BIG DATA

UD2. Caso práctico 1

Proyecto de clasificación de canciones con R

María de Gracia Fernández Suárez

ÍNDICE

INTRODUCCIÓN

Para este caso práctico perteneciente a la unidad 2, debemos realizar un proceso de clustering sobre una base de datos de canciones. Cuando hablamos de clustering nos referimos a "un algoritmo de machine learning no supervisado que organiza y clasifica diferentes objetos, puntos de datos u observaciones en grupos o clústeres basados en similitudes o patrones". Es decir, agrupaciones de datos parecidos.

Para realizar este clustering he utilizado el lenguaje de programación R y como IDE, R Studio. R Studio es un entorno de desarrollo integrado del lenguaje R que se utiliza en el análisis y ciencia de datos. Para la instalación he seguido los pasos descritos en el temario de la UD2.

ELECCIÓN DEL DATASET

En el caso práctico se nos proporciona el enlace a una web llamada http://millionsongdataset.com/ , es una base de datos sobre canciones, al descargar la versión reducida, he solo me permitía descargarla en formato HDF5. Este formato requiere el uso de librerías específicas (rhdf5, h5py, etc.) y técnicas avanzadas de manipulación de datos jerárquicos, ya que los datos aparecen clasificados en carpetas.

Dado que el objetivo de este caso práctico es aplicar técnicas de clustering y análisis exploratorio, y no el preprocesamiento de datos a gran escala, he considerado más adecuado utilizar una versión del dataset en formato .csv, el cual se compone de un conjunto de datos en formato tabular con variables similares a las utilizadas en el milliondataset.

Para ello, he utilizado Kaggle. Kaggle es una plataforma gratuita que pose a disposición de los usuarios una serie de problemas para solucionar con temáticas como la ciencia de datos, el análisis predictivo y lo machine learning. En ella puedes descargar datasets de diferentes temáticas creadas por usuarios.

Primero, elegí un dataset que “copia” al milliondataset, pero tuve que cambiar a otra versión. Al abrir R Studio, cargué dicho data set en una variable, al a ver la función View() sobre dicha variable me apareció lo siguiente:

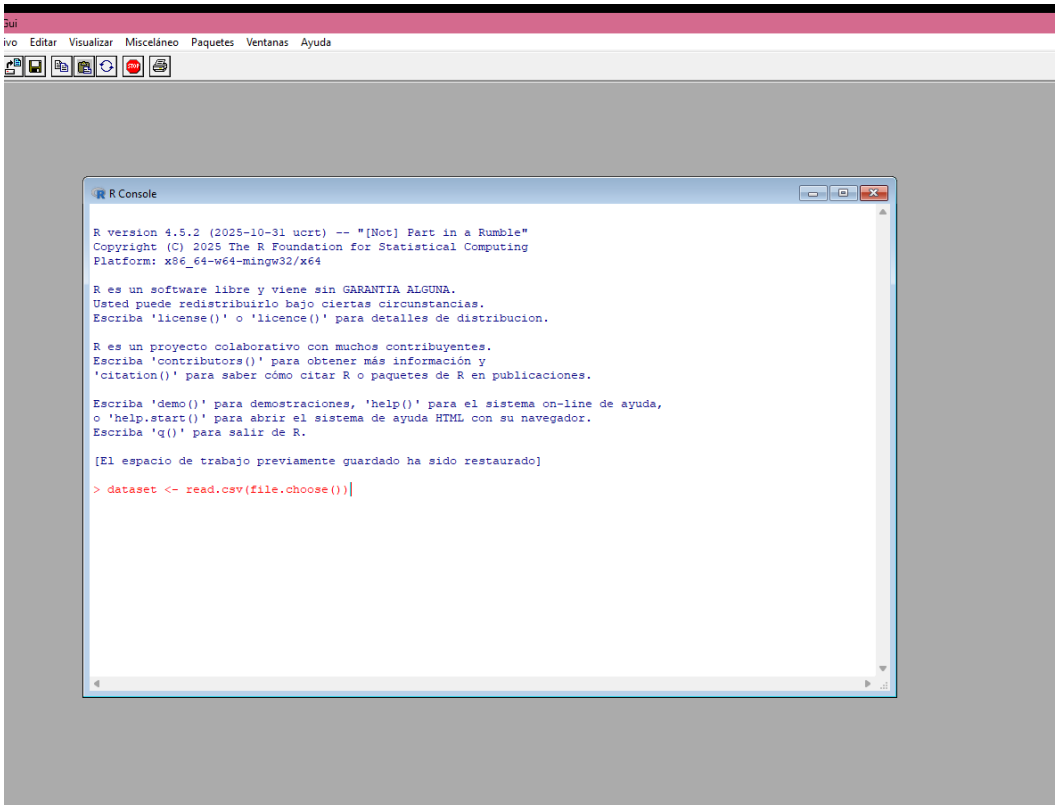
32	Like a Stone	77	293960
33	It's Been Awhile	65	264706
34	I Hate Everything About You	75	231480
35	Rollin' (Air Raid Vehicle)	74	213760
36	Fat Lip	74	178266
37	The Pretender	11	269373
38	Savior	73	242280
39	Bodies	74	201960
40	Sugar	We're Goin Down	79
41	0.6990000000000001;;		
42	Last Nite	70	193373
43	Through Glass	60	282946
44	The Diary of Jane - Single Version	69	200546

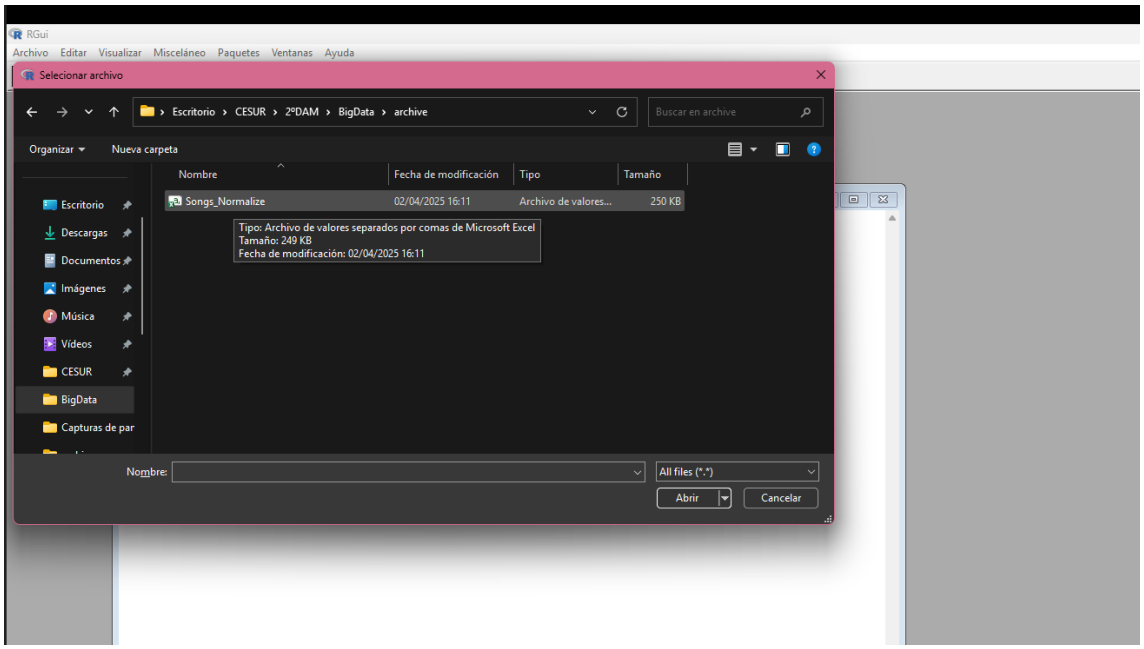
Había registros donde sus atributos, por ejemplo, título de la canción, aparecían en la columna de popularidad. Esto se debe a que el archivo presentaba desalineación de columnas, probablemente causada por delimitadores inconsistentes. En la práctica, esto genera que atributos numéricos y categóricos se mezclen en la misma columna.

