



UNIVERSIDAD  
COMPLUTENSE  
MADRID



# MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA

## *Clustering*

Pablo Flores Vidal

Noviembre 2023

## 1. Objetivos y competencias a alcanzar

- Saber obtener matrices de distancias de cualquier tipo de datos originales.
- Conocer los fundamentos de los algoritmos de Análisis de *clustering* jerárquicos: método de Ward, del vecino más cercano, del vecino más alejado, del centroide, y de la media.
- Conocer la importancia de trabajar con los datos estandarizados para el análisis clustering.
- Conocer los fundamentos de los algoritmos de Análisis Cluster No Jerárquico.
- Conocimiento de técnicas que permitan justificar el número de agrupaciones finalmente obtenidas.
- Obtención e interpretación de las agrupaciones generadas por cualquiera de los dos métodos.

## 2. Introducción

La mayoría de los algoritmos que “están de moda” actualmente (2023) dentro de lo que se engloba como machine learning son de los denominados "supervisados"(supervised algorithms). Y se denominan así porque el aprendizaje está “supervisado” en el sentido de que los datos han sido previamente etiquetados (supuestamente por una persona, de ahí lo de “supervisados”). Imaginemos ahora otra línea de métodos clasificatorios donde nos podríamos encontrar con unos algoritmos en los que no se conocen las etiquetas (o se prescinde de ellas), y en los que nos va a interesar en este caso crear agrupaciones a partir de los datos, más que clasificarlos en unas categorías más o menos preestablecidas. Estas agrupaciones (*cluster*) pueden tener mucha utilidad cuando no se conocen a priori ciertas características que hacen diferenciarse a unos grupos de otros. Y esta búsqueda de diferencias es parte de la filosofía que perseguimos en la Ciencia de los datos, aunque como veremos para encontrar buenas agrupaciones no es suficiente buscar diferencias entre ellos, sino también similitudes.

Pongamos un ejemplo de la utilidad potencial de crear grupos para demostrar la importancia de esta técnica. Supongamos que representamos dos variables de un conjunto de datos sobre medidas de pingüinos mediante un diagrama de dispersión y obtenemos lo que podemos ver en la Figura 1.

Las variables involucran la longitud y anchura del pico de estos. Obviamente esperaríamos encontrar más correlación reflejada en un patrón más claro. Sin embargo, no sólo no es así, sino que parece haber cierta incorrelación (recordemos que si están muy dispersos equivale a incorrelación). ¿A qué se debe esto?

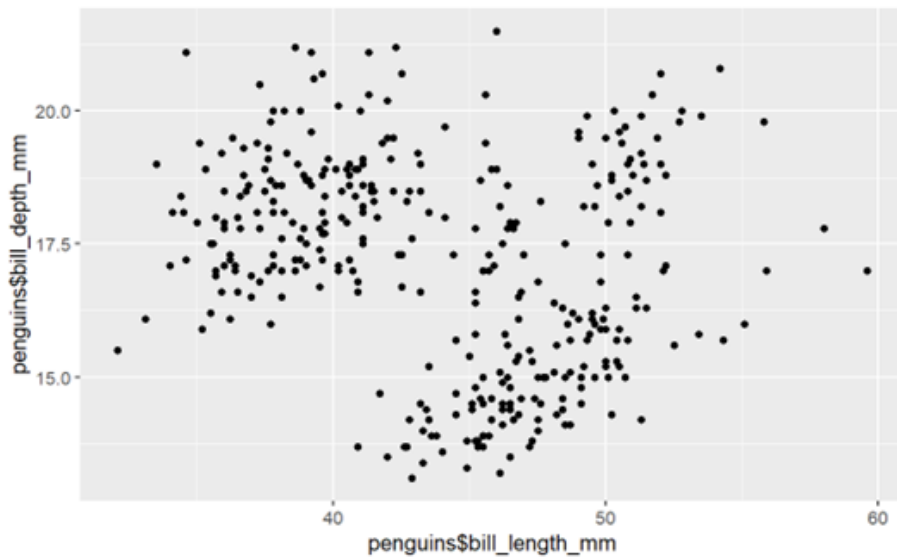


Figura 1: Distribución de los pingüinos

Para responder a la pregunta uno debe fijarse en que no es que no exista apenas patrón en los datos, sino que en realidad puede que haya diferentes “tipos” de datos o grupos. Estos grupos en realidad serían potenciales diferentes subespecies (pues al tener características diferentes presentarían diferentes patrones). Y de hecho, eso es lo que ocurre, sólo que no lo podemos saber a través de esas dos variables. Las diferentes especies de pingüinos no se reflejan en las dos variables representadas directamente, pero al haber no uno, sino dos supuestos patrones (nubes de datos con forma diferenciada) como en la Figura 2 todo parece cobrar más sentido.

Lo que hemos hecho es identificar dos posibles agrupaciones que no tienen porque reflejar una clasificación “real” (de hecho, en realidad no hay dos poblaciones, sino tres poblaciones o especies diferentes de pingüinos, pero eso es “otra historia”), sino meros grupos potenciales de individuos que podrían estar asociados en un patrón diferenciado al resto, hablando en plata, diríamos que hemos encontrado dos posibles cluster (aunque sea a ojo). Esto, pero de forma sofisticada y apoyándonos en métodos más exactos es básicamente lo que vamos a hacer con las técnicas de *clustering*, solo que lo haremos empleando más variables (*features*), lo que nos permitirá extraer posibles grupos con mucha mayor capacidad de penetración en la naturaleza de los datos (imaginemos esto que hemos hecho ahora, pero a nivel n-dimensional y con métodos más exactos y optimizados).

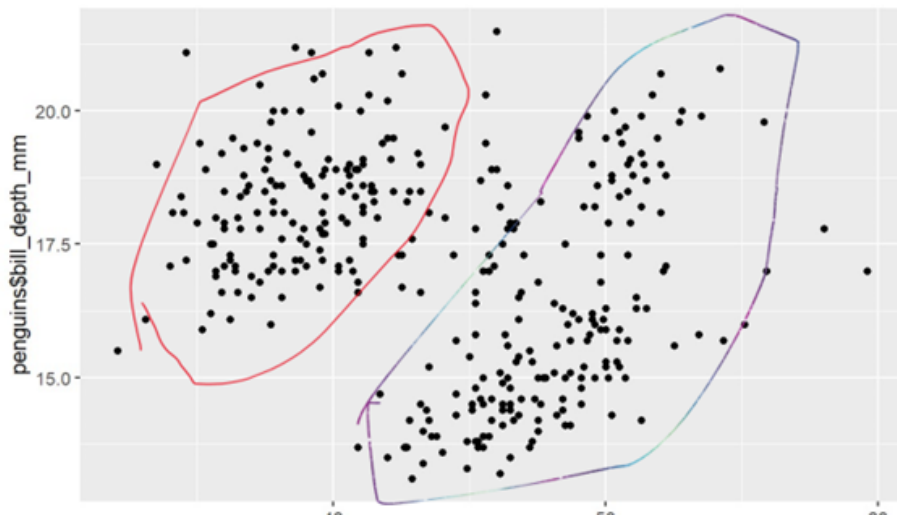


Figura 2: 2 posibles agrupaciones ‘a ojo’ de los pingüinos

Ahora imaginemos otras tantas situaciones que se pueden dar en contextos cotidianos:

- El departamento de marketing de una empresa va a lanzar una campaña publicitaria sobre un nuevo producto. Para ello, desea tener a sus potenciales clientes agrupados según sus necesidades en los distintos aspectos de dicho producto.
- Se desea agrupar a los clientes de un banco para determinar diferentes perfiles a los que se les puede conceder un préstamo.
- En estudios genéticos en los que es habitual encontrar grupos de individuos con carga genética similar.

En todos los casos anteriores, nos encontramos con situaciones similares, en el sentido de que **nuestro objetivo es formar grupos de individuos con características similares con respecto a las puntuaciones que presentan en determinadas variables.**

El Análisis de clustering (o simplemente *clustering*) es la técnica de Análisis Multivariante cuyo propósito es, a partir de un conjunto de individuos, crear “buenos” agrupamientos (clusters).

Por “buenos” agrupamientos entenderemos aquellos que cumplan al menos lo siguiente:

- Los individuos de cada grupo deben ser lo más parecidos que sea posible (homogeneidad interna).
- Los grupos deben ser lo más diferentes entre sí posible (heterogeneidad entre grupos).
- Generalmente los grupos obtenidos son mutuamente excluyentes (cada observación pertenece a un solo grupo).

---

*En las técnicas de clustering lo que nos va a interesar es crear agrupaciones a partir de los datos y no clasificarlos en unas categorías claramente preestablecidas,*

---

La forma más simple de obtener clústeres, es la simple inspección gráfica cruzando variables (como en el ejemplo anterior de los pingüinos), agrupando las observaciones que se encuentren más próximas entre sí, o bien, tras su debido tratamiento, con la representación de las observaciones en planos factoriales. No obstante, en la práctica serán muchas las observaciones a clasificar por lo que será difícil su representación, y especialmente ya no sólo por la cantidad, sino por la  $n$ -dimensionalidad (cuando  $k \gg 2$ ). Otro asunto, que no es baladí (y que acabará redundando en los grupos encontrados), es que hay que tener en cuenta las muchas formas a la hora de medir la proximidad entre dos observaciones, y finalmente, establecer los diferentes procedimientos de agrupación. Por lo tanto, se puede obtener una gama amplia de posibles resultados.

Como en la mayoría de las técnicas multivariantes la representatividad de la muestra es tremendamente importante, como también la existencia o no de multicolinealidad. La diferencia fundamental con respecto a otras técnicas de clasificación es que no se conoce de antemano el número de grupos en los que se divide a la población, ni el valor de la variable que identifica cada grupo.

---

*La diferencia fundamental con respecto a otras técnicas de clasificación es que no se conoce de antemano el número de grupos en los que se divide a la población, ni el valor de la variable que identifica cada grupo.*

---

Ejemplo: Para ilustrar con el problema anterior de clasificación de los pingüinos el tema del no conocimiento previo del grupo al que pertenecen pensemos que, ¡realmente no hay 2 grupos, sino 3!

