Clustering Analysis - Análisis Cluster (con Python)

Pablo Flores-Vidal

21 de noviembre de 2023

Introducción

- Tendencias del aprendizaje automático y el énfasis en algoritmos supervisados.
- Métodos no supervisados y la importancia del *Clustering*
- La relevancia de crear grupos (cluster) basados en las características de los datos.

Ejemplo 'Pingüinos'

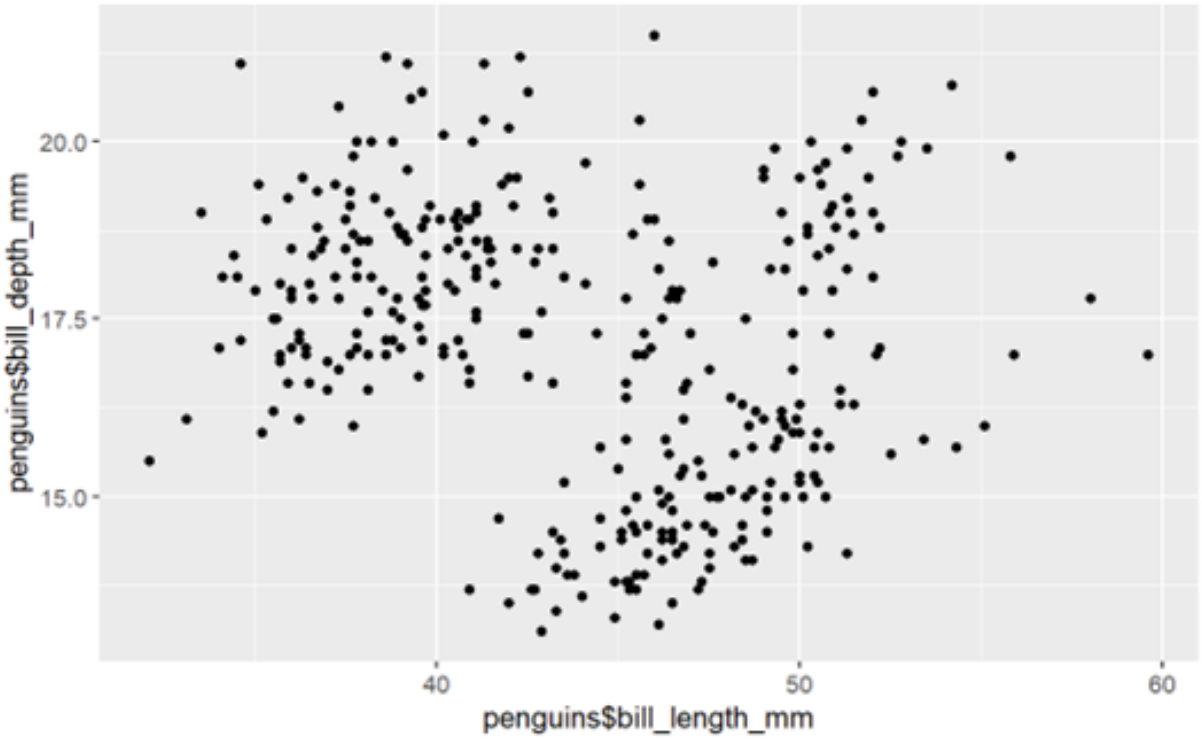
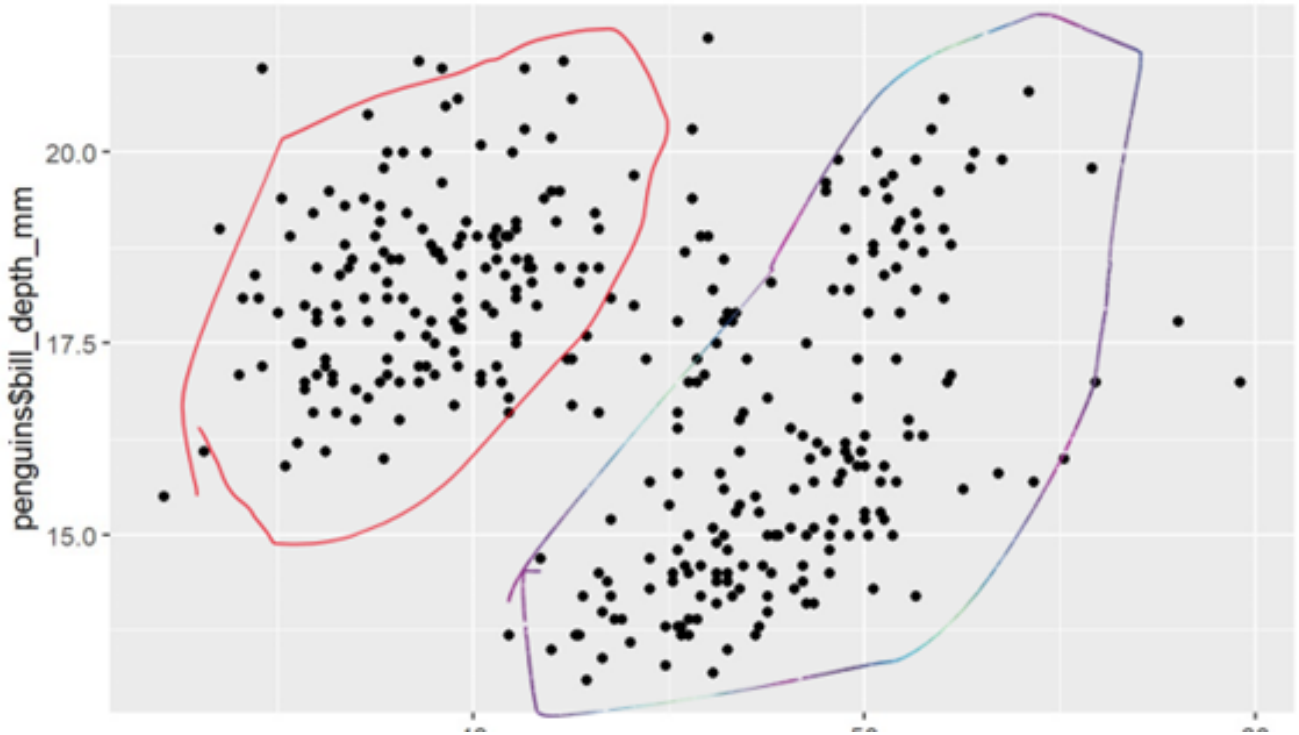


Figura: Distribución de las medidas de pingüinos.

Identificación de grupos potenciales



- Identificados 2 potenciales cluster en los datos.
- Los grupos podrían representar diferentes. subespecies.

Aplicaciones del *Clustering*

- Segmentación aplicada al Marketing (*aka* segmentación de mercados) para campañas selectivas.
- Perfilado de clientes para aprobación de préstamos bancarios.
- Estudios genéticos para identificar individuos con rasgos genéticos similares.
- etc.

¿En qué consiste el *Clustering*?

Clustering: Técnica de análisis multivariante (basada en clasificación no supervisada) que permite la creación de grupos (*cluster*) significativos a partir de un conjunto de individuos.

- Objetivo: Homogeneidad interna, heterogeneidad externa.
- Los grupos son mutuamente excluyentes; cada observación pertenece a un grupo.

Tipologías

- Diferentes medidas de distancia para evaluar la similitud entre observaciones.
- Dos procedimientos de agrupamiento: jerárquico y no jerárquico.
- Recomendación práctica: comenzar con jerárquico y luego aplicar no jerárquico.