La falta de integridad en las columnas suponía un riesgo para el análisis, ya que una agrupación basada en valores incorrectos llevaría a conclusiones erróneas. Por este motivo, y siguiendo las buenas prácticas de análisis de datos, opté por utilizar un dataset alternativo, limpio y correctamente estructurado.

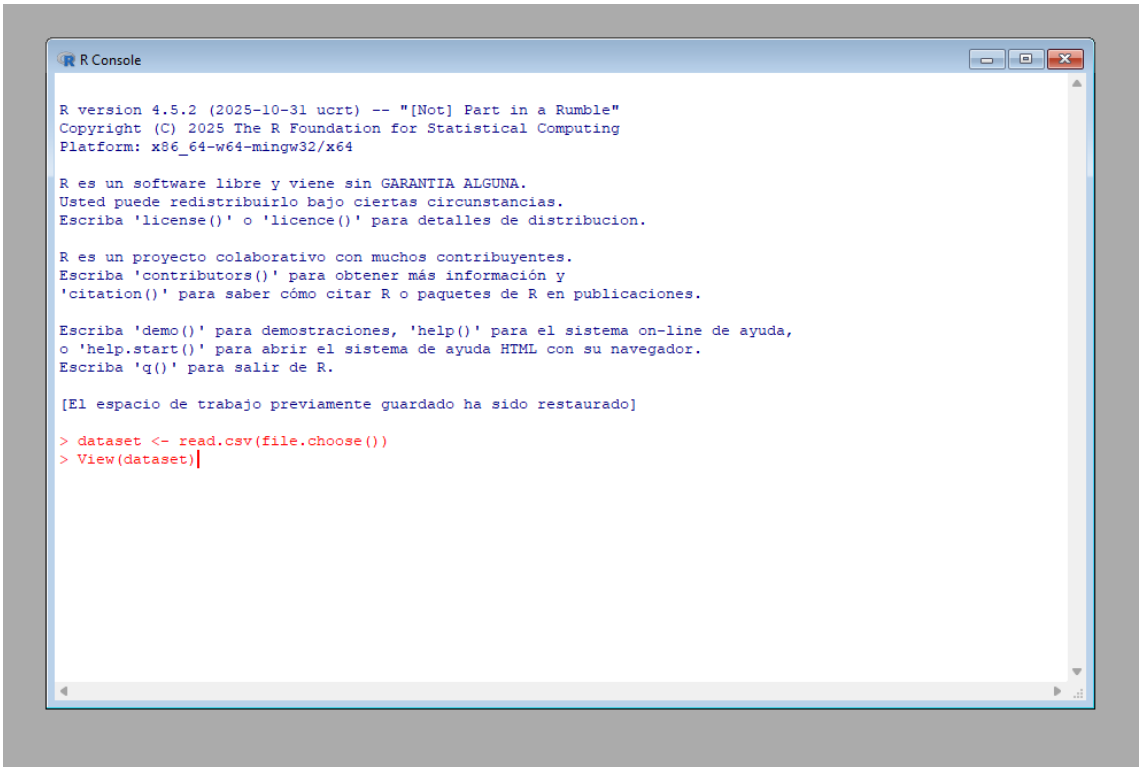
Finalmente, escogí un dataset titulado: Songs Normalize Dataset que mantenía los atributos similares al dataset original y al visualizar la tabla, los datos sí estaban bien estructurados.

Guardé el dataset en una variable llamada dataset. Al encontrarse en formato .csv utilicé la función read.csv() y dentro de dicha función utilicé la función file.choose() porque me resulta más cómodo utilizar el explorador de archivos que copiar la ruta.

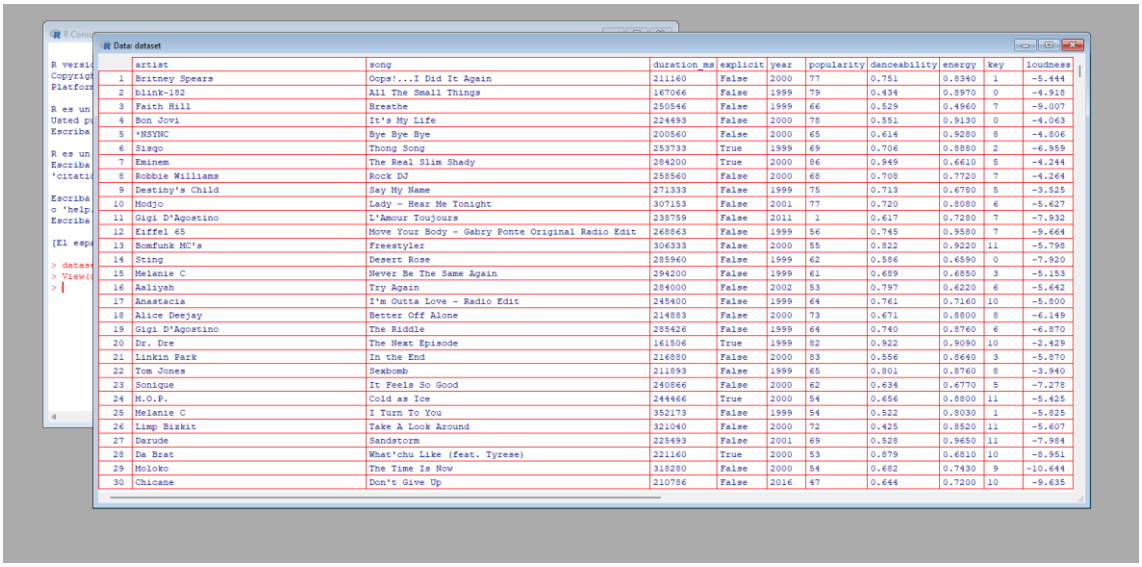




Utilizo la función View() pasándole por parámetro la variable que contiene el dataset:



Al ejecutarlo, pulsando enter, me aparece una ventana con los datos. Comprobé que este archivo sí estuviera bien estructurado. Al comprobar que sí estaba bien, procedí a realizar el ejercicio con este dataset.



	artist	song	duration_ms	explicit	year	popularity	danceability	energy	key	loudness
1	Britney Spears	Oops!...I Did It Again	211160	False	2000	77	0.751	0.8340	1	-5.444
2	blink-182	All The Small Things	167066	False	1999	79	0.434	0.8970	0	-4.918
3	Faith Hill	Breathe	250546	False	1999	66	0.529	0.4960	7	-9.007
4	Bon Jovi	It's My Life	224493	False	2000	78	0.551	0.9130	0	-4.063
5	*NSYNC	Bye Bye Bye	200560	False	2000	65	0.614	0.9280	8	-4.806
6	Sisqo	Thong Song	253733	True	1999	69	0.706	0.8880	2	-6.959
7	Eminem	The Real Slim Shady	284200	True	2000	86	0.949	0.6610	5	-4.244
8	Robbie Williams	Rock DJ	258560	False	2000	68	0.708	0.7720	7	-4.264
9	Destiny's Child	Say My Name	271333	False	1999	75	0.713	0.6780	5	-3.525
10	Modjo	Lady - Hear Me Tonight	307153	False	2001	77	0.720	0.8080	6	-5.627
11	Gigi D'Agostino	L'Amour Toujours	238759	False	2011	1	0.617	0.7280	7	-7.932
12	Eiffel 65	Move Your Body - Gabry Ponte Original Radio Edit	268863	False	1999	56	0.745	0.9580	7	-9.664
13	Bomfunk MC's	Freestyler	306333	False	2000	55	0.822	0.9220	11	-5.798
14	Sting	Desert Rose	285960	False	1999	62	0.586	0.6590	0	-7.920
15	Melanie C	Never Be The Same Again	294200	False	1999	61	0.689	0.6850	3	-5.153
16	Aaliyah	Try Again	284000	False	2002	53	0.797	0.6220	6	-5.642
17	Anastacia	I'm Outta Love - Radio Edit	245400	False	1999	64	0.761	0.7160	10	-5.800
18	Alice Deejay	Better Off Alone	214883	False	2000	73	0.671	0.8800	8	-6.149
19	Gigi D'Agostino	The Riddle	285426	False	1999	64	0.740	0.8760	6	-6.870
20	Dr. Dre	The Next Episode	161506	True	1999	82	0.922	0.9090	10	-2.429
21	Linkin Park	In the End	216880	False	2000	83	0.556	0.8640	3	-5.870
22	Tom Jones	Sexbomb	211893	False	1999	65	0.801	0.8760	8	-3.940
23	Sonique	It Feels So Good	240866	False	2000	62	0.634	0.6770	5	-7.278
24	M.O.P.	Cold as Ice	244466	True	2000	54	0.656	0.8800	11	-5.425
25	Melanie C	I Turn To You	352173	False	1999	54	0.522	0.8030	1	-5.825
26	Limp Bizkit	Take A Look Around	321040	False	2000	72	0.425	0.8520	11	-5.607
27	Darude	Sandstorm	225493	False	2001	69	0.528	0.9650	11	-7.984
28	Da Brat	What'chu Like (feat. Tyrese)	221160	True	2000	53	0.879	0.6810	10	-8.951
29	Moloko	The Time Is Now	318280	False	2000	54	0.682	0.7430	9	-10.644
30	Chicane	Don't Give Up	210786	False	2016	47	0.644	0.7200	10	-9.635