### 3. ¿Cómo se hace un Análisis de *clustering*?

Como se indicó anteriormente, nuestro objetivo es formar grupos de datos homogéneos. Para lograr este objetivo necesitamos evaluar el parecido entre dos observaciones. Utilizaremos las medidas de distancia. Una vez que conocemos la manera de determinar la proximidad o similitud entre observaciones debemos determinar la manera de agruparlas. **Existen dos procedimientos de agrupación: Jerárquicos y no Jerárquicos.** En los primeros no se conoce de antemano el número de cluster o grupos a formar. En los procedimientos no Jerárquicos, el número de grupos tiene que ser establecido de antemano (y establecer este número puede requerir la aplicación de otros métodos a su vez). Por esta razón, **en la práctica se utiliza primero un modelo jerárquico y cuando ya tenemos una idea del número de cluster adecuado se aplica un modelo no jerárquico.**

En tercer lugar, debemos elegir un método de agrupación dentro del procedimiento elegido. A continuación, debemos tomar una decisión acerca del número óptimo de cluster. Por último, la solución elegida en el paso anterior debe ser interpretada.

**Example 3.1** *Ejemplo de un Análisis clustering sobre los países según Esperanza de Vida. Estamos interesados en una clasificación en grupos de países según su esperanza de vida a diferentes edades, según sexo.*

```
import pandas as pd

# Read the Excel file into a DataFrame
df = pd.read_excel('Esperanzadevida.xlsx')

# View the first few rows of the DataFrame
df.head()
```

| index | PAIS       | m0 | m25 | m50 | m75 | w0 | w25 | w50 | w75 |
|-------|------------|----|-----|-----|-----|----|-----|-----|-----|
| 0     | Algeria    | 63 | 51  | 30  | 13  | 67 | 54  | 34  | 15  |
| 1     | Cameroon   | 34 | 29  | 13  | 5   | 38 | 32  | 17  | 6   |
| 2     | Madagascar | 38 | 30  | 17  | 7   | 38 | 34  | 20  | 7   |
| 3     | Mauritius  | 59 | 42  | 20  | 6   | 64 | 46  | 25  | 8   |
| 4     | Reunion    | 56 | 38  | 18  | 7   | 62 | 46  | 25  | 10  |

Figura 3: Un vistazo a los datos sobre Esperanza de vida

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set 'PAIS' as the index df = df.set_index('PAIS')

# Create the heatmap
sns.clustermap(df, cmap='coolwarm', annot=True)
```

```
plt.title('Heatmap with Pais as Categorical Variable')
plt.xlabel('Esperanza de vida a esa edad para hombres (m) y
          mujeres (w)')
plt.ylabel('Pais')

# Display the plot
plt.show()
```

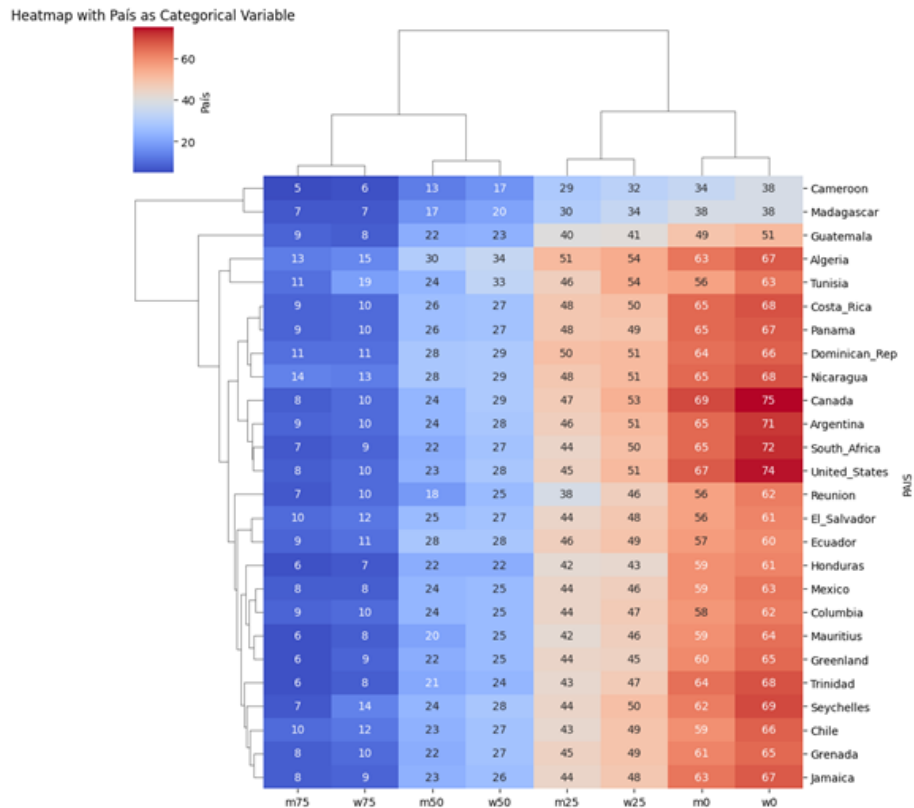


Figura 4: clustermap

*clustermap()* crea un mapa de calor agrupado reorganizando las filas y columnas según la agrupación jerárquica. Puede ayudar a revelar patrones y relaciones dentro de los datos. Pero también nos permite encontrar parecidos entre las variables, mostrándonos además un dendrograma de agrupación posible tanto entre individuos como entre variables.

### 3.1. Medidas de distancias entre observaciones

Para medir la semejanza entre dos observaciones se utilizan medidas de distancia (dos observaciones serán más parecidas cuanto menor distancia haya entre

ellas) o de similitud (dos observaciones serán más parecidas cuanto mayor sea su similitud). Suponiendo que cada observación recoge el valor de  $p$  variables en un individuo, podemos denotar la observación  $i$  por  $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ . Se pueden definir las siguientes medidas de distancia para variables numéricas, las cuales, si bien son de las más utilizadas, no son las únicas. La más utilizada es la **distancia Euclídea**:

$$\sqrt{\sum_{i=1}^p (x_i - y_i)^2} \quad (1)$$

También se suele recurrir a la fórmula más general conocida como **distancia Minkowski**:

$$\sqrt[p]{\sum_{i=1}^p |x_i - y_i|^p} \quad (2)$$

Para el caso  $p = 2$  obtendríamos la distancia Euclídea, y para  $p = 1$  tendríamos la conocida como **distancia Manhattan**.

---

*Las distancias más empleadas son la Euclídea y la ‘Manhattan’, aunque ambas son casos particulares de la de Minkowski.*

---

### 3.2. Medidas de distancia entre variables

Además de poder hacer grupos de observaciones, puede ser de interés hacer grupos de variables cuyo comportamiento sobre el conjunto de individuos sea similar o presenten correlación entre ellas. Para ello, se definen medias de distancia entre variables basadas en los diferentes coeficientes de correlación. Un ejemplo es la **distancia de correlación de Pearson**:

$$\text{Distancia de Correlación de Pearson} = 1 - \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (3)$$

En esta fórmula:

- $n$  es el número de observaciones en las muestras  $x$  e  $y$ .
- $x_i$  y  $y_i$  son las observaciones individuales en las muestras.
- $\bar{x}$  y  $\bar{y}$  representan las medias de las muestras  $x$  e  $y$ , respectivamente.

Esta fórmula calcula la distancia de correlación de Pearson entre dos muestras  $x$  e  $y$ , que es una medida de la similitud entre las dos muestras en función de la correlación entre sus valores.



Otra distancia entre variables es la calculada mediante la **distancia de correlación de Spearman**. El coeficiente de correlación de rangos de Spearman se utiliza cuando los datos son ordinales o cuando no se cumplen los supuestos de la correlación de Pearson (linealidad y homocedasticidad). A diferencia de la correlación de Pearson, la correlación de Spearman se basa en las clasificaciones de los datos en lugar de en los valores de los datos sin procesar. Esto lo hace menos sensible a los valores atípicos y es apropiado para variables que pueden tener una relación monótona pero no necesariamente lineal.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (4)$$

Donde:

- $\rho$  es el coeficiente de correlación de Spearman.
- $d_i$  son las diferencias entre los rangos de las observaciones de dos conjuntos de datos emparejados.
- $n$  es el número de observaciones en los conjuntos de datos emparejados.

Por último, tendríamos la **distancia de correlación de Kendall** (llamado también coeficiente de concordancia de Kendall), que se utiliza para medir la concordancia entre dos conjuntos de datos. A diferencia de la correlación de Pearson y la correlación de Spearman, la distancia de Kendall se centra en la concordancia o discordancia entre los pares de observaciones, sin considerar la magnitud de las diferencias entre las observaciones.

$$\text{Distancia de Kendall} = 1 - \frac{2 \times N^o \text{ de pares concordantes} - N^o \text{ total de pares}}{n(n-1)/2} \quad (5)$$

Donde:

- ‘Número de pares concordantes’ es la cantidad de pares de observaciones que mantienen la misma relación de orden en ambos conjuntos de datos.
- ‘Número total de pares’ es el número total de pares de observaciones.
- ‘n’ es el número de observaciones en los conjuntos de datos.

El resultado del coeficiente de asociación de Kendall es un valor que varía entre -1 y 1, donde:

- Un valor de -1 indica una falta de concordancia completa entre los conjuntos de datos.
- Un valor de 1 indica una concordancia perfecta.
- Un valor de 0 sugiere que no hay una concordancia ni discordancia sistemática.

**Example 3.2** *Ejemplo Matriz de distancias: Calculamos las distancias Euclídeas entre las observaciones (con los valores sin estandarizar):*

```
from scipy.spatial import distance

# Calculate the pairwise Euclidean distances
distance_matrix = distance.cdist(df, df, 'euclidean')

# The distance_matrix is a 2D array containing the Euclidean
# distances between all pairs of observations.

distance_small = distance_matrix[:5, :5]
# Index are added to the distance matrix
distance_small = pd.DataFrame(distance_small, index=df.index
                              [:5], columns=df.index[:5])

distance_small_rounded = distance_small.round(2)
print("Distance Matrix fr:"distance_small_rounded)
```

Cuadro 1: Matriz de distancia entre países

| Country    | Algeria | Cameroon | Madagascar | Mauritius | Reunion |
|------------|---------|----------|------------|-----------|---------|
| Algeria    | 0.00    | 58.12    | 52.70      | 21.24     | 24.39   |
| Cameroon   | 58.12   | 0.00     | 7.14       | 42.39     | 38.20   |
| Madagascar | 52.70   | 7.14     | 0.00       | 38.13     | 34.00   |
| Mauritius  | 21.24   | 42.39    | 38.13      | 0.00      | 6.16    |
| Reunion    | 24.39   | 38.20    | 34.00      | 6.16      | 0.00    |

Por si se desea profundizar, la función anterior *cdist()* admite las siguientes distancias: *Euclidean*, *Manhattan*, *Minkowski*, *Chebyshev*, *Cosine*, *Hamming*, *Jaccard*, *Correlation*, *Mahalanobis*, *Canberra*.

Representemos ahora mediante escalas de color la distancia entre todas las observaciones, esto nos ayudará a detectar grupos de observaciones con grandes distancias entre sí.

