una familia de algoritmos difusos que amplían la noción de pertenencia, permitiendo así un mejor entendimiento de conjuntos de datos con *clusters* poco diferenciados. Para concluir el Capítulo 2, se han tratado los índices de evaluación CH y Silueta, con el fin de estimar el número de *clusters* en el que el algoritmo debería de dividir una muestra de datos, y también el índice Rand para contrastar la semejanza entre la partición obtenida por el conjunto y la clasificación ya conocida del conjunto.

Una vez comprendidos los distintos algoritmos y métricas asociados al proceso de *clustering*, se procede a aplicar la teoría a un ejemplo real. Con este fin, se construye un conjunto de datos musicales con 12 variables numéricas y una etiqueta con su género musical (con un total de 5 géneros diferentes). Con en análisis de este conjunto se busca que el algoritmo aprenda cómo distinguir unos géneros de otros así como las características clave de cada uno de ellos. Los resultados del análisis han sido variados. Por un lado, la música clásica

ha sido claramente identificada en todos los casos, especialmente por su bajo volumen y alta instrumentalidad. Por otro lado, aunque en menor medida, el género *trap* ha logrado también ser distinguido de forma satisfactoria. Sin embargo, la muestra de *edm* no ha podido aislarse bien mediante ningún algoritmo, debido a la gran variedad de subgéneros del *edm*. Respecto a los géneros *rock* y *permanent wave*, aunque se consiguen aislar con éxito de forma conjunta del resto de géneros, no han sido distinguibles por ningún algoritmo. La razón de esto puede ser que las variables utilizadas no son suficientes para diferenciar estos dos géneros.

Este estudio proporciona una introducción a diversas técnicas de *clustering* que permiten afrontar una gran variedad de escenarios y problemas. Además, el ejemplo del análisis musical ha dado buenos indicadores de qué géneros pueden ser diferenciados según las variables estudiadas. Este tipo de análisis musicales son muy útiles para las plataformas de reproducción de música, permitiendo crear algoritmos de búsqueda y recomendación de canciones para cada usuario en función de sus gustos (Kim et al. [14]). El campo de la recomendación de canciones está en constante desarrollo y en busca de nuevos métodos (Zhang et al. [32], Sassi et al. [12], Cataltepe & Altinel [7]). Aun con algoritmos básicos y sin grandes modificaciones, se han conseguidos resultados muy satisfactorios, con valores del índice Rand ajustados entre 0.4 y 0.5 si se tiene en cuenta que el *edm* no se puede clasificar bien en esta partición y que el *rock* y el *permanent wave* son indistinguibles.

El conjunto de datos ha sido creado gracias a los datos ofrecidos de forma pública por Spotify, un servicio de música digital que da acceso a millones de canciones y también a información relativa a las mismas. Sin embargo, una de las limitaciones del análisis ha sido la cantidad de datos con los que se ha podido trabajar, el cual se ha visto reducido debido a un gran número de artistas sin género asociado en Spotify, lo cual ha forzado a descartar muchas de las canciones consideradas inicialmente. Esto remarca la importancia de análisis de este tipo, los cuales podrían utilizarse para asociar a estas canciones cuyo género es desconocido un género real.

Para concluir, se considera que aún siendo el presentado un análisis profundo de los datos obtenidos, existen posibles maneras de ampliar este estudio y continuar dicha investigación en el futuro. La más obvias serían: aumentar la cantidad de canciones para obtener conjuntos más representativos de todos los géneros, utilizar otro tipo de algoritmos de *clustering* para ver si

aumenta la efectividad, o hacer uso de otro tipo de variables. Esta última opción es muy interesante, ya que aunque en este trabajo solo se ha podido trabajar con las variables proporcionadas por Spotify al público, la gran mayoría son valores promedios para toda la canción y por tanto pueden alejarse ligeramente de la realidad en puntos concretos de la misma. Por ejemplo, si se pudiera conocer la evolución a lo largo de la canción de cada una de las variables estudiadas podría aportar mucha información extra y descubrir nuevas diferencias entre géneros o encontrar similitudes mediante el uso de series temporales.

Bibliografía

[1] M. Ankerst, M. M. Breunig, H. Kriegel, y J. Sander. OPTICS: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, 1999.

[2] G. H. Ball y D. J. Hall. *ISODATA, a novel method of data analysis and classification*. Stanford, CA, 1965.

[3] L. Bottou y Y. Bengio. Convergence properties of the k-means algorithms. In G. Tesauro, D. Touretzky, y T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1995.