Si quisiéramos utilizar una ruta para acceder al archivo, se realizaría de la siguiente forma:

```
Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[El espacio de trabajo previamente guardado ha sido restaurado]

> canciones_data <- read.csv("C:\Users\mdgra\Desktop\CESUR\2°DAM\BigData\archive\Songs Normalize.csv")
```

PRUEBAS CLUSTERING

Para realizar el clustering y el análisis posterior, así como la mayoría de las pruebas, he utilizado el método k-means. Además, siguiendo con las indicaciones de la rúbrica he buscado otros algoritmos que se pueden utilizar para realizar un clustering.

Después de realizar la búsqueda, he elegido PAM como segundo algoritmo porque funciona de forma similar a k-means, pero es más robusto ante valores atípicos y utiliza puntos reales del conjunto de datos como representantes de cada clúster.

En k-means, el centro de cada clúster es un centroide, que es un punto calculado matemáticamente, es la media de todos los puntos del clúster. PAM elige como centro del clúster un punto real del dataset, llamado medoid.

En este apartado muestro cómo se realicé las pruebas con ambos métodos, junto con varias capturas de pantalla del proceso. En el apartado siguiente realizaré el análisis e interpretación de los resultados obtenidos.

K-MEANS

El algoritmo k-means es muy utilizado en el aprendizaje automático no supervisado. El objetivo es dividir los datos en grupos que sean lo más parecidos posibles entre sí.

Para ello, se tiene que especificar el número de grupos que queremos obtener (es la k). La idea es minimizar la variación total dentro de cada grupo, también conocida como variación intra-cluster. Esto se logra mediante el cálculo de la suma de las distancias euclidianas al cuadrado entre los elementos y el centroide correspondiente.

La distancia euclidiana es la medida más corta entre dos puntos. Es decir, este algoritmo va a elegir aleatoriamente un punto que es el centro de cada cluster, y va a calcular las distancias del resto de puntos respecto a este centro. Cada punto se asigna al centroide más cercano, y después los centroides se recalculan en función de los puntos asignados. Este proceso se repite hasta que los centroides dejan de cambiar significativamente.

Para ello, la función en R que vamos a utilizar es:

kmeans(dataset, k ,nstart=25).

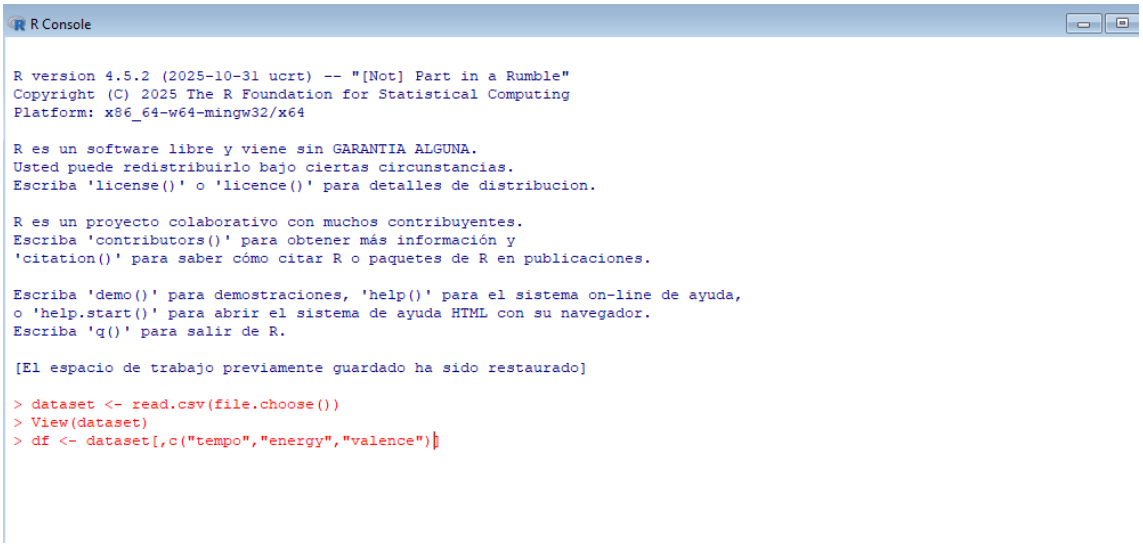
Donde:

- *dataset* es nuestro archivo csv con las columnas que queremos analizar tras utilizar el método scale() sobre él.
- k el número de clústeres que queremos obtener.

- nstart=25. Indica cuántas veces se va a ejecutar el algoritmo k-means desde diferentes centroides iniciales aleatorios. Después de estas 25 ejecuciones, R se queda con la mejor solución (la que tenga menor variación intra-clúster). Se recomienda que sea de 25 a 50 veces para mayor precisión.

He dicho antes que el dataset utilizado en el algoritmo es el dataset tras haber usado el método scale(). Este método estandariza los datos para que tengan toda una media de 0 y una desviación estándar se 1.

Esto es importante porque k-means utiliza distancias euclidianas, y si las variables están en diferentes escalas, las que tengan valores más grandes dominarían el cálculo de las distancias. Al estandarizar, todas las variables contribuyen por igual al proceso de clustering.



Para elegir qué atributos voy a analizar, guardo esa información en una variable llamada df (dataframe (estructura de datos organizada en filas y columnas)). Para ello necesito: el dataset original, seguido entre corchetes, del número de filas que quiero, al querer en este caso todas, lo dejo en blanco y pongo un coma; luego se utiliza la letra c de combine, para indicar entre paréntesis y entrecomillado los atributos.

En esta primera prueba utilicé tempo, energy y valence. La valencia está relacionada con la “emoción de la canción”, es decir una valencia alta indica que la canción es alegre, y una valencia baja significa que la canción es triste. Por lo que quería comprobar si estaba relacionado el tempo y la energía de la canción con ese atributo.

Una vez seleccionado los atributos y creado el dataframe, utilizo el método scale() para estandarizar los datos, como expliqué con anterioridad.