```
plt.figure(figsize=(8, 6))
sns.heatmap(distance_matrix, annot=False, cmap="YlGnBu", fmt="
            .1f")
plt.show()
```

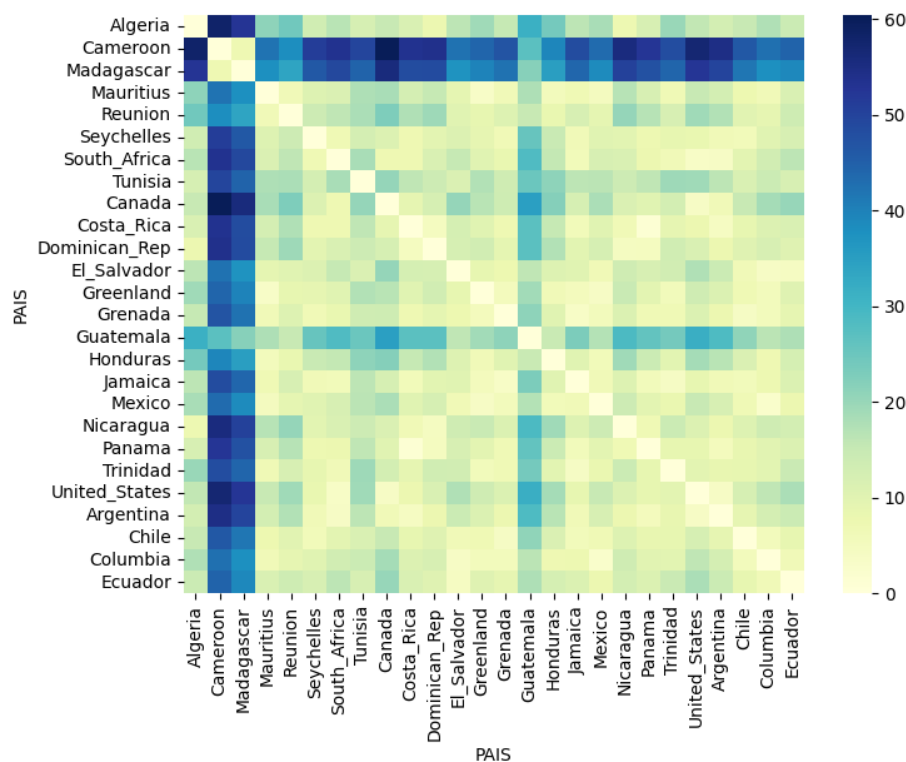


Figura 5: Mapa de calor

Como podemos observar en la Figura 5, hay dos países, Camerún y Madagascar, que tienen una distancia muy superior con el resto de los países. Existen métodos de combinatoria que nos permiten reorganizar los individuos para intentar poner juntos a los más parecidos (aquellos con menor distancia entre sí). Estos métodos nos permiten dibujar mapas de calor como el anterior, pero con la posibilidad de aplicar métodos de reordenación para conseguir una visualización de los grupos óptima de cara a reconocer patrones y similitudes en los datos.

```
# Perform hierarchical clustering to get the linkage matrix
linkage = sns.clustermap(df_distance, cmap="YlGnBu", fmt=".1f",
                        annot=False, method='average').dendrogram_row.linkage

# Reorder the data based on the hierarchical clustering
order = pd.DataFrame(linkage, columns=['cluster_1', 'cluster_2',
                                     'distance', 'new_count']).index
reordered_data = df.reindex(index=order, columns=order)

# Optionally, you can add color bar
sns.heatmap(reordered_data, cmap="YlGnBu", fmt=".1f", cbar=
```

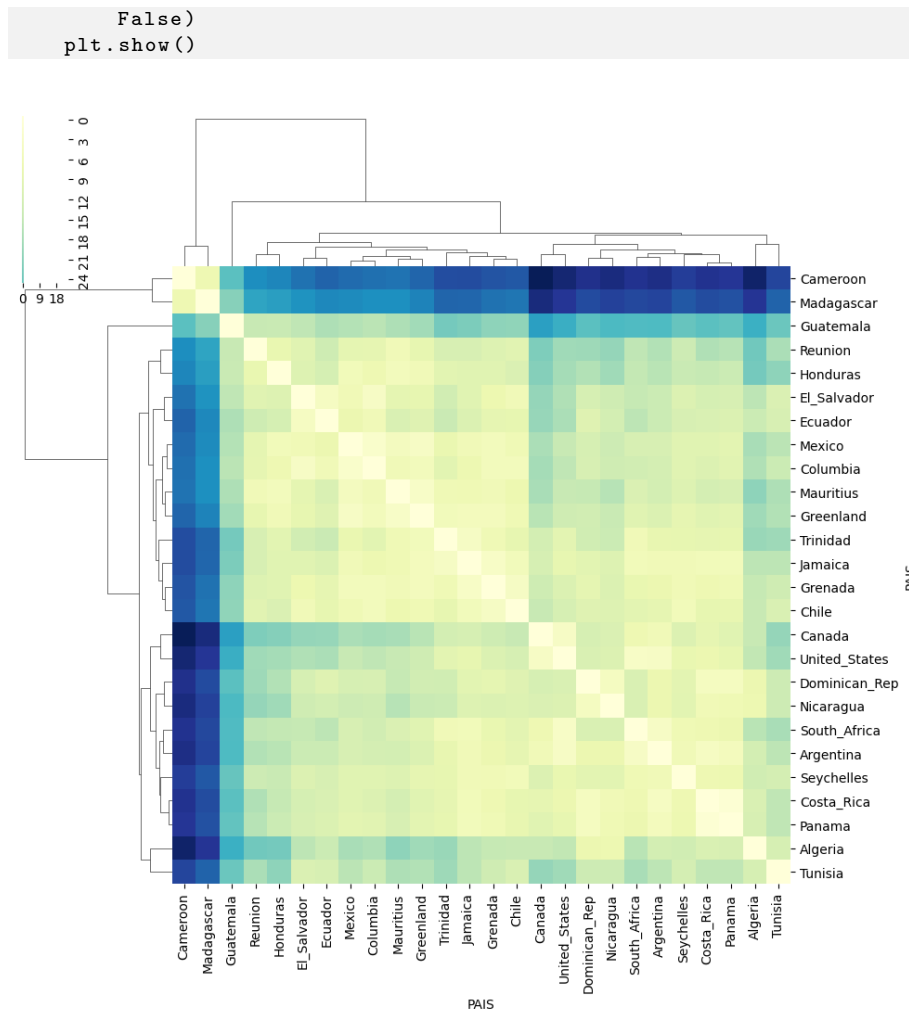


Figura 6: CLustering map de calor

Para que la distancia entre individuos no dependa de las unidades en las que están medidas las variables y refleje mejor las relaciones entre individuos es conveniente estandarizar las variables. Recordamos que estandarizar es restar su media y dividir por su desviación estándar para conseguir que todas las variables tengan media cero y desviación típica uno:

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \quad (6)$$

```
from sklearn.preprocessing import StandardScaler
```

```
# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the DataFrame to standardize the columns
df_std = pd.DataFrame(scaler.fit_transform(df), columns=df.
                      columns)
```

Ahora, las columnas del conjunto de datos estandarizado tienen una media de aproximadamente 0 y una desviación estándar de aproximadamente 1, lo que se conoce como puntuación z. Esto ayuda a que las variables sean comparables y puede resultar útil para diversas tareas de análisis y modelado de datos.

```
# Calculate the pairwise Euclidean distances
distance_std = distance.cdist(df_std, df_std, "euclidean")
print(distance_std[:5, :5].round(2))
```

|   | 1     | 2     | 3     | 4    |      |
|---|-------|-------|-------|------|------|
| 1 | 0,00  | 11,77 | 10,22 | 6,27 | 6,4  |
| 2 | 11,77 | 0,00  | 1,87  | 7,21 | 6,73 |
| 3 | 10,22 | 1,87  | 0,00  | 6,27 | 5,75 |
| 4 | 6,27  | 7,21  | 6,27  | 0,00 | 1,39 |
| 5 | 6,4   | 6,73  | 5,75  | 1,39 | 0,00 |

Podemos ver en la Tabla 3.2 que las distancias son ahora menores al estar estandarizadas las variables. Podemos intentar de nuevo visualizar el mapa de calor de las distancias para datos estandarizados, ordenando los países de forma que estén próximos los más parecidos. Conviene preguntarse aquí: ¿cuál es la diferencia respecto al mapa de calor anterior?