Clustering jerárquico

- Primero, elija un método de vinculación para la agrupación jerárquica.
- Decidir sobre el número óptimo de clusters.
- Interpretar la solución elegida.

Ejemplo: agrupar países según la esperanza de vida para diferentes edades y género.

```
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()
```

Inspeccionando los datos de Esperanza de vida

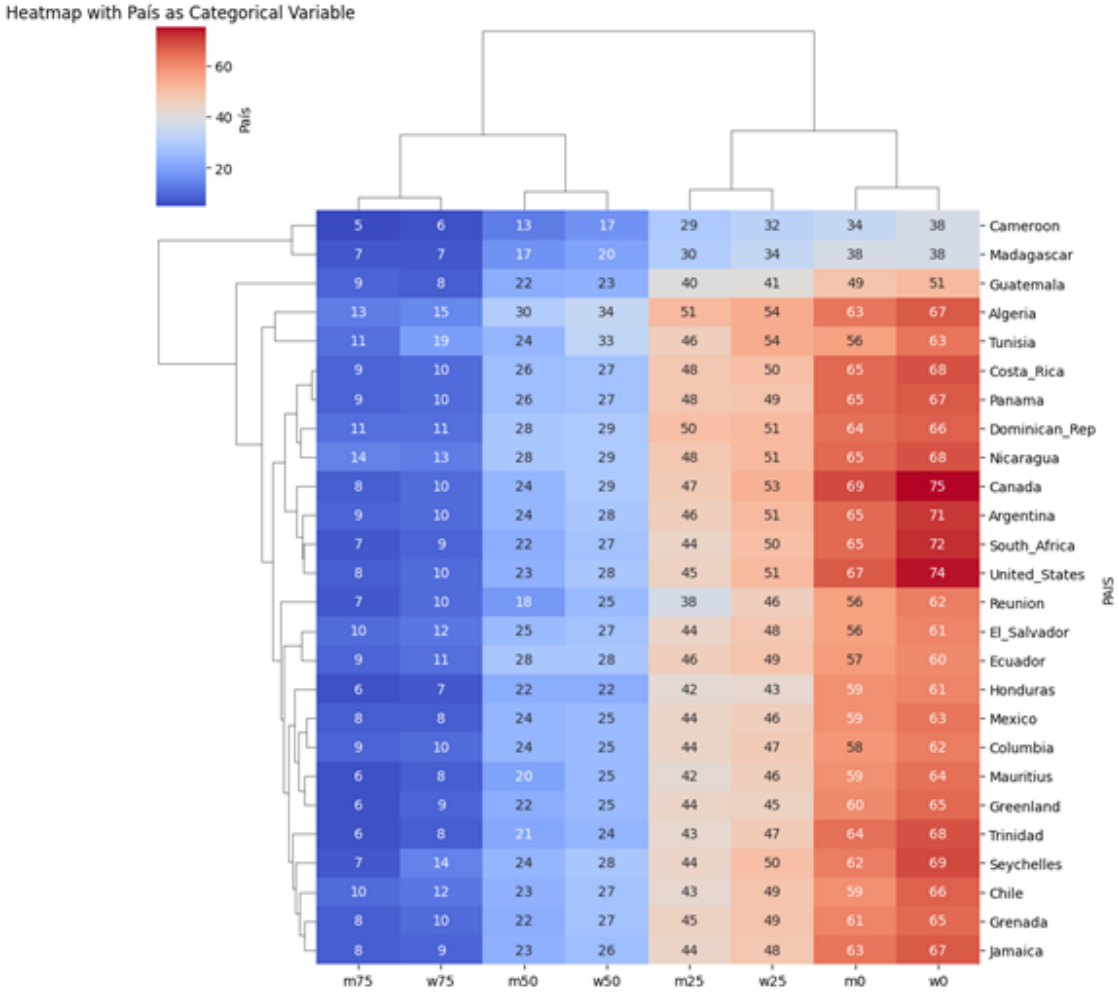
ndex	PAIS	m0	m25	m50	m75	w0	w25	w50	w75
0	Algeria	63	51	30	13	67	54	34	15
1	Cameroon	34	29	13	5	38	32	17	6
2	Madagascar	38	30	17	7	38	34	20	7
3	Mauritius	59	42	20	6	64	46	25	8
4	Reunion	56	38	18	7	62	46	25	10

Figura: Análisis exploratorio de datos: variables de esperanza de vida.

Clustering jerárquico en Python

```
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()
```

Clustering jerárquico (mapa de calor) *heatmap*.



Medidas de distancia entre observaciones

- Utilizar medidas de distancia para evaluar la similitud entre observaciones.
- Medidas comunes: Euclidiana, Manhattan, Minkowski, Chebyshev, etc.

```
from scipy.spatial import distance

distancematrix = distance.cdist(df,df,'euclidean')
distancesmall = distancematrix[:5, :5]
distancesmall = pd.DataFrame(distancesmall, index = df.
    index[:5], columns=df.index[:5] )
distancesmallrounded = distancesmall.round(2)
print('Distance Matrix:', distancesmallrounded)
```

Matriz de distancias

Cuadro: Matriz de distancias (fragmento).