[4] L. Breiman. Random forest. *Mach. Learn.*, 45(1):5–32, 2001.

[5] R. L. Brennan y R. J. Light. Measuring agreement when two observers classify people into categories not defined in advance. *British Journal of Mathematical and Statistical Psychology*, 27(2):154–163, 1974.

[6] T. Calinski y J. Harabasz. A dendrite method for cluster analysis. *Commun. Stat. Theory Methods*, 3(1):1–27, 1974.

[7] Z. Cataltepe y B. Altinel. Music recommendation based on adaptive feature and user grouping. In *2007 22nd international symposium on computer and information sciences*, pages 1–6, 2007.

[8] J. M. Chambers, W. S. Cleveland, B. Kleiner, y P. A. Tukey. *Graphical methods for data analysis*. Chapman & Hall/CRC, Filadelfia, PA, Estados Unidos de América, 1983.

[9] R. Cilibrasi, P. Vitanyi, y R. de Wolf. Algorithmic clustering of music. In *Proceedings of the Fourth International Conference onWeb Delivering of Music, 2004. EDELMUSIC 2004.*, pages 110–117, 2004.

[10] M. M. Deza y E. Deza. *Encyclopedia of Distances*. Springer, Berlín, Alemania, 2018.

[11] L. Hubert y P. Arabie. Comparing partitions. *J. Classif.*, 2(1):193–218, 1985.

[12] S. B. Yahia I. B. Sassi y I. Liiv. Morec: At the crossroads of context-aware and multi-criteria decision making for online music recommendation. *Expert Systems with Applications*, 183:115375, 2021.

[13] A. K. Jain, M. N. Murty, y P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3), 1999.

[14] D. Kim, K. Kim, K. Park, J. Lee, y K. M. Lee. A music recommendation system with a dynamic k-means clustering algorithm. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 399–403, 2007.

[15] V. Kumar y J. Kumar Chhabradinseh. Performance evaluation of distance metrics in the clustering algorithms. *INFOCOMP*, 13(1):38–51, 2014.

[16] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, y N. Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65, 2017.

[17] P. Macnaughton-Smith, W. T. Williams, y L. G. Dale, M.B .and Mockett. Dissimilarity analysis: A new technique of hierarchical sub-division. *Nature*, 202:1034–1035, 1964.

[18] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, y S. D. Brown. An introduction to decision tree modeling. *J. Chemom.*, 18(6):275–285, 2004.

[19] L. Narens. A general theory of ratio scalability with remarks about the measurement-theoretic concept of meaningfulness. *Theory Decis.*, 13(1):1–70, 1981.

[20] L. Narens. On the scales of measurement. *J. Math. Psychol.*, 24(3):249–275, 1981.

[21] M. A. Patwary, D. Palsetia, A. Agrawal, W. Liao, F. Manne, y A. Choudhary. Scalable parallel OPTICS data clustering using graph

algorithmic techniques. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2013. ACM.

[22] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.*, 66(336):846, 1971.

[23] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65, 1987.

[24] Santhi y V. M. Bhaskaran. Improving the efficiency of image clustering using modified non euclidean distance measures in data mining. *Int. J. Comput. Commun. Control*, 9(1):56, 2014.

[25] A. Singh, A. Yadav, y A. Rana. K-means with three different distance metrics. *Int. J. Comput. Appl.*, 67(10):13–17, 2013.

[26] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946.

[27] J. W. Tukey. The future of data analysis. *ann. math. stat.*, 33(1):1–67, 1962.

[28] F. van de Vijver y K. Leung. *Methods and data analysis for cross-cultural research*. Cambridge University Press, Cambridge, Inglaterra, 2 edition, 2021.

[29] E. Wagenmakers, R. Wetzels, D. Borsboom, H. L. J. van der Maas, y R. A. Kievit. An agenda for purely confirmatory research. *Perspect. Psychol. Sci.*, 7(6):632–638, 2012.

[30] S. Watanabe. *Pattern Recognition: Human and Mechanical*. JohnWiley andSons, Inc, New York, NY, 1985.

[31] K. West y P. Lamere. A model-based approach to constructing music similarity functions. *EURASIP J. Adv. Signal Process.*, 2007(1), 2006.

[32] Y. C. Zhang, D. Ó Séaghdha, D. Quercia, y T. Jambor. Auralist: Introducing serendipity into music recommendation. WSDM '12, page 13–22, New York, NY, USA, 2012. Association for Computing Machinery.