```
plt.figure(figsize=(8, 6))
df_std_distance = pd.DataFrame(distance_std, index = df_std.index,
                               columns = df.index)
sns.heatmap(df_std_distance, annot=False, cmap="YlGnBu", fmt=".1f")
plt.show()
```

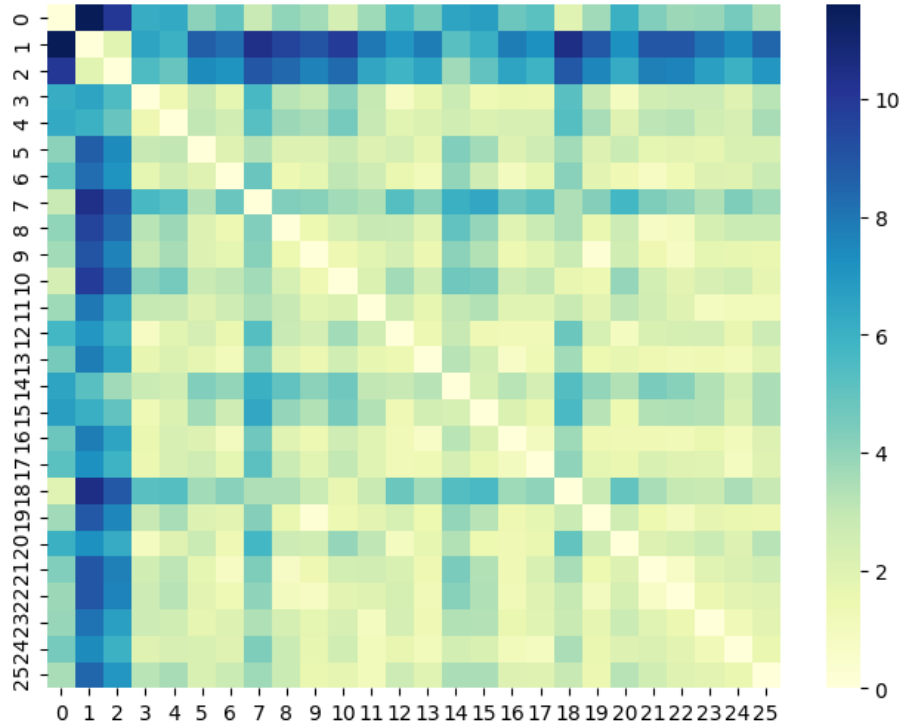


Figura 7: Matriz de distancias visual entre observaciones

#### 4. Algoritmos de clasificación jerárquica

Los métodos de clasificación jerárquica no producen una clasificación en un número determinado de cluster, sino que configuran grupos con estructura arborescente de forma que los cluster de niveles más 'bajos' (que empiezan siendo observaciones individuales) van siendo englobados en otros de niveles 'superiores' (que abarcan más observaciones).

Los pasos a seguir para realizar un *Clustering* Jerárquico son los siguientes:

1. Se parte de tantos grupos como observaciones
2. Se genera una matriz de dimensión  $n \times n$  (simétrica) que indique las distancias entre todos los pares de observaciones (esta distancia debe haber sido definida con anterioridad).
3. A continuación, se agrupan las dos observaciones más próximas. Con esto, el número de cluster existentes es uno menos que en el paso anterior.
4. Se vuelve a obtener una matriz de distancias con los cluster formados en el paso 3. Observa que para obtener esta nueva matriz, se necesita elegir un método de cálculo para la distancia entre cluster.

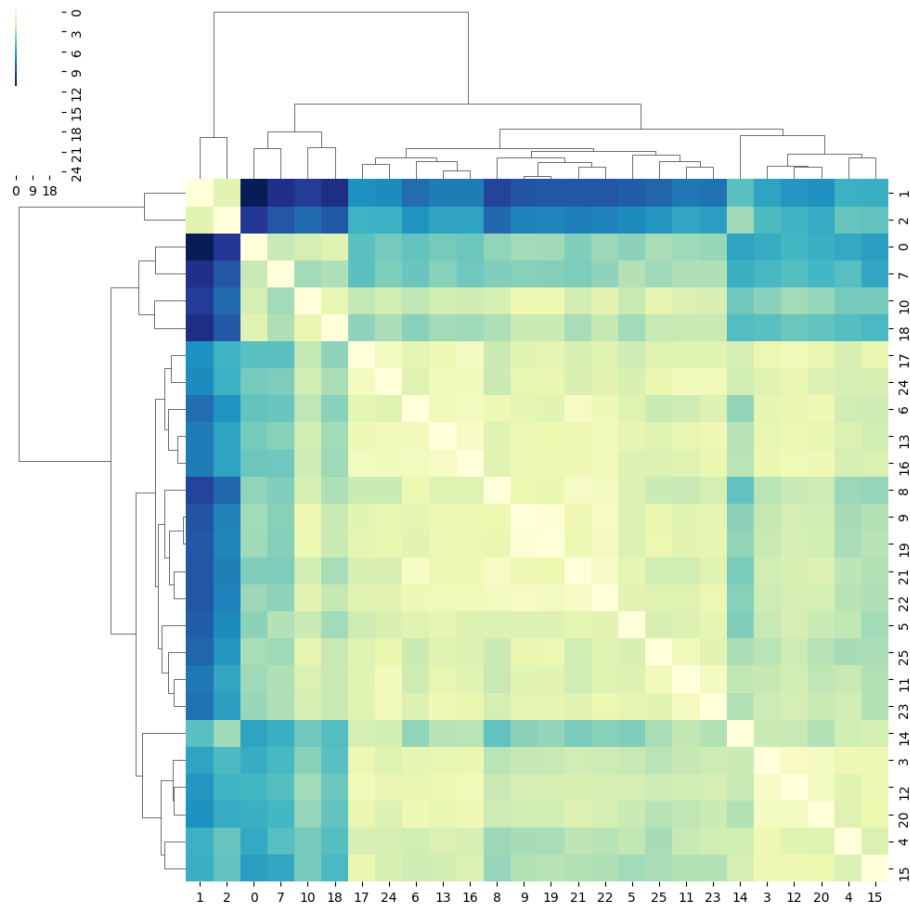


Figura 8: Matriz de distancias visual entre observaciones

5. A continuación, se vuelven a realizar las agrupaciones.
6. El proceso continúa hasta que todas las observaciones están agrupadas en un solo cluster.

Observemos que es necesario definir con claridad, tanto la distancia entre observaciones, como la distancia entre clústeres o grupos de observaciones. Por lo tanto, bajo este esquema y con un mismo conjunto de datos, variando esas dos definiciones se podrán obtener múltiples clasificaciones diferenciadas.

#### 4.1. Evaluando la distancia entre cluster o grupos de observaciones

Supongamos que tenemos una agrupación denominada  $C_k$  y otra agrupación  $C_l$ , los algoritmos más comunes son:

- **Enlace simple o del vecino más cercano:** Se establece la siguiente distancia:

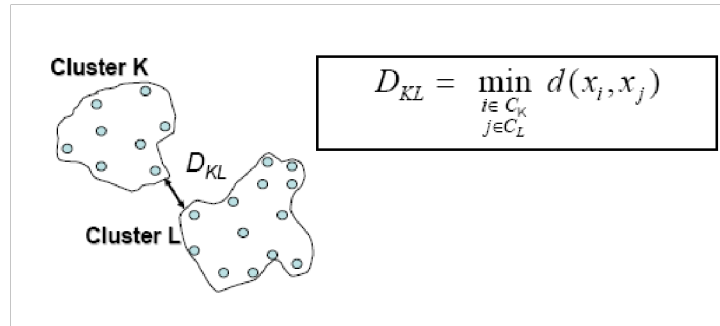


Figura 9: Distancia entre cluster: Vecino más cercano

Es decir, la distancia entre dos grupos es la distancia entre las dos observaciones más cercanas, pertenecientes cada una de ellas a un grupo distinto. Tienden a crear grupos con muchas observaciones.

- **Enlace completo o del vecino más lejano:** La distancia elegida entre dos cluster será:

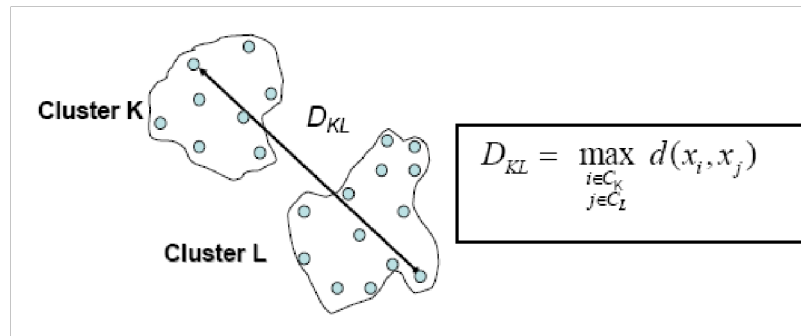


Figura 10: Distancia entre cluster: Vecino más lejano

Luego, la distancia entre dos grupos, es la distancia entre las dos observaciones más alejadas de cada uno de los dos grupos. Con este método, los grupos formados son más compactos que los obtenidos con el método del vecino más próximo.



- **Enlace medio:** La distancia entre dos agrupaciones es la distancia media entre una observación de la agrupación  $C_k$  y otra observación de la agrupación  $C_l$ . Los grupos así formados tienen varianza similar y además pequeña.

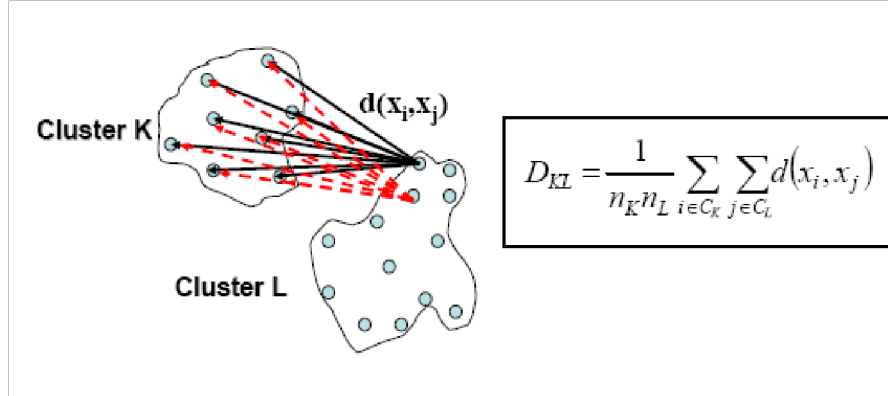


Figura 11: Distancia entre cluster: enlace medio

- **Distancia entre centroides:** La distancia entre grupos será la distancia entre los centroides, que representarán el vector medio obtenido en las p variables para todos los individuos que formen parte del grupo.

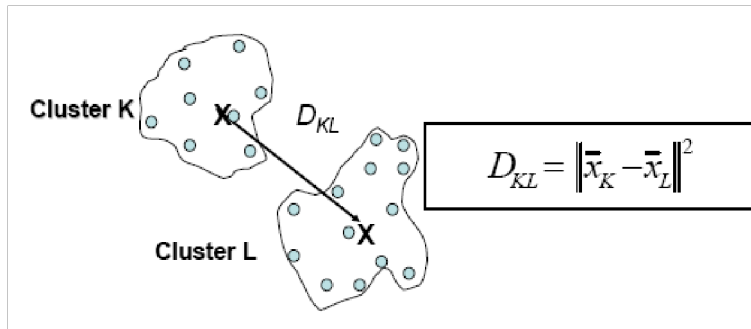


Figura 12: Distancia entre centroides

- **Distancia de Ward o de la mínima varianza:** Este método, entre todas las uniones de cluster posibles en cada nivel, selecciona aquella unión que minimiza la variabilidad interna de los cluster resultantes. Mide la disimilitud entre dos grupos basándose en el aumento en la suma de distancias euclidianas al cuadrado cuando se fusionan.

$$d_{\text{Ward}}(C_i, C_j) = \frac{n_i n_j}{n_i + n_j} \sum_{x_{ij} \in C_i \cup C_j} \|x_{ij} - \mu_{C_i \cup C_j}\|^2$$

En esta fórmula:

- $d_{\text{Ward}}(C_i, C_j)$  es la distancia de Ward entre los grupos  $C_i$  y  $C_j$ .
- $n_i$  y  $n_j$  representan el número de observaciones en los grupos  $C_i$  y  $C_j$ , respectivamente.
- $x_{ij}$  son las observaciones en el grupo resultado de unir los cluster  $C_i$  y  $C_j$ .
- $\mu_{C_i \cup C_j}$  es la media (centroide) de las observaciones en la unión de los conglomerados  $C_i$  y  $C_j$ .
- $|x_{ij} - \mu_{C_i \cup C_j}|^2$  representa la distancia euclidiana al cuadrado entre la observación  $x_{ij}$  y el centroide del grupo fusionado  $C_i \cup C_j$ .

La distancia de Ward se usa comúnmente en agrupaciones jerárquicas aglomerativas para determinar qué grupos deben fusionarse para crear una jerarquía. Algunas de sus ventajas son:

- Minimización de la varianza: el método Ward tiene como objetivo minimizar la varianza dentro de cada grupo. Evalúa el aumento de la varianza que resulta de la fusión de conglomerados y busca minimizar este aumento. Esto puede dar lugar a agrupaciones más compactas y homogéneas.
- Equilibrio de tamaños de conglomerados: el método de Ward tiende a producir conglomerados de tamaños aproximadamente iguales. Esto puede resultar ventajoso en determinadas aplicaciones en las que es deseable tener grupos de tamaños similares.
- Sensible a la forma de los cúmulos: el método Ward es relativamente sensible a la forma de los cúmulos y, a menudo, funciona bien cuando los cúmulos son globulares o elípticos. Esto lo hace adecuado para conjuntos de datos donde los grupos subyacentes tienen una estructura determinada.
- Estructura jerárquica: el método Ward produce naturalmente una estructura de agrupamiento jerárquica, representada en un dendrograma. Esto puede resultar útil para visualizar relaciones entre clústeres en diferentes niveles de granularidad.
- Menos sensible a las condiciones iniciales: en comparación con otros métodos de agrupación, el método de Ward es menos sensible a las condiciones iniciales. Puede ser más estable en diferentes ejecuciones.

Tras ver las distancias anteriores, uno podría preguntarse: **¿Cuál es la distancia entre clusters más adecuada?**

No existe una respuesta exacta a esta pregunta (aunque hayamos resaltado las ventajas de la de Ward). Una posibilidad es probar varias distancias y tomar una decisión en función de los resultados que se obtengan. Si varios métodos nos dan agrupaciones similares, se puede pensar que existe

|                                   |     |         |       |    |    |    |  |                                 |       |     |       |    |    |    |  |
|-----------------------------------|-----|---------|-------|----|----|----|--|---------------------------------|-------|-----|-------|----|----|----|--|
|                                   | 1   | 2       | 3     | 4  | 5  | 6  |  |                                 | 1     | 2   | 3,5   | 4  | 6  |    |  |
| 1                                 |     | 31      | 23    | 32 | 26 | 25 |  |                                 | 1     |     | 31    | 23 | 32 | 25 |  |
| 2                                 |     |         | 34    | 21 | 36 | 28 |  |                                 | 2     |     |       | 34 | 21 | 28 |  |
| 3                                 |     |         |       | 31 | 4  | 7  |  |                                 | 3,5   |     |       | 27 | 7  |    |  |
| 4                                 |     |         |       |    | 27 | 28 |  |                                 | 4     |     |       |    |    | 28 |  |
| 5                                 |     |         |       |    |    | 9  |  |                                 | 6     |     |       |    |    |    |  |
| 6                                 |     |         |       |    |    |    |  |                                 |       |     |       |    |    |    |  |
| $C_1 = \{[1],[2],[3,5],[4],[6]\}$ |     |         |       |    |    |    |  | $C_2 = \{[1],[2],[3,5,6],[4]\}$ |       |     |       |    |    |    |  |
|                                   | 1   | 2       | 3,5,6 | 4  |    |    |  |                                 | 1     | 2,4 | 3,5,6 |    |    |    |  |
| 1                                 |     |         | 31    | 23 | 32 |    |  |                                 | 1     |     | 31    | 23 |    |    |  |
| 2                                 |     |         |       | 28 | 21 |    |  |                                 | 2,4   |     |       | 27 |    |    |  |
| 3,5,6                             |     |         |       |    | 27 |    |  |                                 | 3,5,6 |     |       |    |    |    |  |
| 4                                 |     |         |       |    |    |    |  |                                 |       |     |       |    |    |    |  |
| $C_3 = \{[1],[2,4],[3,5,6]\}$     |     |         |       |    |    |    |  | $C_4 = \{[2,4],[1,3,5,6]\}$     |       |     |       |    |    |    |  |
|                                   | 2,4 | 1,3,5,6 |       |    |    |    |  |                                 |       |     |       |    |    |    |  |
| 2,4                               |     |         | 27    |    |    |    |  |                                 |       |     |       |    |    |    |  |
| 1,3,5,6                           |     |         |       |    |    |    |  |                                 |       |     |       |    |    |    |  |
| $C_5 = \{1,2,3,4,5,6\}$           |     |         |       |    |    |    |  |                                 |       |     |       |    |    |    |  |

Figura 13: Distancia entre cluster

una forma natural de formarse grupos de observaciones. En la práctica el método más utilizado es el de Ward, porque nos asegura la máxima homogeneidad dentro de los cluster.

**Example 4.1** *Llevamos a cabo un ejemplo de agrupación mediante un algoritmo jerárquico, sobre un conjunto de seis observaciones de las que presentamos sus distancias. Utilizamos como distancia entre cluster la distancia simple o del vecino más cercano. El proceso a seguir, es el indicado en la Figura 13.*

## 4.2. Resultados del clúster jerárquico: El Dendrograma

Es frecuente presentar los resultados del análisis clúster jerárquico con un gráfico como el de la Figura 15. Tiene la estructura de un árbol que permite plasmar el proceso de aglomeración y composición de grupos (para cualquier número de ellos) junto con la distancia entre cada dos grupos unidos en una gráfica. Este tipo de diagrama contiene ramas que unen puntos y muestran el orden en que se asignan las observaciones a los agrupamientos. Las longitudes de las ramas son proporcionales a las distancias entre los puntos/agrupamientos. Este diagrama depende de la distancia entre elementos y entre cluster utilizada, y nos puede ayudar a determinar en qué momento del proceso de agrupación nos deberemos detener.

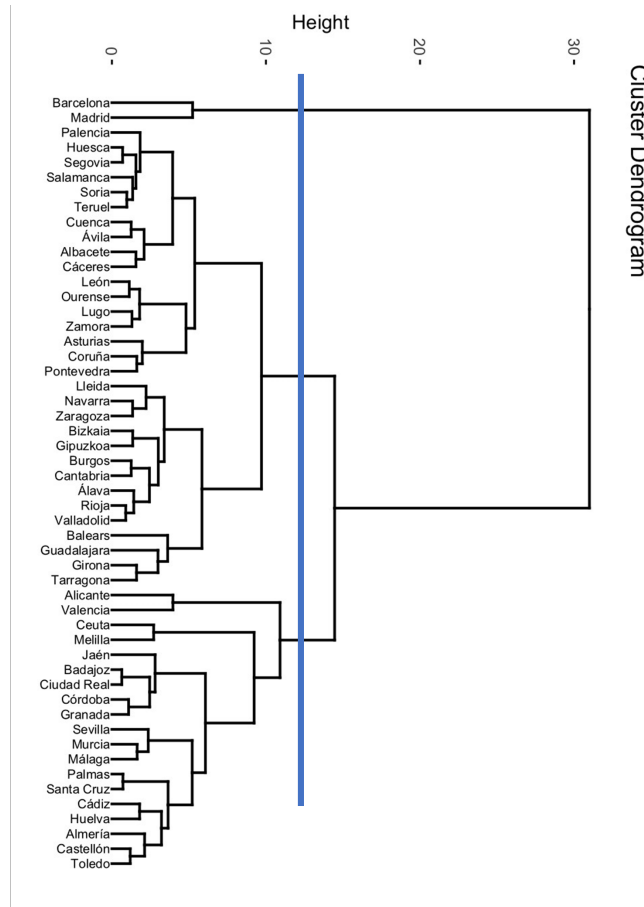


Figura 14: Dendrograma

Según dónde cortemos vemos la estructura de  $k$ -ramas, cada una correspondiente a un cluster. En nuestro ejemplo vemos la composición para  $k = 3$ .