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

Mapa de calor

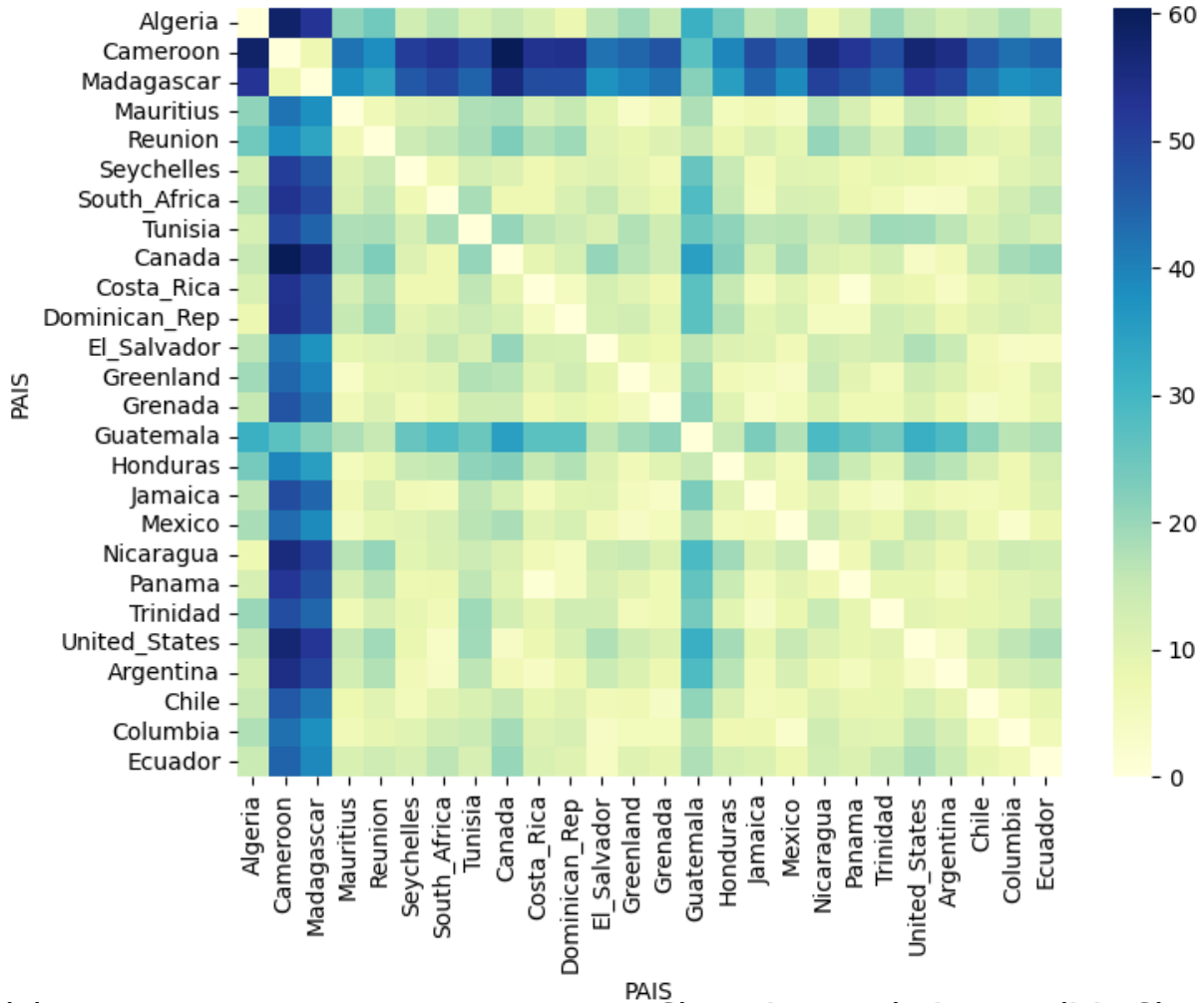


Figura: Mapa de calor *heatmap*.

Estandarización de variables

- Al estandarizar las variables las medidas se hacen independientes de las unidades.
- Fórmula para estandarizar: $z_{ij} = \frac{x_{ij} - \overline{x}_j}{s_j}$