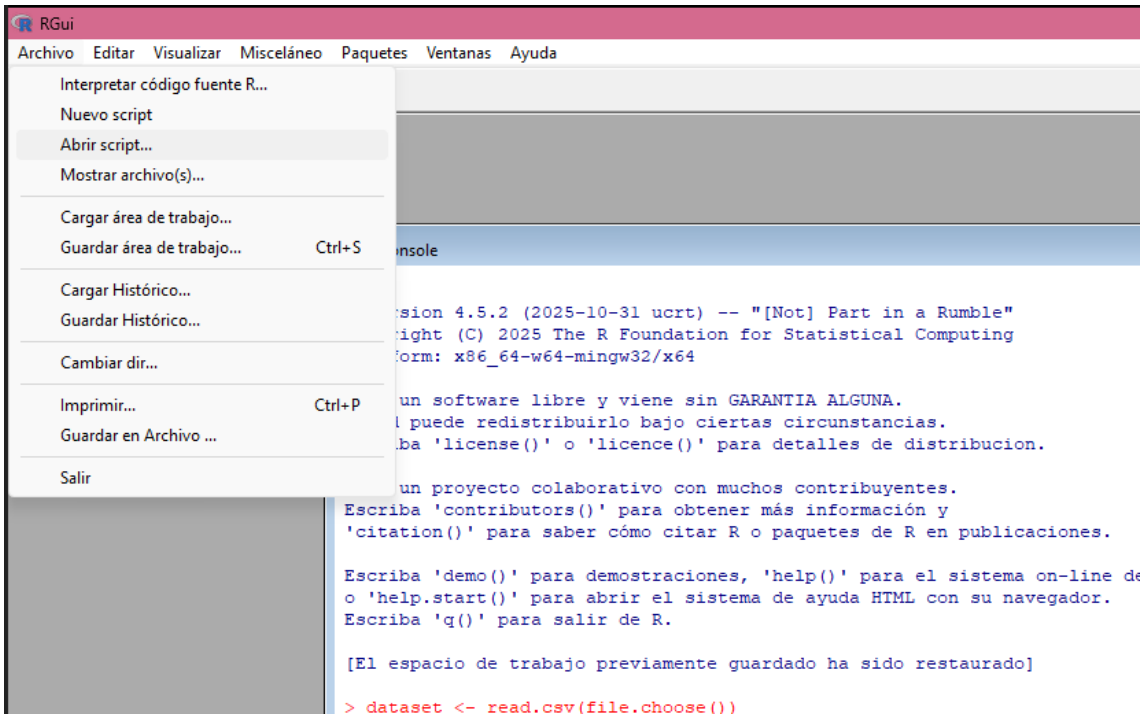
```
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[El espacio de trabajo previamente guardado ha sido restaurado]

> dataset <- read.csv(file.choose())
> View(dataset)
> df <- dataset[,c("tempo","energy","valence")]
> df_scaled <- scale(df)
```

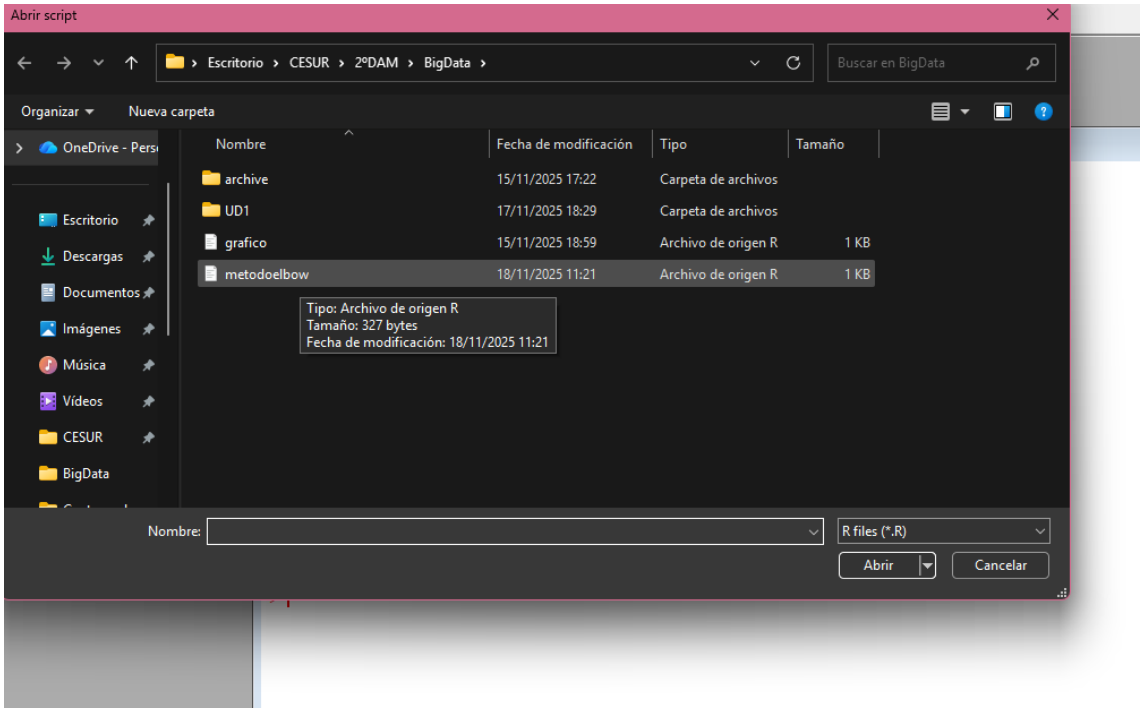
Una vez que los datos estaban preparados y normalizados,y antes de realizar el algoritmo k-means, surgió la pregunta: ¿cuál es el número adecuado de clústeres? Una opción era ir probando cada cifra una a una; pero realicé una búsqueda para encontrar si había algún método en R que pudiera darme respuesta a esa pregunta. Tras investigar distintas opciones encontré un método llamado el método Elbow.

Este método consiste en representar gráficamente cómo disminuye la variabilidad intraclúster (la suma de las distancias al cuadrado dentro de cada grupo) a medida que aumentamos el número de clústeres.

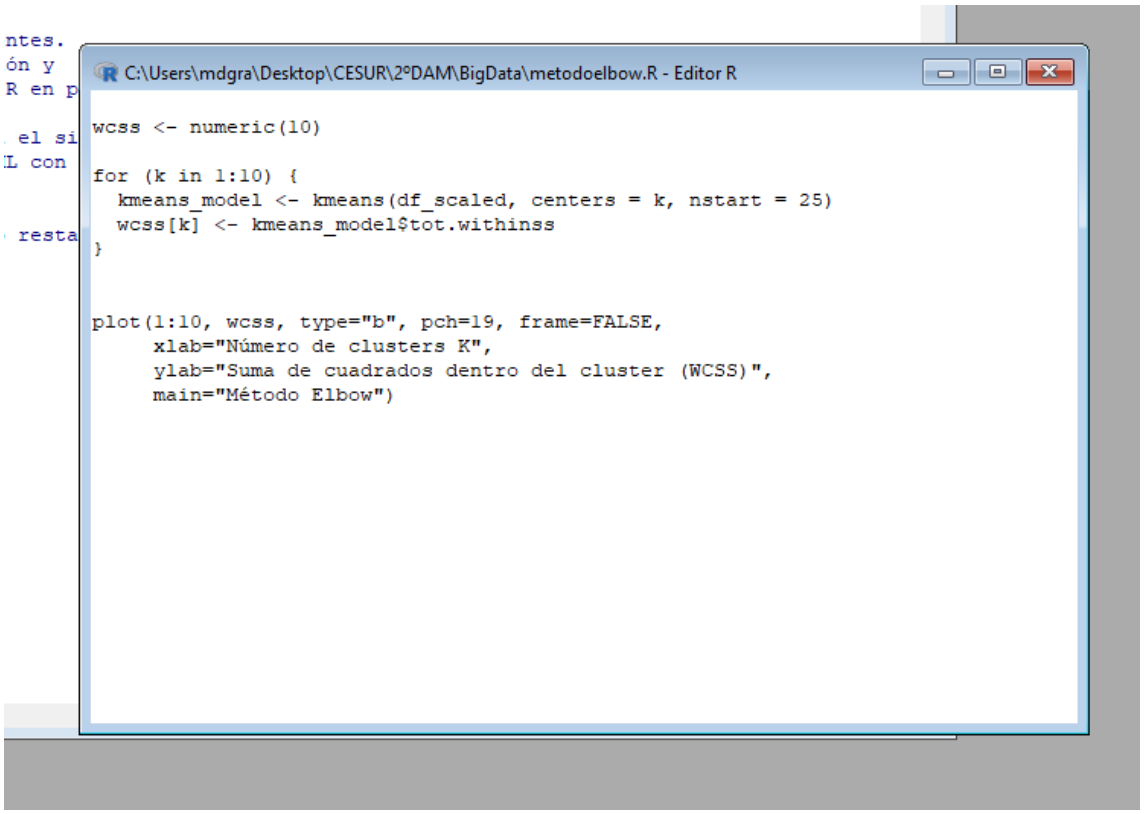
Este caso práctico lo he estado realizando en varios días, por lo que las líneas de código para realizar el método Elbow que encontré, las guardé en un script. Para ello, hice clic en Archivo – Nuevo Script. Puse ahí el código y guardé el script en mi carpeta de la asignatura.