```

import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt

# Calculate the linkage matrix
linkage_matrix = sch.linkage(df_std_distance, method='ward') #
    You can choose a different linkage method if needed

# Create the dendrogram
dendrogram = sch.dendrogram(linkage_matrix)

# Display the dendrogram
plt.show()

```

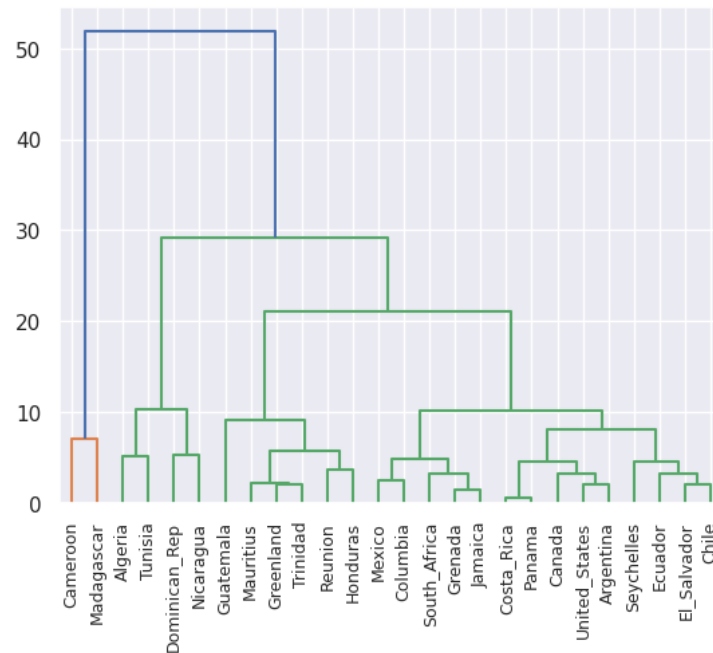


Figura 15: Dendrograma

Los diferentes colores en el dendrograma representan diferentes conglomerados o grupos de puntos según la salida del algoritmo de agrupamiento jerárquico (con el método de Ward). Los puntos que comparten un color común pertenecen al mismo grupo. La jerarquía se crea fusionando grupos más pequeños en otros más grandes.

El dendrograma comienza con cada observación como un grupo separado y los fusiona progresivamente hasta que todos los puntos están en un solo grupo en la parte superior. La altura a la que se fusionan dos grupos (o puntos) en el dendrograma representa la distancia entre ellos. Los grupos que se fusionan a menor altura son más similares entre sí.

El dendrograma permite entonces elegir un nivel apropiado de agrupamiento sin más que cortar el árbol a una altura específica, lo que equivale a elegir el número de grupos que se desea obtener.

Observando la estructura del dendrograma podemos hacernos una idea de cual podría ser el número más adecuado de cluster. Por ejemplo, en nuestro caso podría ser cuatro, ya que se puede apreciar como las uniones de esos cuatro cluster se realizan "pronto" (es decir están a poca distancia) y para que lleguen a unirse algunos de estos cluster hay que alejarse bastante. En ese caso podemos guardar los 4 cluster que hemos elegido para más tarde poder caracterizar cada uno.