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

\# Fit and transform the DataFrame to standardize the
    columns

dfstd = pd.DataFrame(scaler.fit\_transform(df),columns=
    df.columns)
```

Mapa de calor ‘estandarizado’

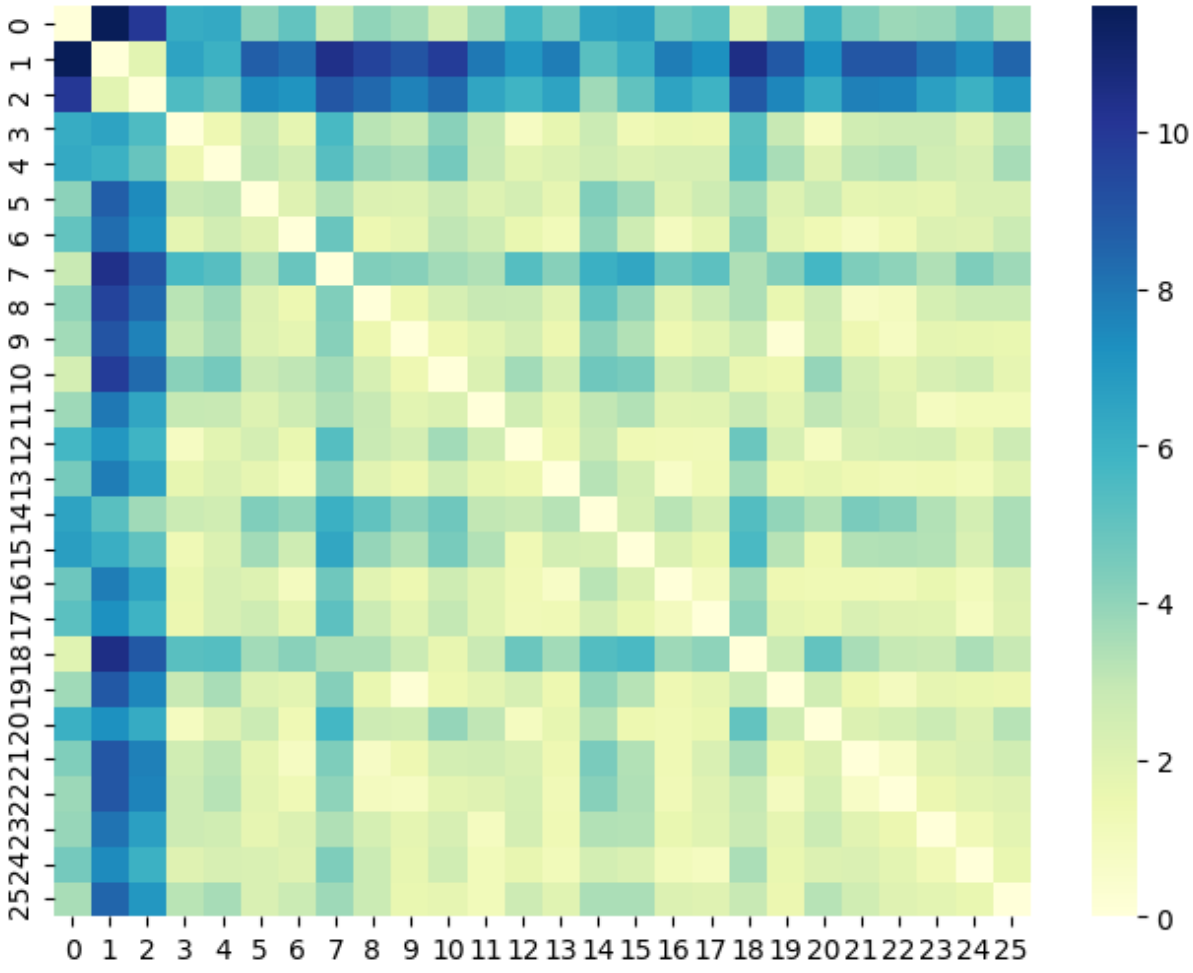


Figura: Mapa de calor ‘estandarizado’.

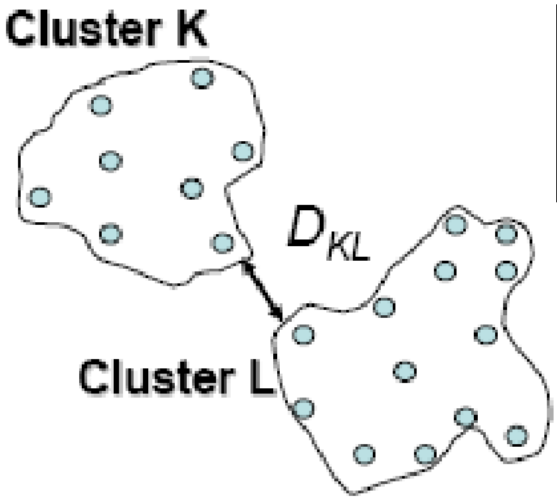
Clasificación Jerárquica

- Configura grupos con estructura arborescente.
- No produce un número fijo de cluster.
- Empiezan siendo observaciones individuales, y a cada paso van siendo englobados en otros cluster con más observaciones.

Pasos para el Clustering Jerárquico

1. Se inicia con tantos grupos como observaciones.
2. Se genera una matriz de distancias de dimensión *nxn*.
3. Se agrupan las dos observaciones más cercanas.
4. Se obtiene una nueva matriz de distancias.
5. Se repite el proceso hasta una única agrupación que contiene todas las observaciones.

Enlace Simple (vecino más cercano)

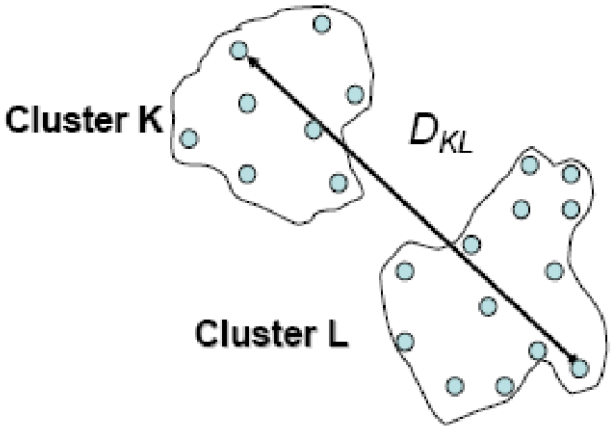


$$D_{KL} = \min_{\substack{i \in C_K \\ j \in C_L}} d(x_i, x_j)$$

Figura: Distancia entre cluster: Vecino más cercano.

- Calcula la distancia entre dos grupos como la distancia entre sus observaciones más cercanas.
- Tiende a crear grupos con muchas observaciones.