Si tienes guardado un script, simplemente tienes que hacer clic en Archivo – buscar tu script en el explorador de archivos. Se ejecuta una ventana aparte con el script. De ahí modifiqué dicho script con el nombre de mi dataframe dentro de los parámetros del k-means y copié y pegué el código en mi área de trabajo.

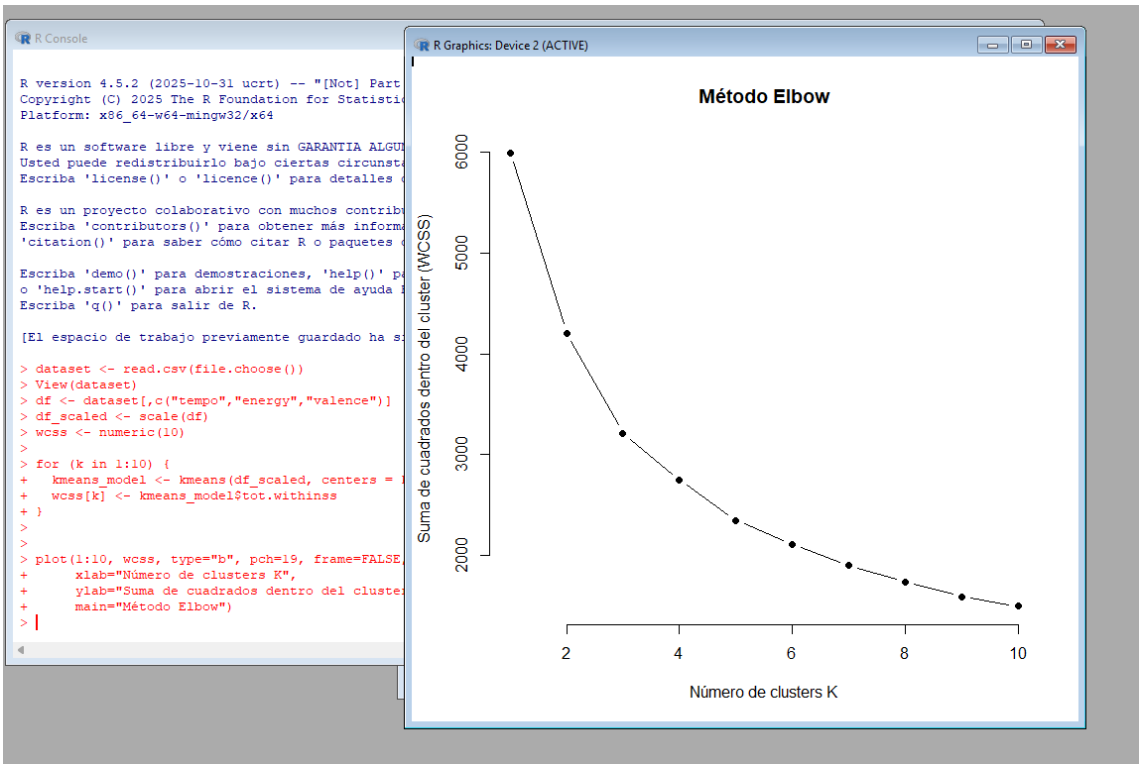

RGui
Archivo Editar Visualizar Misceláneo Paquetes Ventanas Ayuda
Interpretar código fuente R...
Nuevo script
Abrir script...
Mostrar archivo(s)...
Cargar área de trabajo...
Guardar área de trabajo... Ctrl+S
Cargar Histórico...
Guardar Histórico...
Cambiar dir...
Imprimir... Ctrl+P
Guardar en Archivo ...
Salir
nsole
sion 4.5.2 (2025-10-31 ucrt) -- "[Not] Part in a Rumble"
ight (C) 2025 The R Foundation for Statistical Computing
form: x86_64-w64-mingw32/x64
un software libre y viene sin GARANTIA ALGUNA.
l puede redistribuirlo bajo ciertas circunstancias.
ba 'license()' o 'licence()' para detalles de distribucion.
un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.
Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.
[El espacio de trabajo previamente guardado ha sido restaurado]
> dataset <- read.csv(file.choose())


Abrir script
Escritorio › CESUR › 2ºDAM › BigData ›
Buscar en BigData
Organizar
Nueva carpeta
OneDrive - Pers
Escritorio
Descargas
Documentos
Imágenes
Música
Vídeos
CESUR
BigData
Nombre
Fecha de modificación
Tipo
Tamaño
archive 15/11/2025 17:22 Carpeta de archivos
UD1 17/11/2025 18:29 Carpeta de archivos
grafico 15/11/2025 18:59 Archivo de origen R 1 KB
metodoelbow 18/11/2025 11:21 Archivo de origen R 1 KB
Tipo: Archivo de origen R
Tamaño: 327 bytes
Fecha de modificación: 18/11/2025 11:21
Nombre:
R files (*.R)
Abrir
Cancelar

Podemos ver que lo que realiza este script y este método es mediante un bucle for ir realizando el k-means sobre nuestro dataframe, hasta 10 veces, y dibuja luego dichos resultados en una gráfica.

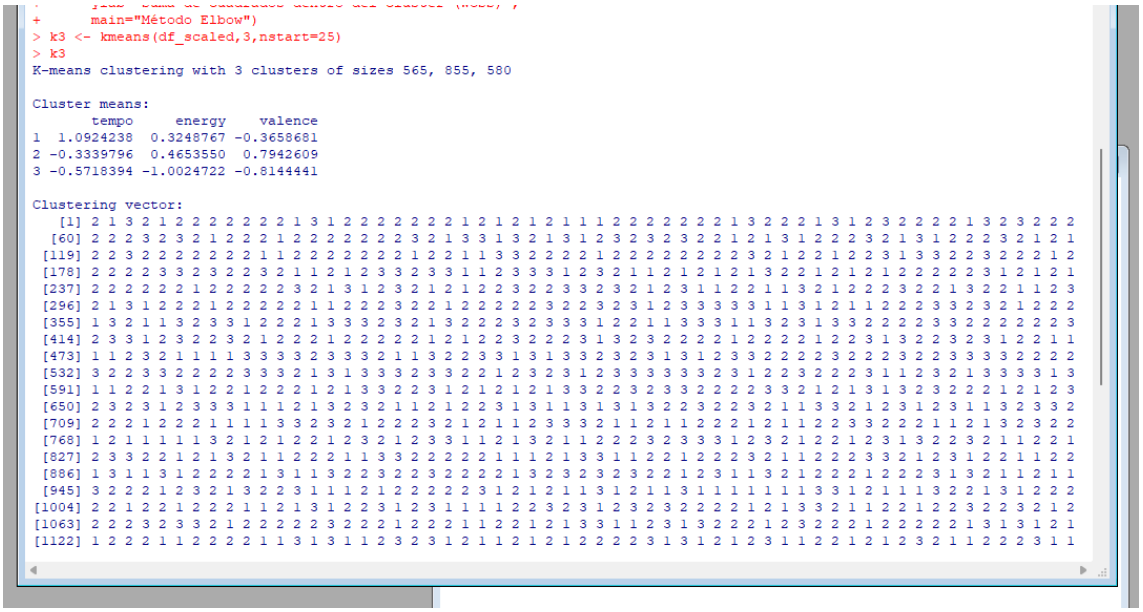


En la gráfica que se generó se puede observar que pasar de 1 a 2 clúster reduce de forma significativa la variabilidad, al igual que pasa de ampliar de 2 a 3. A partir del 3, a mi parecer es cuando la disminución empieza a no ser tan significativa. De todas formas, decidí probar a realizarlo tanto con k=3 como con k=4.