```
# Assign data points to 4 clusters
from scipy.cluster.hierarchy import fcluster
num_clusters = 4
cluster_assignments = fcluster(linkage_matrix, num_clusters,
                              criterion='maxclust')

# Display the cluster assignments
print("Cluster Assignments:", cluster_assignments)

# Display the dendrogram
plt.show()

Cluster Assignments: [2 1 1 3 3 4 4 2 4 4 2 4 3 4 3 3 4 4 2 4 3
                     4 4 4 4 4]
```

La opción 'maxclust' tiene en cuenta la jerarquía y garantiza que no se formen más del número especificado de cluster. ( $num\_clusters$ ). Esto lo hace cortando el dendrograma a una altura determinada, lo que da como resultado el número deseado de grupos.

```
# Create a new column 'Cluster' and assign the '
    cluster_assignments' values to it
df['Cluster4'] = cluster_assignments
# Now 'df' contains a new column 'Cluster' with the cluster
    assignments
print(df["Cluster4"])
```



| PAIS          |   |
|---------------|---|
| Algeria       | 2 |
| Cameroon      | 1 |
| Madagascar    | 1 |
| Mauritius     | 3 |
| Reunion       | 3 |
| Seychelles    | 4 |
| South_Africa  | 4 |
| Tunisia       | 2 |
| Canada        | 4 |
| Costa_Rica    | 4 |
| Dominican_Rep | 2 |
| El_Salvador   | 4 |

Figura 16: Países y cluster (no se muestran todos)

Si además deseamos representar la situación relativa de los cluster calculados, podemos realizar un Análisis de Componentes Principales (ACP o PCA en inglés) sobre la matriz de correlaciones de los datos y utilizar las dos primeras componentes para representar los individuos en estos planos (Esto funcionará siempre y cuando el porcentaje de variabilidad recogida por estas dos componentes sea suficiente).

Se puede lograr lo anterior utilizando las bibliotecas de Python scikit-learn y matplotlib. Tendríamos que seguir estos 2 pasos:

1. Realiza un PCA para reducir los datos a 2 componentes principales.
2. Crear un diagrama de dispersión de los puntos de datos con colores según sus asignaciones de grupos.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Assuming 'df' is your original DataFrame with data
# 'cluster_assignments' contains cluster assignments

# Step 1: Perform PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(df)

# Create a new DataFrame for the 2D principal components
df_pca = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Step 2: Create a scatter plot with colors for clusters
plt.figure(figsize=(10, 6))

# Loop through unique cluster assignments and plot data points with
# the same color
for cluster in np.unique(cluster_assignments):
    plt.scatter(df_pca.loc[cluster_assignments == cluster, 'PC1'],
```

```

df_pca.loc[cluster_assignments == cluster, 'PC2'],
label=f'Cluster {cluster}'))

plt.title("2D PCA Plot with Cluster Assignments")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.grid()
plt.show()

```

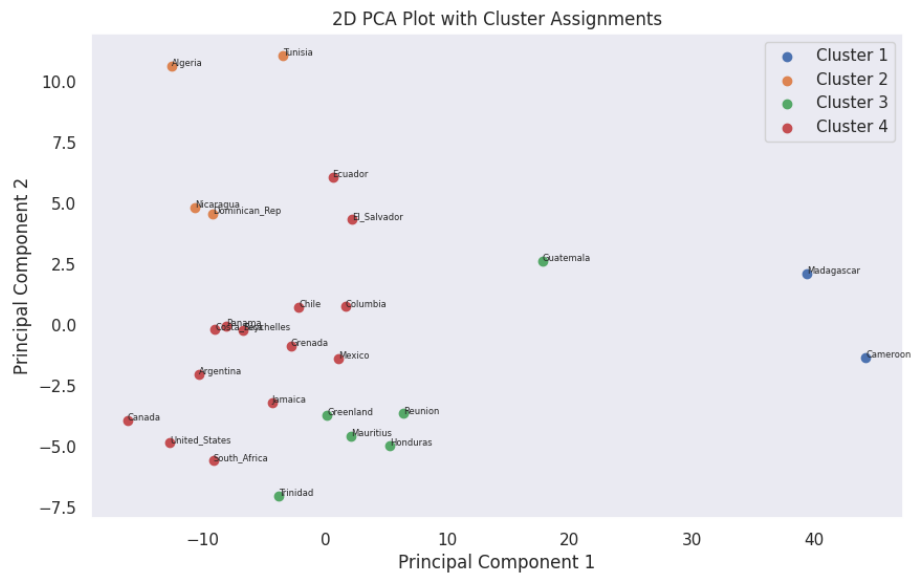


Figure 17: colored and labeled clustering



## 5. Algoritmos de clasificación no jerárquicos

Los métodos para realizar agrupaciones no jerárquicas se diferencian entre sí por el modo de escoger los centroides iniciales y por el criterio empleado para reasignar observaciones a los distintos grupos.

Para utilizar los algoritmos de agrupación no jerárquicos, es necesario fijar de antemano el número de grupos en que se pretende dividir las observaciones. Por esta razón, normalmente se utiliza un cluster jerárquico previamente para tener una idea aproximada del número de cluster adecuado. **Las técnicas de agrupación para realizar análisis cluster no jerárquicos siguen los siguientes pasos:**

1. Seleccionar  $K$  observaciones como centroides iniciales de los cluster a construir, siendo  $K$  el número deseado de cluster.
2. Asignar cada una de las observaciones restantes al cluster más próximo.
3. Reasignar cada observación a uno de los  $K$  cluster de acuerdo con una regla de parada determinada previamente.
4. Parar si no se reasignan observaciones a un grupo distinto del de partida, o si la reasignación satisface la regla de parada. En caso contrario, volver a 2.

Algunos de los métodos utilizados para obtener los centroides iniciales son:

- Seleccionar las  $K$  primeras observaciones con datos no faltantes.
- Seleccionar la primera observación con datos conocidos como el primer centroide. El segundo centroide es aquella observación cuya distancia al primer centroide sea tan grande como una distancia seleccionada previamente. El tercer centroide es la observación cuya distancia a los dos primeros centroides sea mayor que la distancia seleccionada. Y así sucesivamente.
- Seleccionar aleatoriamente  $K$  observaciones con datos conocidos.
- Elegir centroides que estén entre sí lo más lejanos posible.
- Utilizar centroides proporcionados por el investigador.

Una vez que se han identificado los centroides, se pueden formar los Clusters iniciales asignando cada una de las  $N - K$  observaciones restantes al Cluster correspondiente al centroide más próximo.

En cuanto a la forma de reasignar observaciones, algunas de las reglas más utilizadas son las siguientes:

1. **Criterio de convergencia:** Calcular el centroide de cada cluster y reasignar sujetos a los cluster cuyo centroide sea el más cercano. **Los centroides no varían mientras se reasignan observaciones, sino que se recalculan después de hacer la nueva asignación.** Si el cambio en el centroide es mayor que el valor determinado por un criterio de convergencia, se vuelve a hacer una reasignación de observaciones, y se vuelven a calcular los centroides. **El proceso de reasignación continúa hasta que el cambio en los centroides es menor que el valor dado por el criterio de convergencia.**
2. **Criterio estadístico:** Reasignar observaciones de acuerdo con algún criterio estadístico, como minimizar la traza de la matriz SSCP (dentro de los grupos) que es la suma de productos cruzados dentro de los grupos, etc.

Combinando los distintos métodos de selección de centroides iniciales y de asignación de observaciones se pueden desarrollar un gran número de algoritmos de cluster no jerárquicos.

Utilizamos la función *KMeans()* con una semilla concreta (1234 por ejemplo) indicando que el número de cluster es 4.

```
from sklearn.cluster import KMeans

# Set the number of clusters (k=4)
k = 4

# Initialize the KMeans model
kmeans = KMeans(n_clusters=k, random_state=0)

# Fit the KMeans model to your standardized data
kmeans.fit(df_std)

# Get the cluster labels for your data
kmeans_cluster_labels = kmeans.labels_
```

**ejercicio:** Puede apreciarse que la asignación de los cluster es casi idéntica que con el método jerárquico aplicado anteriormente. ¿Podrías encontrar las diferencias entre la agrupación final con ambos métodos?

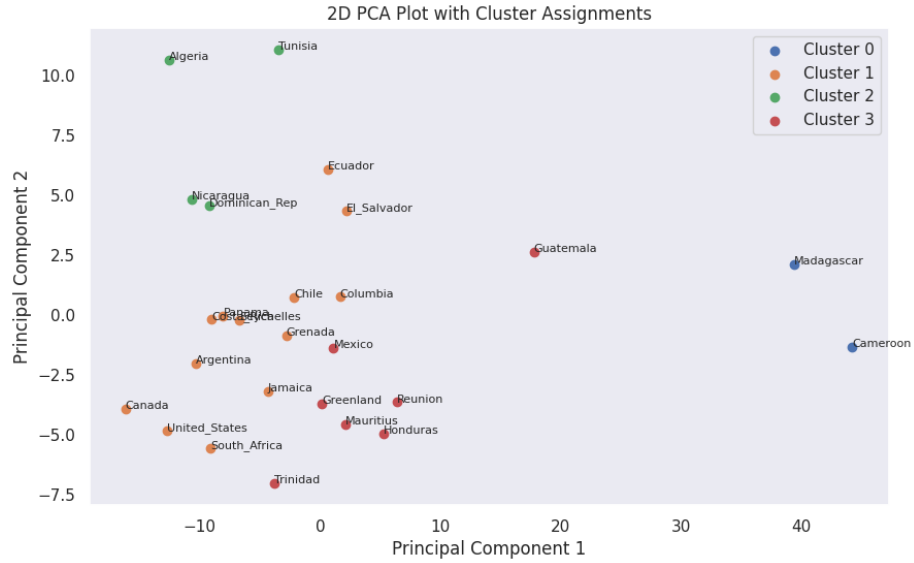


Figura 18: Caption

## 6. En busca del número adecuado de cluster

Después de obtener una solución en análisis Cluster, el paso siguiente es evaluar esta solución y determinar el número real de grupos existentes en nuestros datos. Nuestro objetivo es formar grupos lo más homogéneos “dentro de sí” y lo más diferentes “entre sí”. Para medir esto dividiremos en dos partes la variabilidad total de los datos, por un lado tendremos la que mide la variabilidad dentro de los grupos, y por el otro la que mide la variabilidad entre los grupos.