Enlace completo (o método del vecino más alejado)

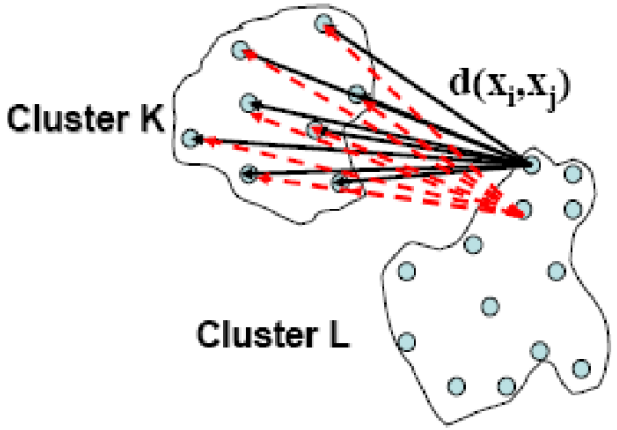


$$D_{KL} = \max_{\substack{i \in C_K \\ j \in C_L}} d(x_i, x_j)$$

Figura: Distancia entre cluster: Vecino más lejano.

- Distancia entre dos cluster como la distancia entre observaciones más alejadas.
- Tiende a crear grupos más compactos que el vecino más cercano.

Enlace Medio



$$D_{KL} = \frac{1}{n_K n_L} \sum_{i \in C_K} \sum_{j \in C_L} d(x_i, x_j)$$

Figura: Distancia entre cluster según el método del enlace medio.

- Distancia entre dos cluster como la distancia media entre observaciones.
- Grupos con varianza similar y pequeña.

Distancia entre centroides

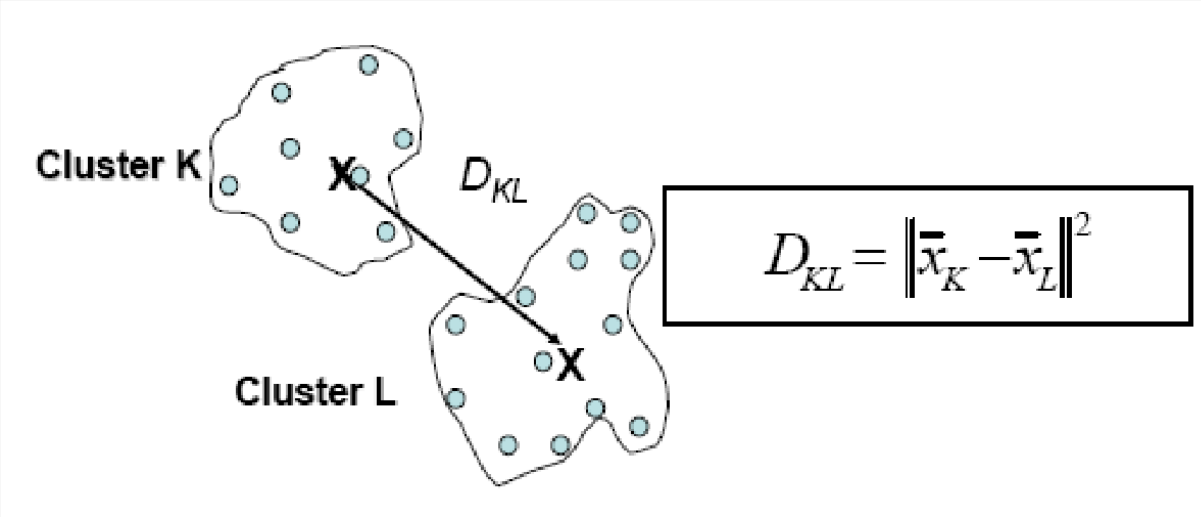


Figura: Distancia entre centroides.

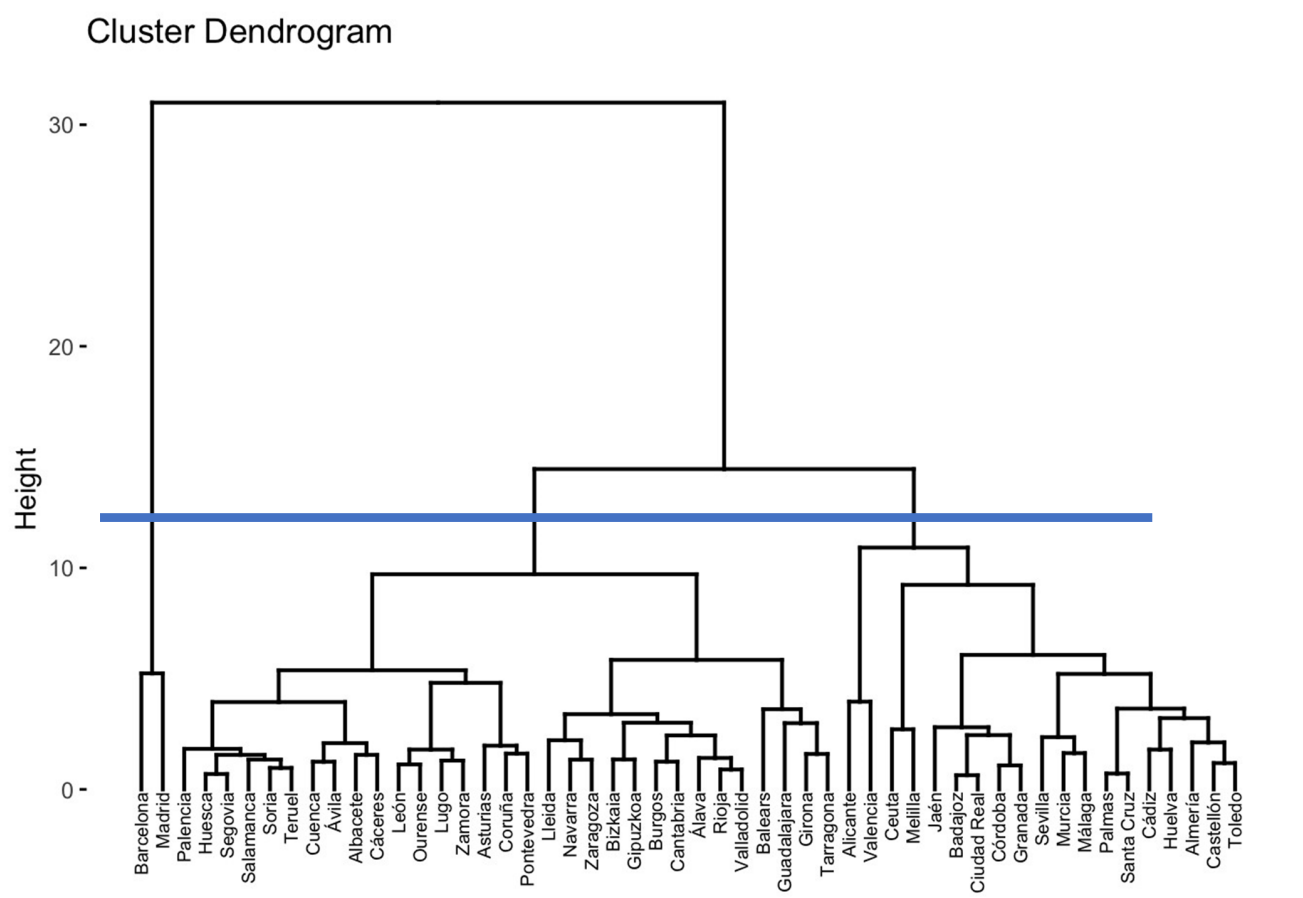
- Distancia entre dos grupos como la distancia entre sus centroides.
- Los centroides representan el vector medio de las p variables. Sería el centro de gravedad de cada cluster.

Distancia de Ward

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

- Mínima varianza interna de los grupos resultantes.
- Selecciona la unión que minimiza la variabilidad.

El dendrograma



- Ayuda a elegir un nivel apropiado de agrupamiento.

El dendograma en Python