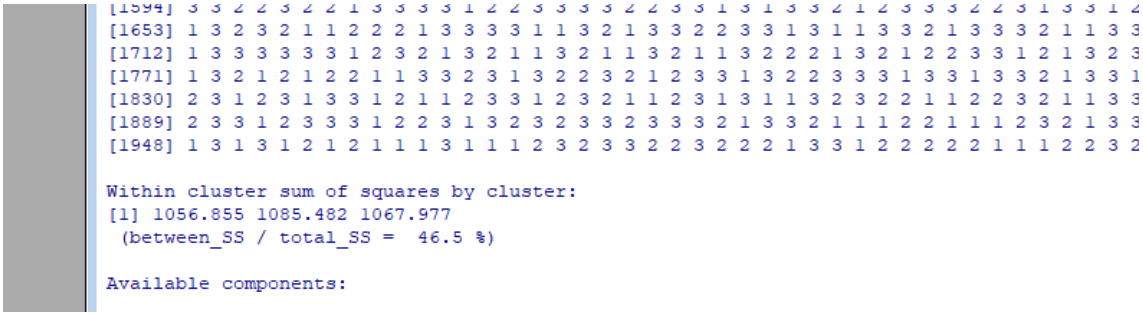


Ahora sí, hice mi primera prueba con el algoritmo k-means. Para ello utilicé la fórmula explicada anteriormente en el apartado y guardé los resultados en una variable llamada k3.

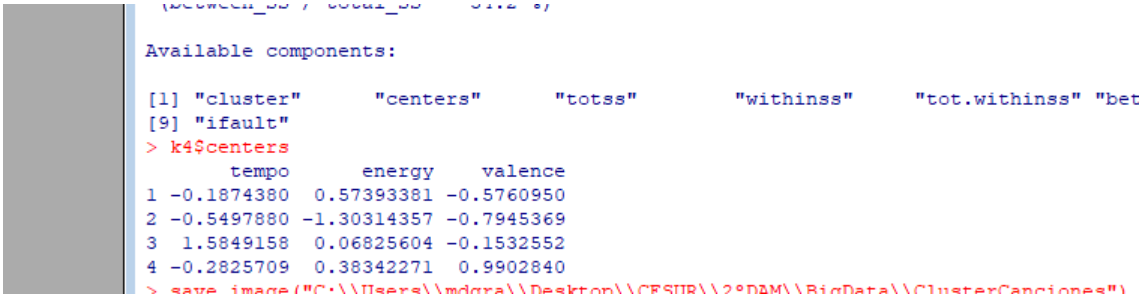
En los resultados podemos observar que se hacen tres clústeres de 565, 855 y 580 elementos cada uno. Podemos interpretar que hay tres perfiles/grupos musicales diferenciados. Con canciones rápidas (tempo elevado), energía moderada y tono emocional algo negativo en el clúster 1. En el segundo, canciones más lentas (tempo negativo), energía un poco más elevada y más positivas (valencia cercana a 1); y en el tercero tenemos un extremo con canciones lentas (tempo bajo), energía baja y tono emocional más “depresivo/triste”.



El valor between_SS / total_SS obtenido fue del 46.5%. Esto significa que el modelo k-means explica aproximadamente el 46.5% de la variabilidad total de los datos a través de la separación entre los clústeres. En otras palabras, casi la mitad de la dispersión del conjunto de datos se debe a diferencias reales entre los grupos formados.



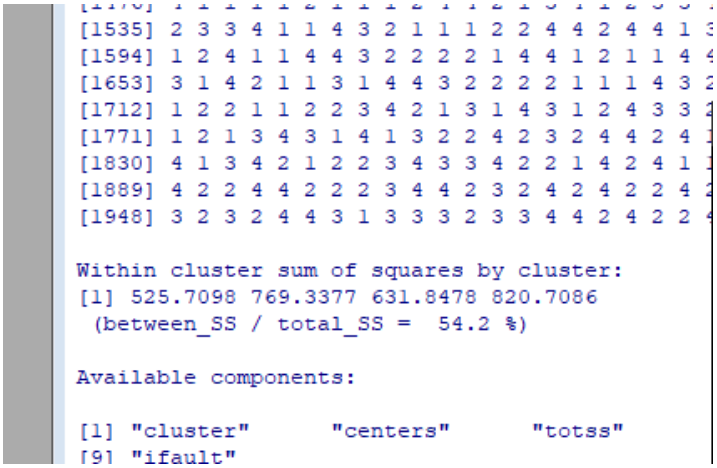
Realicé la prueba siguiendo los mismos pasos, pero esta vez con k=4 y guardé los resultados en una variable llamada k4. Si escribo k4$centers nos aparen los centroides de cada clúster. Apareciéndonos los siguientes valores:



Se puede seguir observando cómo es realizan grupos de canciones, me ha llamado la atención que la mayoría de los clústeres tenían una valencia negativa. Cuando realizamos 4 clústeres el valor between_SS / total_SS aumenta a un 54.2%. , lo que implica una mejora estadística en la capacidad del modelo para explicar la variabilidad total.

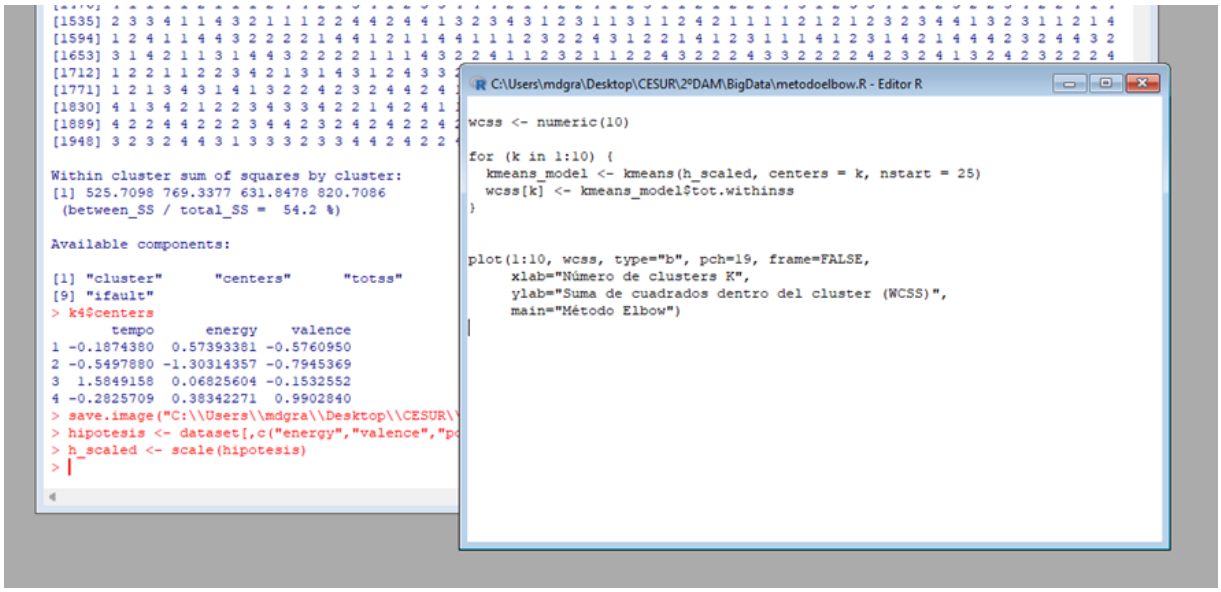
Sin embargo, es importante señalar que un incremento en este porcentaje no implica necesariamente que el modelo sea mejor desde el punto de vista interpretativo.