Dentro de las **estrategias para evaluar una solución de agrupación**, se pueden calcular varias medidas relacionadas con la variabilidad dentro de los grupos, entre grupos y la variabilidad total.

### ■ Variabilidad dentro de los grupo (suma de cuadrados dentro de los grupos o WCSS):

El WCSS (*Within-Cluster Sum of Squares*) mide la variabilidad dentro de cada grupo  $i$  ( $i$  de 1 a  $K$ ). Cuantifica qué tan lejos están los puntos de datos dentro de un grupo del centroide del grupo.

$$WCSS = \sum_{i=1}^K \sum_{j=1}^{n_i} d(x_{ij}, c_i)^2 \quad (7)$$

-  $K$  es el número de conglomerados. -  $n$  es el número de datos en el grupo

$i$ . -  $d(x_{ij}, c_i)$  es la distancia entre el dato  $x_{ij}$  y el centroide del grupo al que pertenece dicho dato  $i$ .

■ **Variabilidad entre conglomerados (suma de cuadrados entre conglomerados, BCSS):**

El BCSS (*Between-Cluster Sum of Squares*) mide la variabilidad entre los centroides de los conglomerados. Cuantifica qué tan lejos están los centroides del grupo entre sí.

$$\text{BCSS} = \sum_{i=1}^K n_i \cdot d(c_i, c)^2 \quad (8)$$

-  $n_i$  es el número de puntos de datos en el grupo  $i$ . -  $d(c_i, c)$  es la distancia entre el centroide del grupo  $i$  y el centroide general  $c$ , que es la media aritmética o centro de gravedad de toda la distribución.

■ **Variabilidad total (Suma total de cuadrados, TSS):**

El TSS mide la variabilidad total de los datos y es la suma de las distancias al cuadrado entre los puntos de datos y el centroide general.

$$\text{TSS} = \sum_{i=1}^K \sum_{j=1}^{n_i} d(x_{ij}, c)^2 \quad (9)$$

-  $n_i$  es el número total de datos en el grupo  $i$ . -  $d(x_{ij}, c)$  es la distancia entre el punto de datos  $x_{ij}$  y el centroide general  $c$ .

■ **Coefficiente de determinación  $R^2$ :**

$R^2$  es una medida de qué tan bien el agrupamiento explica la variabilidad de los datos. Se calcula como la relación entre BCSS y TSS.

$$R^2 = \frac{\text{BCSS}}{\text{TSS}} = 1 - \frac{\text{WCSS}}{\text{TSS}} \quad (10)$$

Estas medidas pueden ayudar a evaluar la calidad de la solución de clustering. Los valores de  $R^2$  más altos indican una mejor solución de agrupación, ya que explican una mayor parte de la variabilidad total de los datos. Los valores más bajos de WCSS sugieren que los datos están más cerca de sus respectivos centroides.

## 6.1. El método del codo

Tanto dentro del análisis jerárquico, como en el no jerárquico, existen procedimientos (además de las gráficas) que nos ayudan a decidir el número más adecuado de cluster. Una de esas técnicas es el conocido como método de *elbow* (método del codo):

El método del codo es una técnica popular para determinar el número óptimo  $K^*$  de conglomerados en un algoritmo de agrupamiento no jerárquico como por ejemplo el K-means. Este método ayuda a encontrar el punto (conocido como ‘punto de codo’ por que se produce un punto de inflexión equiparable al del brazo) en el que agregar más grupos no mejora significativamente la calidad de la agrupación. Así es como funciona:

1. Se calcula la WCSS (suma de cuadrados dentro de un grupo) para diferentes valores de K. La WCSS mide la variabilidad dentro de cada grupo y cuantifica qué tan lejos están entre sí los puntos dentro de un grupo respecto al centroide de ese mismo grupo.

2. Trazar los valores de WCSS frente al número de grupos (K). Normalmente verá una curva que comienza en alto y disminuye a medida que K aumenta. El punto donde la curva comienza a doblarse y formar una forma de ‘codo’ es el número óptimo de grupos.

El ‘punto de codo’ es donde el WCSS comienza a disminuir a un ritmo más lento, lo que indica que agregar más grupos no reduce significativamente el WCSS. Esta suele ser una buena opción para determinar la cantidad de cluster que se utilizarán en la solución (agrupación final).

El método del codo es un método heurístico útil para elegir el número de conglomerados, pero es importante recordar que no siempre es definitivo, y que también se deben considerar otros factores, como el conocimiento del dominio y los requisitos específicos del problema, al seleccionar la  $K$  óptima ( $K^*$ ).

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Create an array to store the WCSS values for different values
# of K:
wcss = []

for k in range(1, 11): # You can choose a different range of K
    values
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(df_std)
    wcss.append(kmeans.inertia_) # Inertia is the WCSS value#

Trace los valores de WCSS frente al n mero de grupos (K) y
busque el punto "codo":

plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='-', color='b')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters (K)')
```

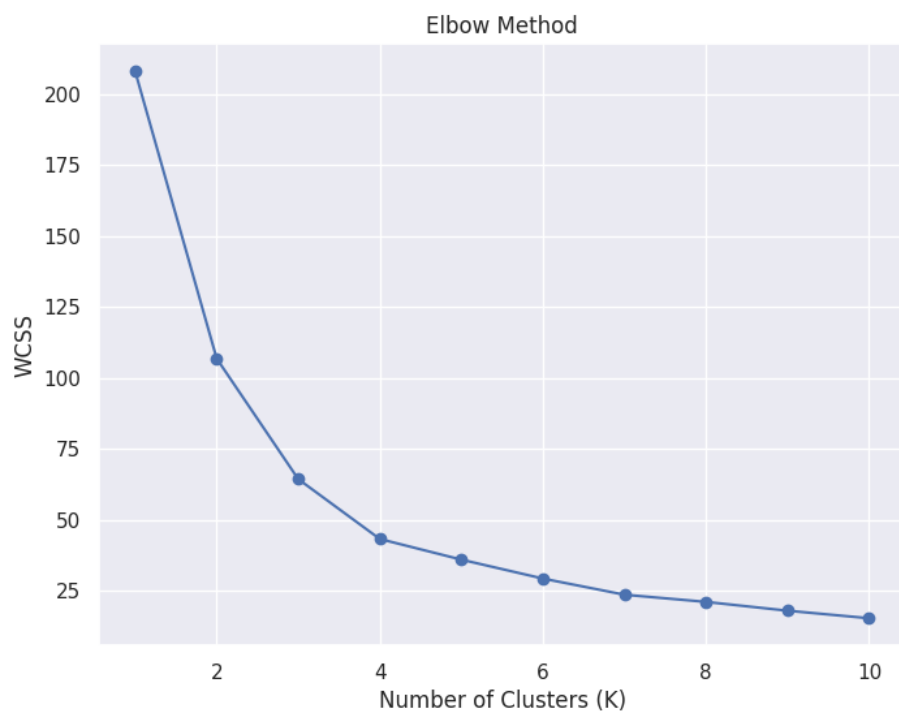


Figura 19: Elbow method

```
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

Se puede apreciar que el *WCSS* deja de reducirse drásticamente a partir de  $k = 4$ , y coincide con lo hallado por el clúster jerárquico. Con un quinto cluster la mejora sería muy poca.

## 6.2. El método de la silueta

Es otra técnica utilizada para determinar el número óptimo  $K^*$  de conglomerados. Se procede así:

1. Calcula la distancia promedio desde la observación a todos los demás puntos en el mismo grupo. Denota esto como  $a_i$ , donde  $a_i$  es la distancia promedio dentro del grupo para el punto de datos  $i$ .
2. Calcula la distancia promedio desde la observación hasta todos los puntos del cluster más cercano (un grupo al que no pertenece el punto de datos). Denota esto como  $b_i$ , donde  $b_i$  es la distancia promedio entre grupos para la observación  $i$ .

3. Calcula la puntuación de silueta para cada observación:

$$\text{Puntuación de silueta}_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

4. Se calcula la puntuación general de silueta para la solución de agrupación, que es el promedio de las puntuaciones de silueta de todas las observaciones.

$$\text{Puntuación de silueta} = \frac{1}{N} \sum_{i=1}^N \text{Puntuación de silueta}_i$$

**La puntuación de silueta varía entre -1 y 1.** Una puntuación cercana a 1 indica que la observación coincide bien con su propio grupo y no coincide con los grupos vecinos. Una puntuación cercana a 0 indica grupos superpuestos, donde la observación podría estar en cualquiera de los grupos adyacentes. Una puntuación cercana a -1 indica que la observación probablemente esté asignado al grupo incorrecto.

En la práctica, se puede utilizar la puntuación de silueta para comparar diferentes soluciones de agrupación y elegir la que tenga la puntuación de silueta más alta, ya que representa un buen equilibrio entre cohesión y separación en las agrupaciones.

Para implementar el método de silueta en Python procedemos así:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Create an array to store silhouette scores for different
# values of K

silhouette_scores = []

# Run K-means clustering for a range of K values and calculate
# the silhouette score for each K:

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(df_std)
    labels = kmeans.labels_
    silhouette_avg = silhouette_score(df_std, labels)
    silhouette_scores.append(silhouette_avg)

plt.figure(figsize=(8, 6))
plt.plot(range(2, 11), silhouette_scores, marker='o', linestyle='-', color='b')
plt.title('Silhouette Method')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.grid(True)
plt.show()
```



Figura 20: Silouhette method

En el gráfico, debes buscar el valor de K que corresponda a la puntuación de silueta más alta, ya que suele ser una buena opción para la cantidad de clústeres que deberías utilizar en tu solución (agrupamiento final). En este caso da a elegir 2, pero si nos parece una solución demasiado simplista el siguiente máximo local con un K más razonable es 4, que es precisamente el número que venimos manejando.

Otra variante del gráfico anterior nos genera valores de silueta para cada cluster una vez elegido un número de cluster. Se ejecuta entonces la agrupación de K-medias con el número óptimo de grupos (determinado mediante el método de silueta).