```
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
\# Se calcula la matriz de enlace
linkage\_matrix =
sch.linkage(df\_std\_distance, method='ward')
\# Crea el dendrograma
dendrogram = sch.dendrogram(linkage\_matrix)
\# Muestra el dendrograma
plt.show\(\)
```

- Asignación de puntos a cluster utilizando el dendrograma.
- Visualización de cluster con diferentes colores.

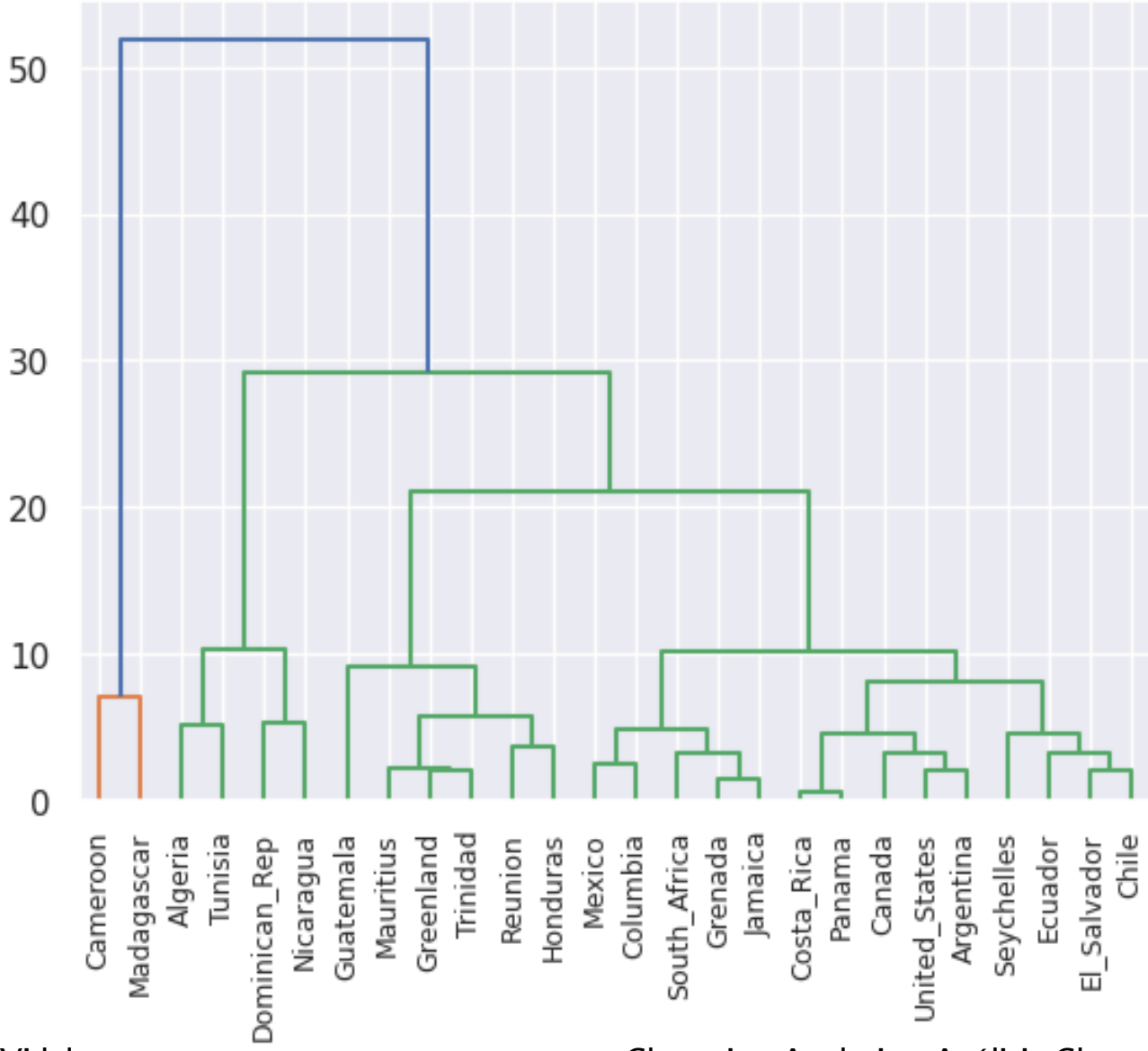
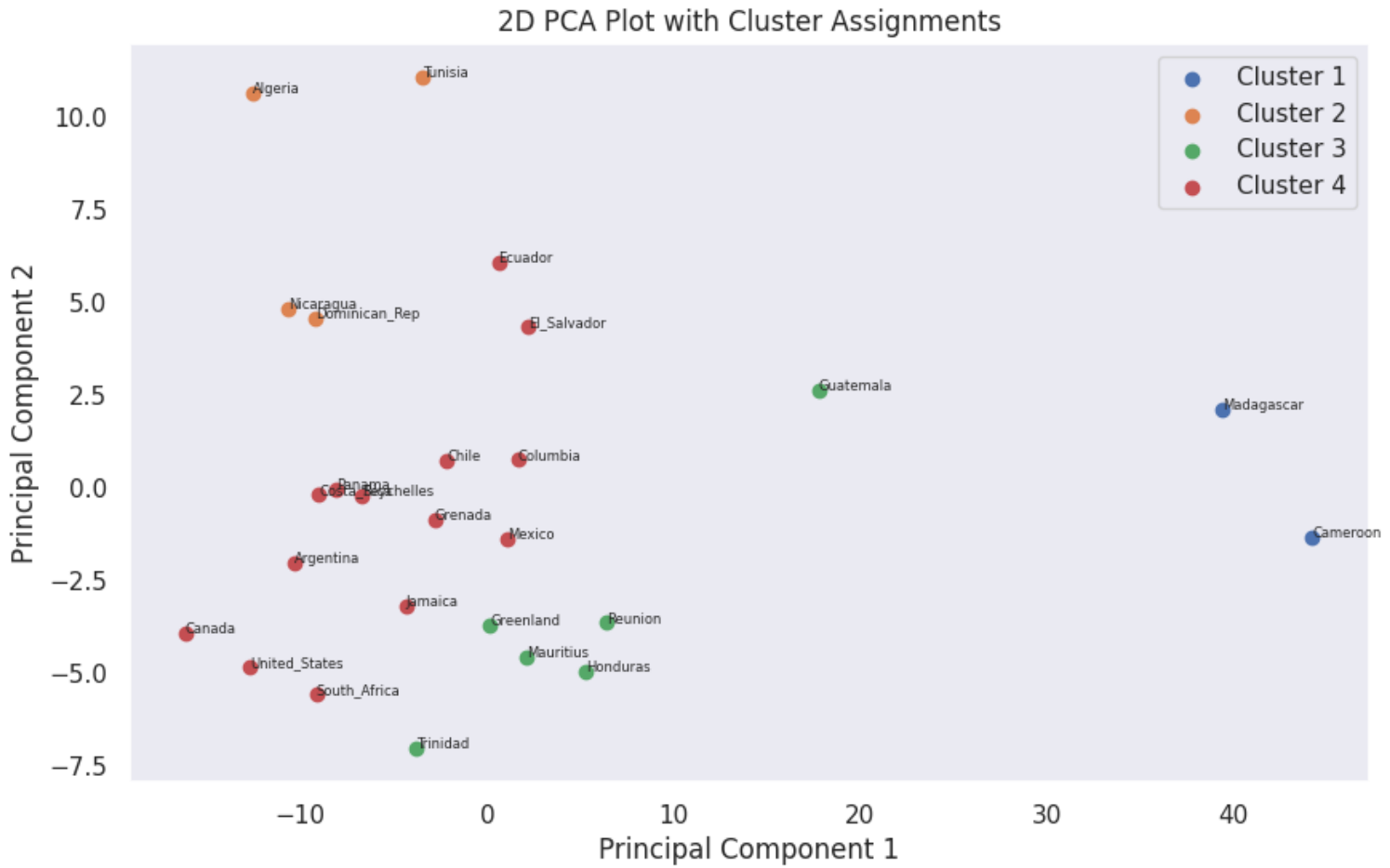


Figura: Dendograma

El número de cluster según el dendrograma



Conclusión

- El método de Ward es comúnmente utilizado por su homogeneidad.
- Las longitudes de las ramas son proporcionales a las distancias entre los puntos/agrupamientos.
- Depende de la distancia entre elementos y entre cluster utilizada.
- Nos ayuda a determinar el número adecuado de cluster.

Algoritmos de clustering no jerárquicos

- Es necesario fijar de antemano el número de grupos (limitación principal).
- Se puede utilizar un método jerárquico previamente para tener una idea aproximada del número.

Pasos en el clustering no jerárquico

1. Selecciona k observaciones como centroides iniciales.
2. Asigna cada observación (de las restantes $N - k$) al grupo más cercano.
3. Reasigna las observaciones según una regla de detención predefinida.
4. Se detiene si no se reasignan observaciones o se cumple la regla; de lo contrario, regrese al paso 2.

Métodos para inicializar los centroides

- 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.
- Elegir centroides que estén entre sí lo más lejanos posible.
- Utilizar centroides proporcionados por el investigador.

Reglas para reasignar observaciones

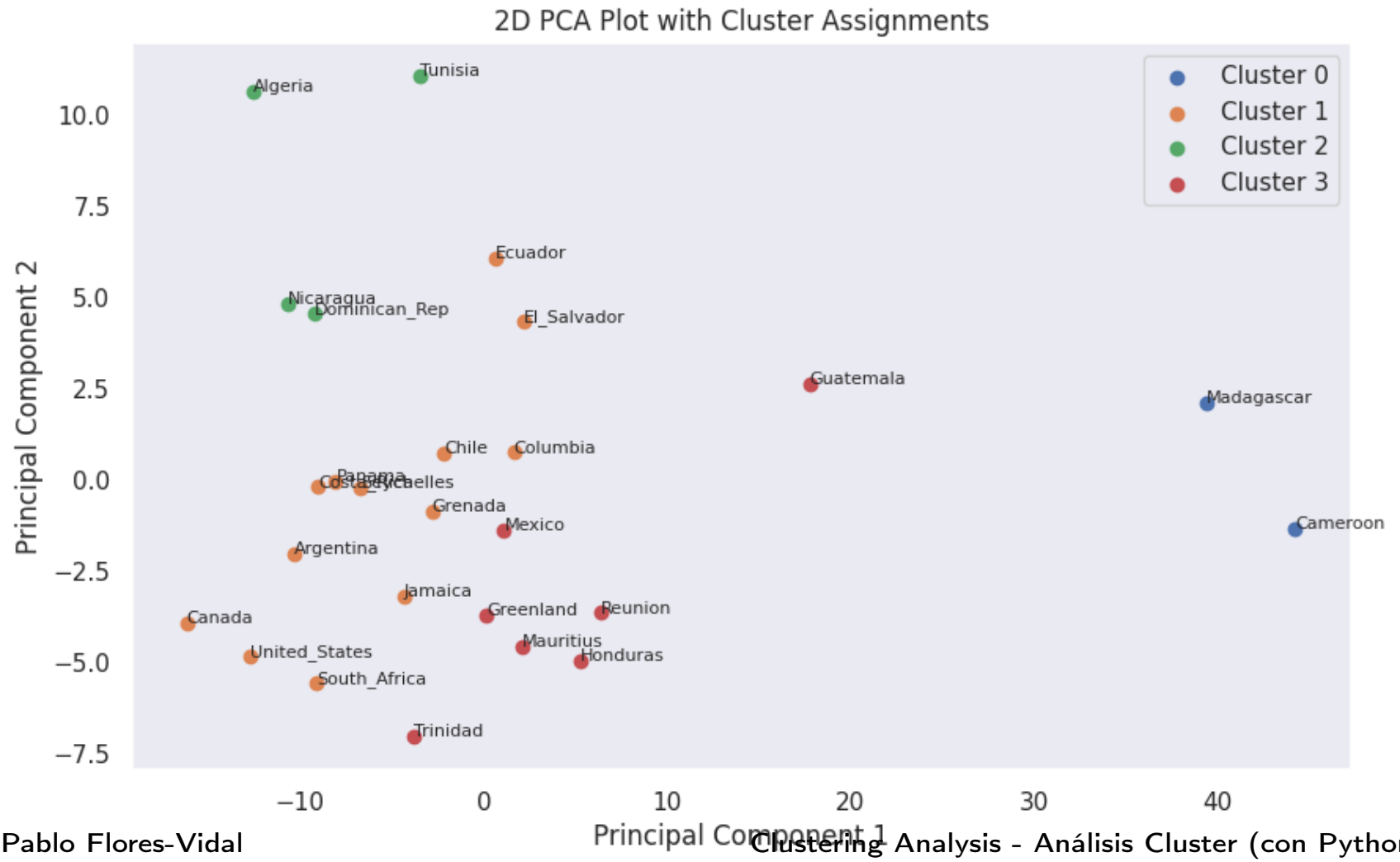
1. Calcular los centroides de los conglomerados y asignar sujetos a los conglomerados más cercanos.
2. Asignación de sujetos según criterios estadísticos, por ejemplo, minimizando el rastro de la matriz SSCP.

Implementación del método de K-Means

- Utilizando la función *KMeans*() con una semilla fija y 4 cluster.
- Garantizando valores aleatorios consistentes en todas las computadoras.