El k-means siempre aumentará la varianza explicada al añadir más clústeres, ya que divide los datos en grupos más pequeños y específicos. No obstante, si el número de clústeres crece demasiado, la interpretación de los grupos pierde claridad y la utilidad práctica del análisis disminuye.

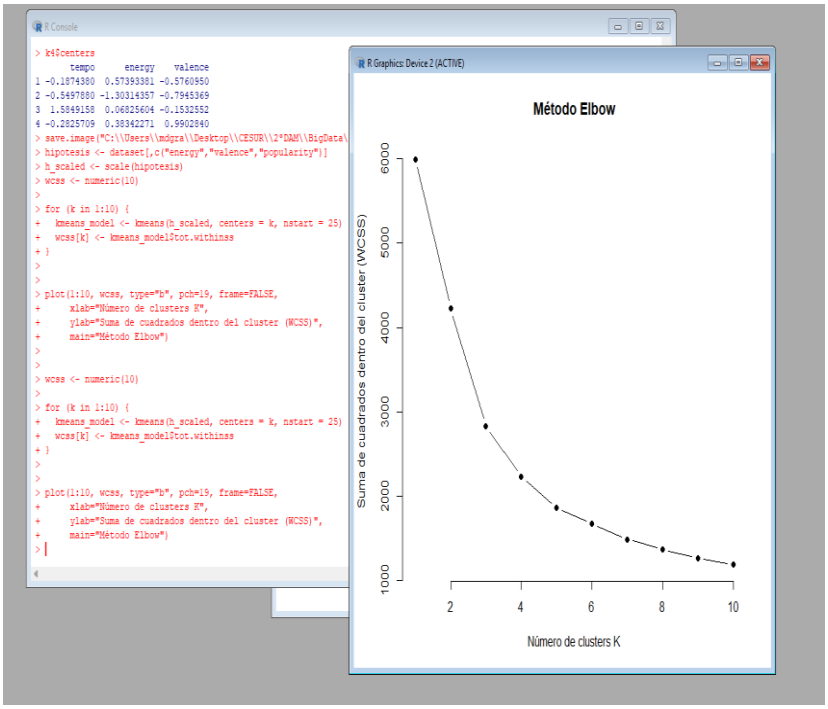


Tras observar, como he dicho antes, que me llamaba la atención que la mayoría de los clústeres tenían una valencia negativa, decidí realizar una nueva prueba, pero esta vez cambiando la variable tempo por popularity. ¿Influye el tono emocional de la canción/valencia con la popularidad de estas? Por lo que repetí el mismo proceso de antes:

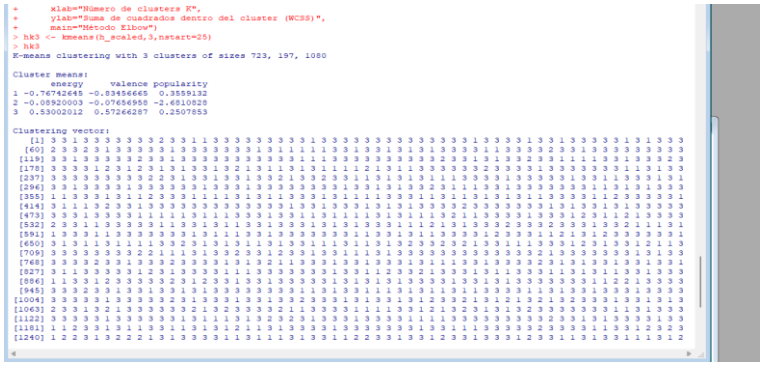
- Guardé en una variable, esta vez llamada hipotesis, el dataframe que iba a utilizar: *hipotesis <- dataset (,c("energy","valence","popularity")]ç*
- Normalicé los datos: *h_scaled <- scale(hipotesis)*
- Utilicé el script del método elbow para determinar los clústeres para realizar el k-means.



En el gráfico se nota claramente que hay una diferencia en la bajada del gráfico a partir de k=3. Por lo que, para comprobar mi hipótesis, utilicé el k-means con k=3.



Los resultados obtenidos me parecieron muy interesantes: la popularidad es alta tanto en canciones con un tono claramente melancólico/triste (clúster 1) como en aquellas más alegres y energéticas (clúster 3). Sin embargo, las canciones con valores de energía y valencia más cercanos a la media (clúster 2) presentan una popularidad mucho menor. Esto sugiere una preferencia de las personas por estímulos musicales situados en los extremos emocionales.

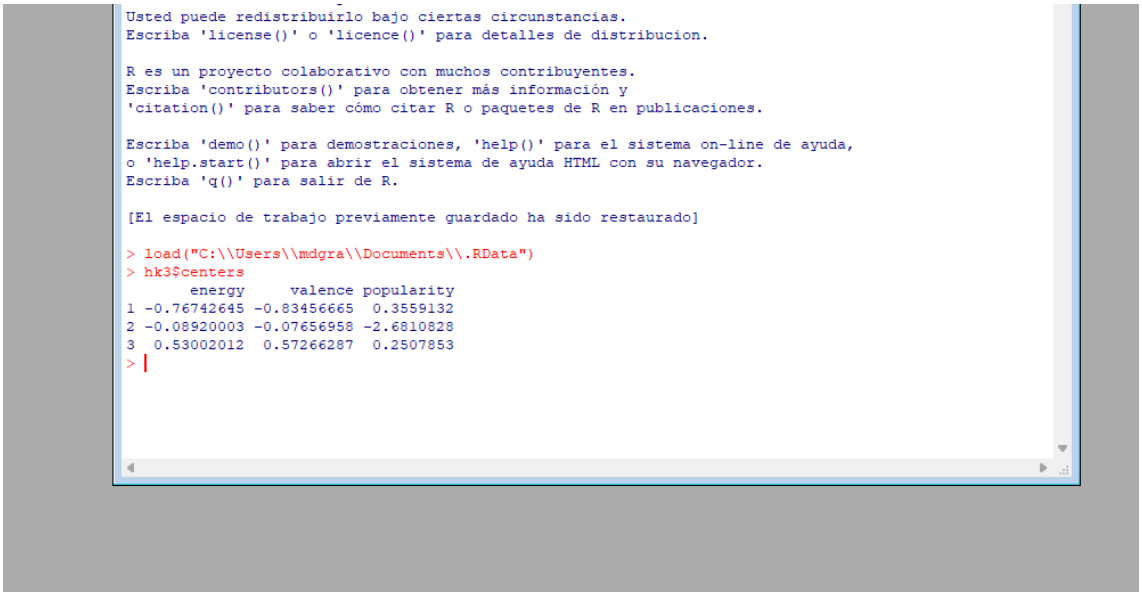


PAM

Para continuar desarrollando esta hipótesis, que detallaré más adelante en el apartado de análisis y conclusiones, apliqué un segundo algoritmo con las mismas variables (energy, valence y popularity), tal como indica la rúbrica.

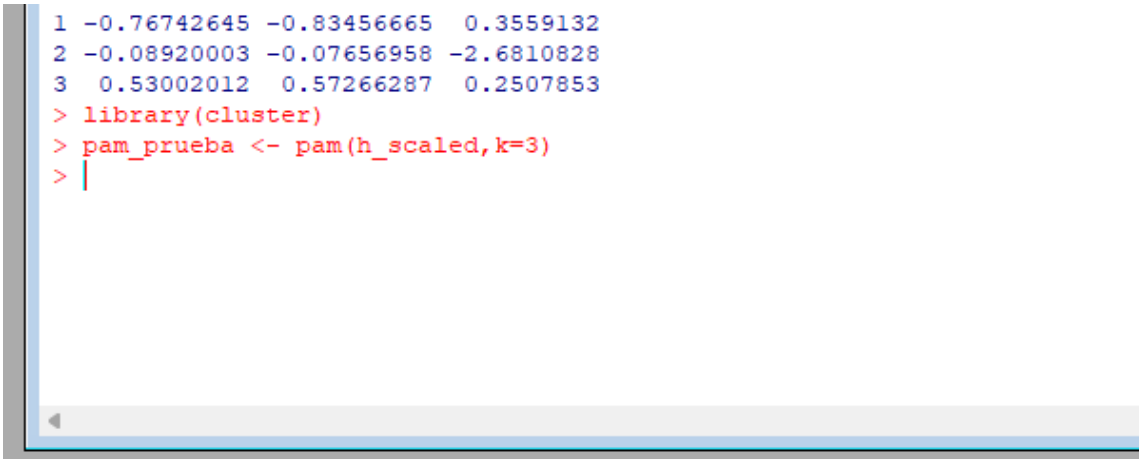
En este caso utilicé el método PAM (Partitioning Around Medoids). A diferencia de k-means, PAM utiliza como centro del clúster un medoide, es decir, una observación real del dataset que actúa como representante del grupo. Esto evita que el centro sea un valor promedio abstracto (como ocurre en k-means) y hace que el método sea más robusto ante valores extremos o ruido.