```
from sklearn.metrics import silhouette_samples

# Calculates silhouette scores for each cluster
silhouette_values = silhouette_samples(df_std, labels)

plt.figure(figsize=(8, 6))
y_lower = 10
for i in range(4):
    ith_cluster_silhouette_values = silhouette_values[labels == i]
    ith_cluster_silhouette_values.sort()
```



```

size_cluster_i = ith_cluster_silhouette_values.shape[0]
y_upper = y_lower + size_cluster_i

color = plt.cm.get_cmap("Spectral")(float(i) / 4)
plt.fill_betweenx(np.arange(y_lower, y_upper),
                  0, ith_cluster_silhouette_values,
                  facecolor=color, edgecolor=color, alpha
                    =0.7)
plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
y_lower = y_upper + 10
plt.title("Silhouette Plot for Clusters")
plt.xlabel("Silhouette Coefficient Values")
plt.ylabel("Cluster Label")
plt.grid(True)
plt.show()

```

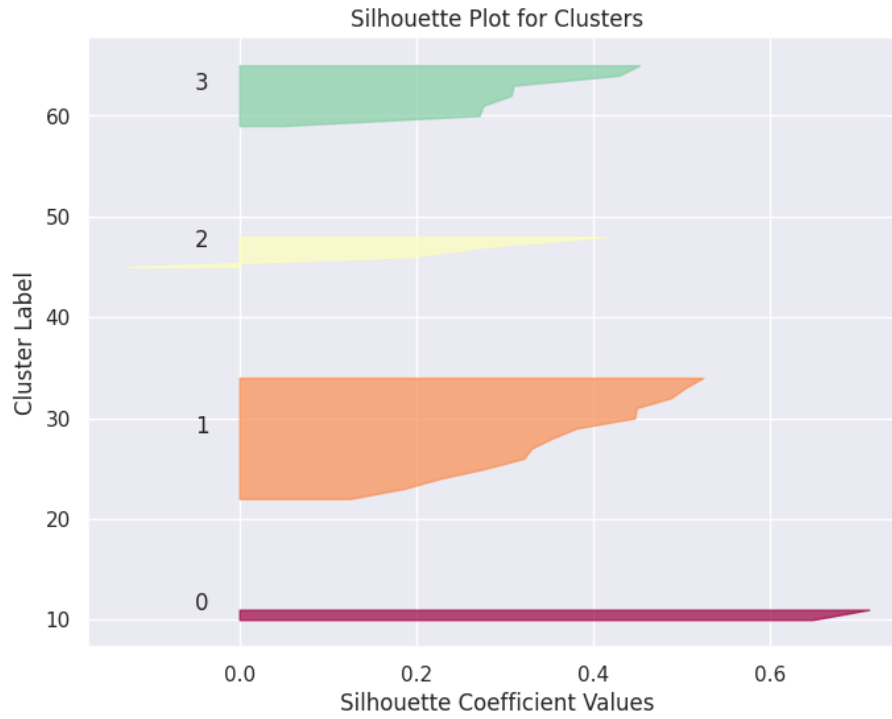


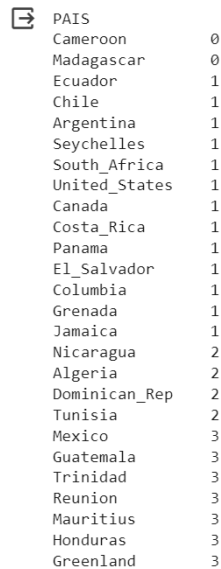
Figura 21: Puntuaciones de las siluetas siluetas de los cluster

En este código  $optima_k$  es el número de grupos determinados mediante el método de silueta.  $silueta\_samples$  calcula las puntuaciones de silueta para cada observación. Cada grupo es representado por una región coloreada. El ancho de cada región representa la puntuación de silueta para cada observación dentro del grupo. Este gráfico proporciona información sobre la calidad del agrupamiento al mostrar qué tan bien separados están las observaciones dentro de sus respectivos grupos. Los coeficientes de silueta más altos indican grupos mejor definidos. De este modo se ve claramente que el grupo 0 es el mejor definido ya que sus observaciones tienen todas ellas una puntuación superior a 0.6, seguido del 1, aunque este grupo tiene cierto porcentaje de observaciones con puntuaciones más bajas (lo que podría ser indicativo de que se podría plantear la inclusión de otro grupo en la solución). El grupo 3 es muy homogéneo (las puntuaciones tienen menor oscilación o rango). El grupo más problemático sería el 2 al ser un grupo con puntuaciones muy dispares que hasta llegan a ser negativas sugiriendo que podrían haber incluso pertenecido a otro grupo e indican también la existencia de solapamiento con otro de los grupos.

## 7. Caracterización de los cluster

Una vez que se ha decidido la partición de los cluster, el siguiente paso consistirá en caracterizarlos. Se debe realizar análisis descriptivo sobre las variables activas utilizadas en el análisis, con ello se determinará las medias y varianzas todas las observaciones. Ello nos permitirá una primera caracterización permitiendo interpretar las características de cada cluster con respecto a las variables originales. Lo primero que debemos tener claro son los individuos que pertenecen a cada cluster

```
# Add the labels as a new column to the DataFrame
df_std['label'] = labels
# Sort the DataFrame by the "label" column
df_std_sort = df_std.sort_values(by="label")
df_std_sort
```



The screenshot shows a Jupyter Notebook interface with a DataFrame named 'df\_std\_sort'. The DataFrame has two columns: 'PAIS' (Country) and a numerical 'label' column. The countries are grouped by their label, with labels 0, 1, 2, 3, and 4. The countries are listed in the following order: Cameroon, Madagascar, Ecuador, Chile, Argentina, Seychelles, South\_Africa, United\_States, Canada, Costa\_Rica, Panama, El\_Salvador, Columbia, Grenada, Jamaica, Nicaragua, Algeria, Dominican\_Rep, Tunisia, Mexico, Guatemala, Trinidad, Reunion, Mauritius, Honduras, and Greenland.

| PAIS          | label |
|---------------|-------|
| Cameroon      | 0     |
| Madagascar    | 0     |
| Ecuador       | 1     |
| Chile         | 1     |
| Argentina     | 1     |
| Seychelles    | 1     |
| South_Africa  | 1     |
| United_States | 1     |
| Canada        | 1     |
| Costa_Rica    | 1     |
| Panama        | 1     |
| El_Salvador   | 1     |
| Columbia      | 1     |
| Grenada       | 1     |
| Jamaica       | 1     |
| Nicaragua     | 2     |
| Algeria       | 2     |
| Dominican_Rep | 2     |
| Tunisia       | 2     |
| Mexico        | 3     |
| Guatemala     | 3     |
| Trinidad      | 3     |
| Reunion       | 3     |
| Mauritius     | 3     |
| Honduras      | 3     |
| Greenland     | 3     |

Figura 22: Cluster ordenados

Si analizamos las medias de cada uno de los cluster de que disponemos en la salida del cluster jerárquico, no tenemos una información para encontrar las diferencias de cada cluster porque

```
# Group the data by the 'label' column and calculate the mean
of each group
cluster_centroids = df_std_sort.groupby('label').mean()
cluster_centroids.round(2)
# 'cluster_centroids' now contains the centroids of each
cluster
```

Cuadro 2: Las medias (centroides) de los cluster formados

| Label | <i>m0</i> | <i>m25</i> | <i>m50</i> | <i>m75</i> | <i>w0</i> | <i>w25</i> | <i>w50</i> | <i>w75</i> |
|-------|-----------|------------|------------|------------|-----------|------------|------------|------------|
| 0     | -2.92     | -2.86      | -2.3       | -1.16      | -2.91     | -2.81      | -2.27      | -1.37      |
| 1     | 0.42      | 0.32       | 0.27       | 0.04       | 0.45      | 0.4        | 0.26       | 0.11       |
| 2     | 0.36      | 1.03       | 1.21       | 1.79       | 0.28      | 0.98       | 1.42       | 1.57       |
| 3     | -0.15     | -0.36      | -0.54      | -0.76      | -0.17     | -0.51      | -0.64      | -0.71      |

Para caracterizar mejor las diferencias entre clusters es mucho más útil obtener los estadísticos de resumen de las variables originales ya que se puede interpretar según la magnitud de las variables.

```
# Add the labels as a new column to the DataFrame
df['label'] = labels
# Sort the DataFrame by the "label" column
df_sort = df.sort_values(by="label")

# Group the data by the 'label' column and calculate the mean of
# each group
cluster_centroids_orig = df_sort.groupby('label').mean()
cluster_centroids_orig.round(2)
# 'cluster_centroids' now contains the centroids of each cluster
```

Cuadro 3: Las medias (centroides) de los cluster formados

| Label | <i>m0</i> | <i>m25</i> | <i>m50</i> | <i>m75</i> | <i>w0</i> | <i>w25</i> | <i>w50</i> | <i>w75</i> | Cluster4 |
|-------|-----------|------------|------------|------------|-----------|------------|------------|------------|----------|
| 0     | 36.0      | 29.5       | 15.0       | 6.0        | 38.0      | 33.0       | 18.5       | 6.5        | 1.0      |
| 1     | 62.46     | 45.23      | 24.15      | 8.54       | 67.46     | 49.54      | 27.23      | 10.54      | 4.0      |
| 2     | 62.0      | 48.75      | 27.5       | 12.25      | 66.0      | 52.5       | 31.25      | 14.5       | 2.0      |
| 3     | 58.0      | 41.86      | 21.29      | 6.86       | 62.0      | 44.86      | 24.14      | 8.29       | 3.14     |

Por ejemplo, los cluster 1 a 3 son los que tiene mayor esperanza de vida tanto en hombres como en mujeres. Al nacer son el 1 y el 2 los que tienen mayor esperanza de vida, mientras que una vez llegado a la tercera edad es el clúster 2 el que presenta superioridad (12 años más de vida frente a 8 y 6 respectivamente de los clúster 1 y 3). Podríamos comentar a continuación los países que los integran o algunos de estos países a modo de ejemplo para completar la caracterización. Muy diferente es el cluster 0 es el que tiene menor. Esta interpretación nos da una explicación verosímil del trabajo de clasificación realizado. Es importante recordar que sin esta última parte el análisis quedaría incompleto.