```
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_
```

Resultado del K-Means



En busca del número apropiado de cluster: medidas de variabilidad I

1. WCSS: Variabilidad dentro del grupo.

$$\text{WCSS} = \sum_{i=1}^{K} \sum_{j=1}^{n} d(x_{ij}, c_i)^2 \tag{1}$$

2. TWSS: Variabilidad total dentro de todos los cluster.

$$\text{TWSS} = \text{WCSS} \tag{2}$$

3. BCSS: Variabilidad dentro de los cluster.

$$\text{BCSS} = \sum_{i=1}^{K} n_i \cdot d(c_i, c)^2 \tag{3}$$

En busca del número apropiado de cluster: medidas de variabilidad II

1. TSS: Variabilidad total en los datos.

$$\text{TSS} = \sum_{i=1}^{K} \sum_{j=1}^{n} d(x_{ij}, c)^2 \tag{4}$$

2. R^2: Coeficiente de determinación.

$$R^2 = 1 - \frac{\text{WCSS}}{\text{TSS}} \tag{5}$$

Medidas de evaluación: método de *Elbow*

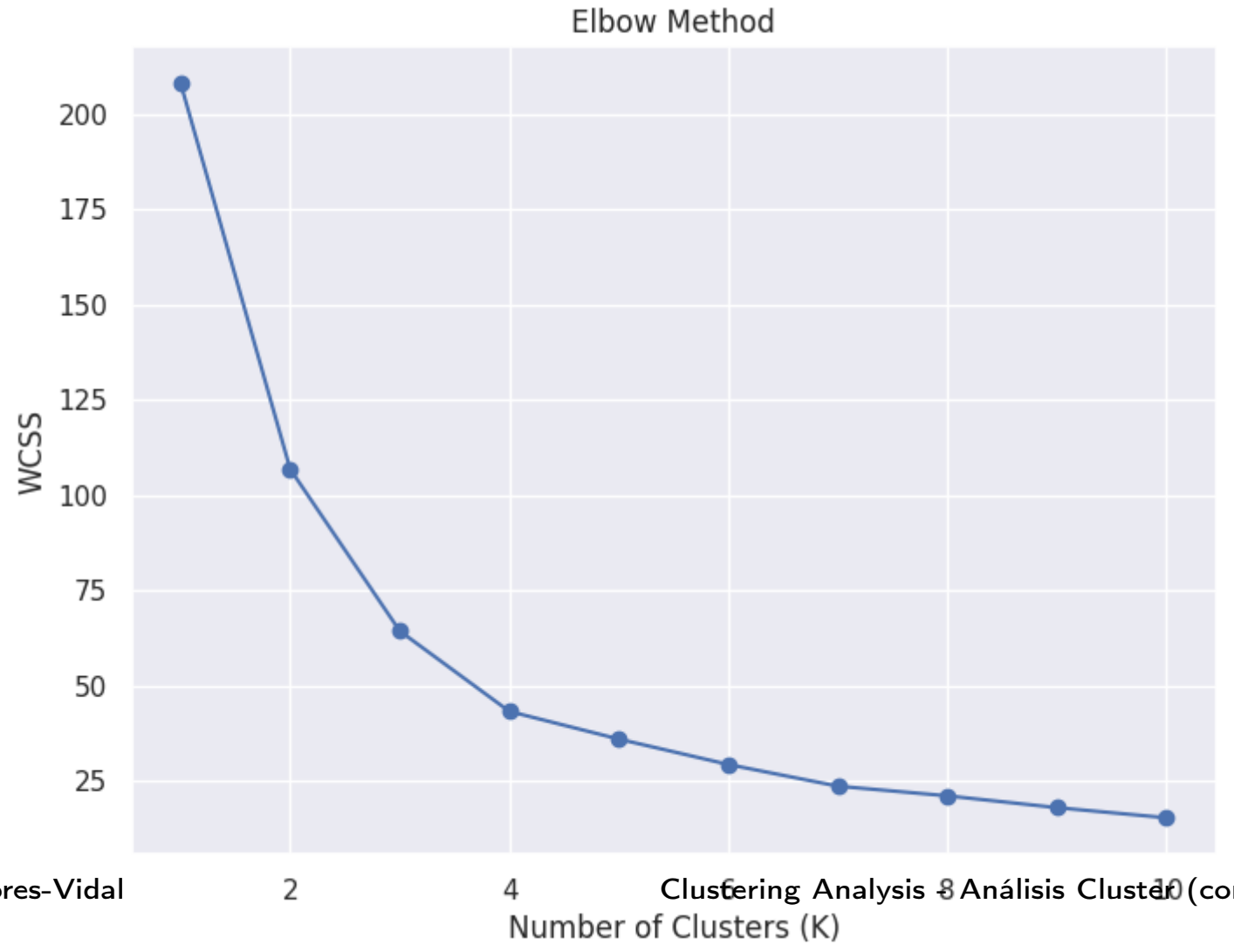
- Técnica para determinar el número óptimo de clusters k en el método de K-medias.

$$\text{WCSS}(K) = \sum_{i=1}^{K} \sum_{j=1}^{n} d(x_{ij}, c_i)^2 \tag{6}$$

Gráfico del método de *Elbow*

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
# Store the WCSS values for different values of K:
wcss = []
for k in range(1, 11):
    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)')
    label('WCSS')
    rid(True)
    how()
```