Para recapitular, estos fueron los resultados que obtuvimos con k-means, siendo el valor de k=3.

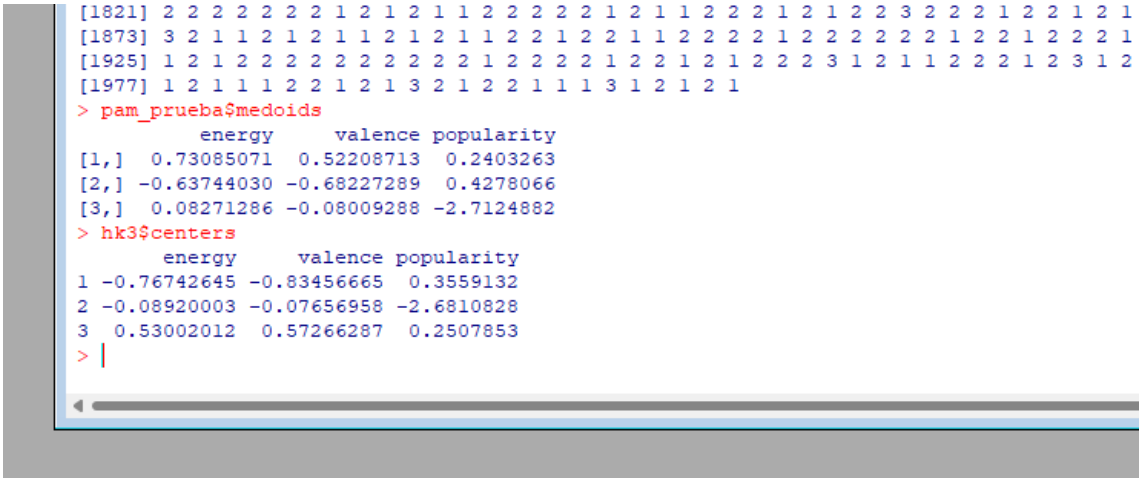


Para poder utilizar el algoritmo PAM, hace falta cargar la librería “cluster”. En R, el paquete cluster no forma parte del núcleo base, pero sí viene preinstalado con la mayoría de las distribuciones de R, incluida la que instala RStudio en Windows, por lo que solo he tenido que cargarlo utilizando el método library(cluster).

Para realizar el algoritmo, la formula me ha recordado a la de k-means. En este caso también la he hecho con k=3 y para que sean tres clústeres y poder comparar los resultados con los obtenidos anteriormente.



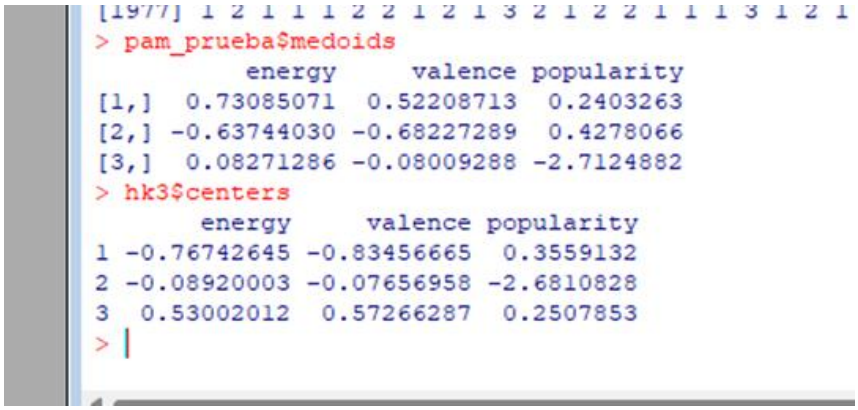
Tras ejecutar dicho algoritmo, lo que me interesaba era ver lo medoides y poder compararlo con los centroides generados con k-means. Podemos comprobar que los resultados obtenidos son muy similares, habiendo de nuevo tres grupos claros, el clúster 1 y 2 con alta popularidad en dos extremos emocionales y de energía diferentes; y el clúster 3 con valores más cercanos a la media, siendo el menos popular.



El motivo por el cual cambié de variables y decidí centrar el análisis en energía, valencia y popularidad, surge de la observación personal y profesional sobre la música y su papel en la regulación emocional. Soy graduada en Enfermería y especialista en Salud Mental vía EIR, y en la práctica clínica la música es una herramienta habitual en grupos psicoterapéuticos, ya que permite activar o modular distintos estados emocionales.

Esto me llevó a plantear la hipótesis de si en la vida cotidiana elegimos las canciones más populares no solo por factores externos (artista, promoción, tendencia), sino también por su carga emocional.

Y efectivamente, podemos observar que aquellos clústeres que tienen valores más extremos tanto en energía como en valencia tienen mayor popularidad que el clúster que tiene valores más cercanos a la media.



Realicé una búsqueda en SciELO para ver si existía respaldo teórico en el ámbito de la psicología. Encontré una revisión sistemática reciente titulada “Respuesta emocional ante la percepción musical. Una revisión sistemática”.

Este artículo señala que “las cualidades de la respuesta emocional a la música dependen principalmente de los aspectos musicales” (tempo, modo, intensidad, etc.) y que estos elementos generan cambios tanto en arousal como en valencia emocional.

Por tanto, mis resultados, que muestran que las canciones emocionalmente más neutras son las menos populares, se alinean con la idea de que la música de carácter moderado provoca respuestas emocionales más débiles.

Estos resultados también invitan a reflexionar sobre la producción musical actual. Si las características emocionales que favorecen la popularidad son

relativamente predecibles, no es extraño que muchas canciones contemporáneas sigan patrones similares.

Esto también puede ser un arma de doble filo: cuanto más se repite una fórmula emocional, menos efecto tiene. El cerebro se acostumbra, y lo que al principio impacta mucho, después deja de generar la misma respuesta. Por eso quizá tantas canciones actuales suenan parecidas: buscan provocar el mismo impacto, pero a costa de una fórmula que cada vez emociona menos.

REFERENCIAS

1. Bobbitt, Z. (2022, 8 septiembre). *How to Use the Elbow Method in R to Find Optimal Clusters*. Statology. https://www.statology.org/elbow-method-in-r/
2. Ibm. (2025, 18 febrero). *¿Qué es el clustering?* IBM. https://www.ibm.com/es-es/think/topics/clustering
3. López, M., Justel, N., & Abrahan, V. D. (2024). Respuesta emocional ante la percepción musical. Una revisión sistemática. *Actualidades En Psicología*, *38*(136), 88-107. https://doi.org/10.15517/ap.v38i136.57422
4. *RPubs - Introducción a los Modelos de Agrupamiento en R*. (s. f.). https://rpubs.com/rdelgado/399475
5. *RPubs - Reescalar, normalizar y variables indicadoras*. (s. f.). https://rpubs.com/ydmarinb/429761
6. *Songs normalize dataset*. (2025, 2 abril). Kaggle. https://www.kaggle.com/datasets/mohammedashraf2004/songs-normalize-dataset?resource=download