Gráfico del método de *Elbow*



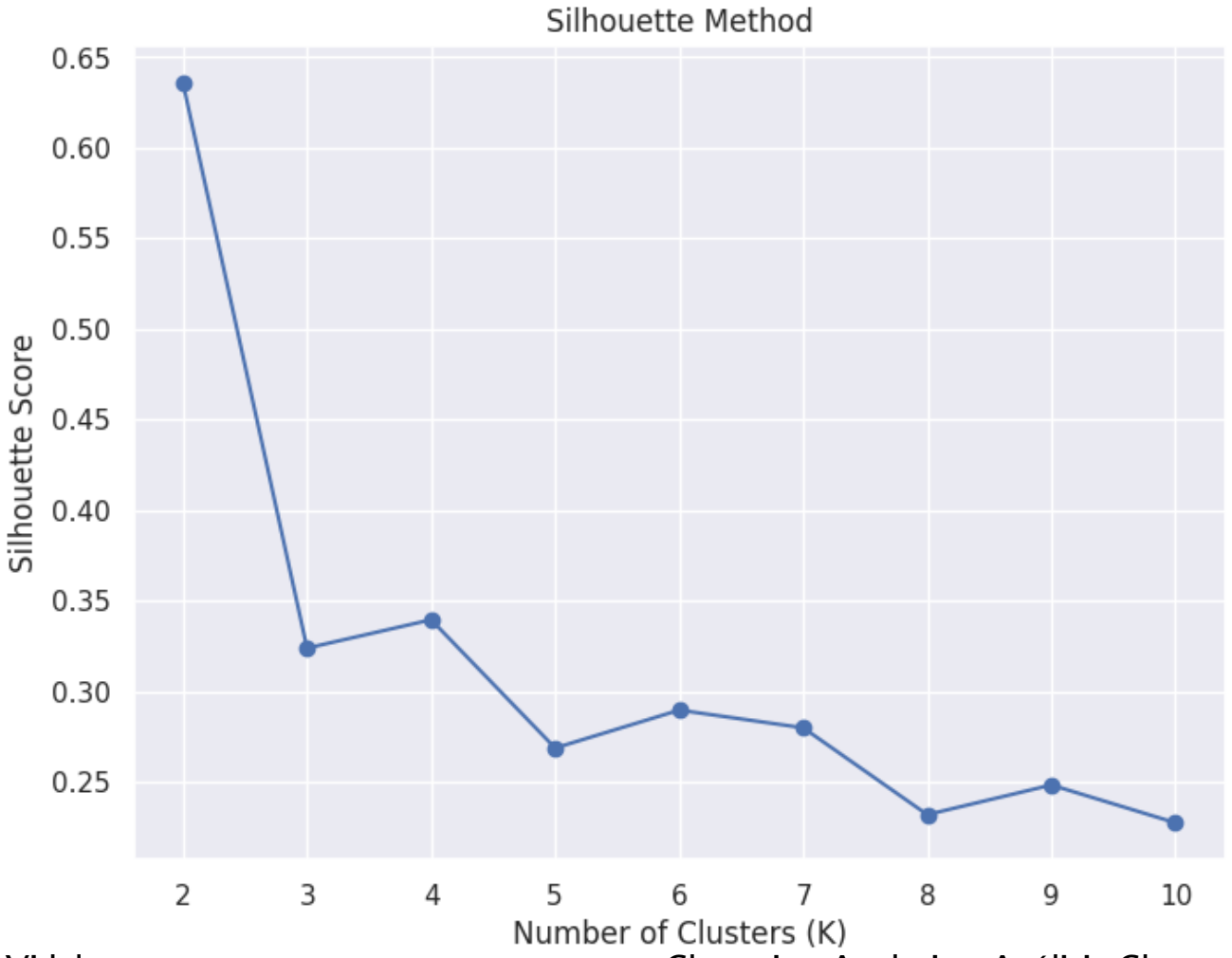
El método de la Silueta: Código I

```
# Tecnica para determinar el numero optimo de
    conglomerados utilizando puntuaciones de silueta
    .

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
silhouette_scores = []

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

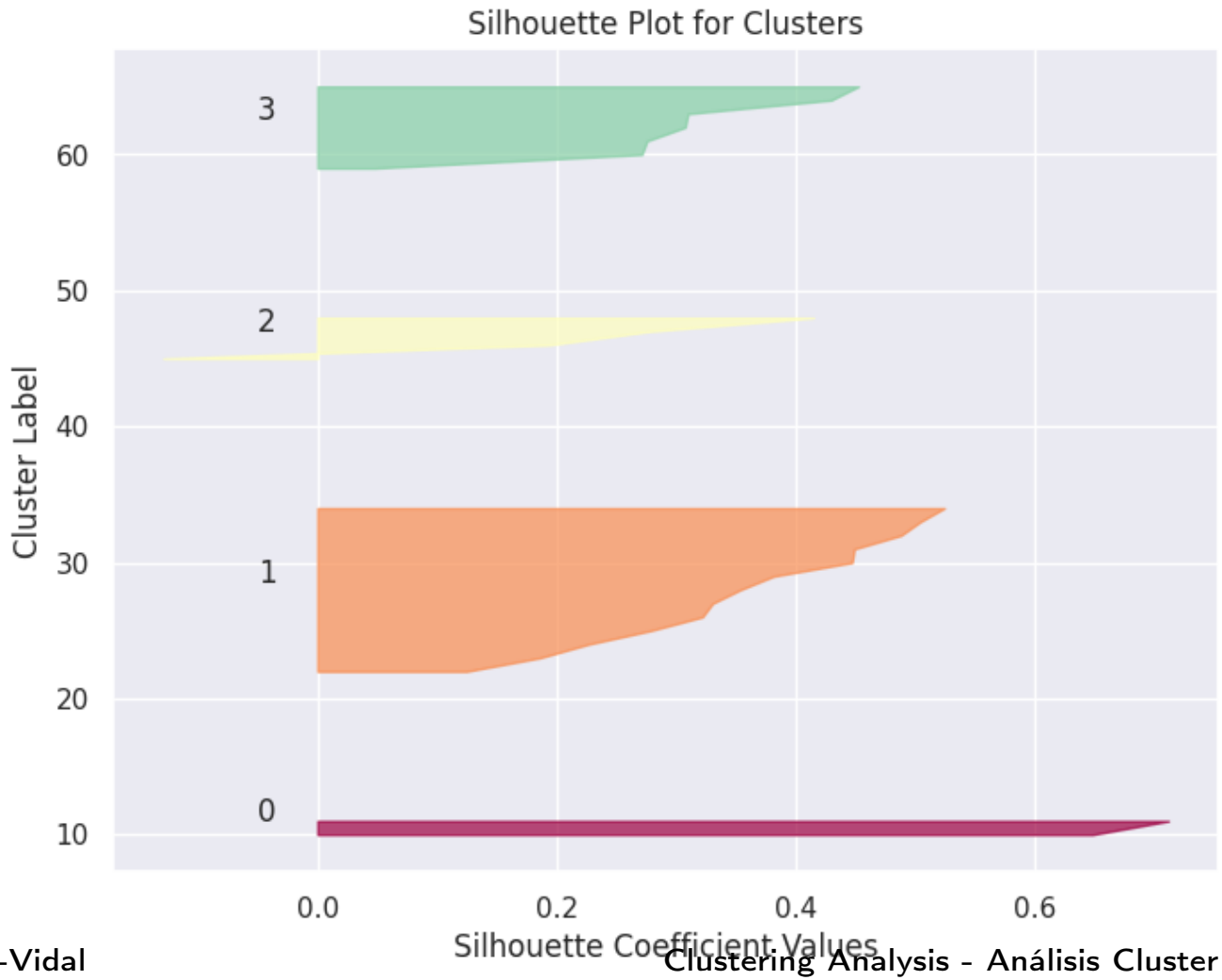
El método de la Silueta: Código II

```
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()
```

El método de la Silueta



Visualización de puntuaciones de silueta para cada grupo



Step 1: Cluster Labels Assignment

```
# 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
```

PAIS	
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

Step 2: Cluster Centroids Analysis

```
# 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: Centroides de los cluster (datos estandarizados)

Label	*m*0	*m*25	*m*50	*m*75	*w*0	*w*25	*w*50	*w*75
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	-0.11	-0.36	-0.54	-0.76	-0.17	-0.51	-0.64	-0.71

Step 3: Original Data Analysis

```
# 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: Centroides de cada cluster (original)

Label	$m0$	$m25$	$m50$	$m75$	$w0$	$w25$	$w50$	$w75$	Cluster4
0	36.0	29.5	15.0	6.0	38.0	33.0	18.5	6.5	1
1	62.46	45.23	24.15	8.54	67.46	49.54	27.23	10.54	4
2	62.0	48.75	27.5	12.25	66.0	52.5	31.25	14.5	2
3	58.0	41.86	21.29	6.86	62.0	44.86	24.14	8.29	3

Conclusión

- Los grupos 1 a 3 muestran una mayor esperanza de vida en ambos sexos.
- Los grupos 1 y 2 tienen la mayor esperanza de vida al nacer.
- En la vejez, el cluster 2 supera a los demás (12 años más que los cluster 1 y 3).
- El grupo 0 exhibe la esperanza de vida más baja.
- El análisis de las estadísticas de datos originales mejora la interpretación de las diferencias de los conglomerados.
- Es importante considerar los países dentro de cada grupo para una caracterización integral.