Program_TP2_Grupo4

October 17, 2025

1 ==========

2 TP2 — Taller de Programación (UBA 2025)

3 Grupo 4: Ángel Zapata, David Robalino y Federico Kisza

4 Docente: María Noelia Romero

5 ----------

6 Objetivo: Aplicar métodos no supervisados y visualización

7 sobre la base EPH Patagónica 2005–2025.

8 ==========

[1]:
```
# === 1) Librerías ===
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import linkage, dendrogram

# Configuración general
pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:,.2f}'.format
sns.set(style="whitegrid", palette="deep")
```

[2]:
```
# === 2) Cargar base limpia ===
import os
import pandas as pd
```

```
# Ruta actualizada a la carpeta del TP2
DATOS = r"C:\Users\econz\OneDrive\Documentos\Lecturas libros\Cursos␣
 ↪UBA\Programación\Grupo4_UBA_2025\TP2"
ruta = os.path.join(DATOS, "base_total_patagonica_limpia.csv")

# Cargar base
eph = pd.read_csv(ruta)

# Verificación rápida
print("Base cargada correctamente.")
print("Filas y columnas:", eph.shape)
print("\nVista previa:")
display(eph.head(3))
```

```
Base cargada correctamente.
Filas y columnas: (8588, 24)

Vista previa:
```

	ano4	codusu	nro_hogar	ch04	ch06	ch07	\
0	2005	125814	1.00	Varón	46.00	Casado	
1	2005	125814	1.00	Mujer	32.00	Casado	
2	2005	125814	1.00	Varón	14.00	Soltero	

	ch08	nivel_ed	estado	\
0	Obra social (incluye PAMI)	Secundaria Incompleta	Ocupado	
1	Obra social (incluye PAMI)	Secundaria Incompleta	Inactivo	
2	Obra social (incluye PAMI)	Primaria Completa	Inactivo	

	cat_ocup	cat_inac	itf	ipcf	p21	p47t	pp07h	\
0	Obrero o empleado	0.0	2,400.00	480.00	2,400.00	2,400.00	Sí	
1	0.0	Ama de casa	2,400.00	480.00	0.00	0.00	0.0	
2	0.0	Estudiante	2,400.00	480.00	0.00	0.00	0.0	

	pp03c	pp03g	edad2	educ	horastrab	\
0	…un sólo empleo/ocupación/actividad?	No	2,116.00	NaN	NaN	
1	0.0	0.0	1,024.00	NaN	NaN	
2	0.0	0.0	196.00	NaN	NaN	

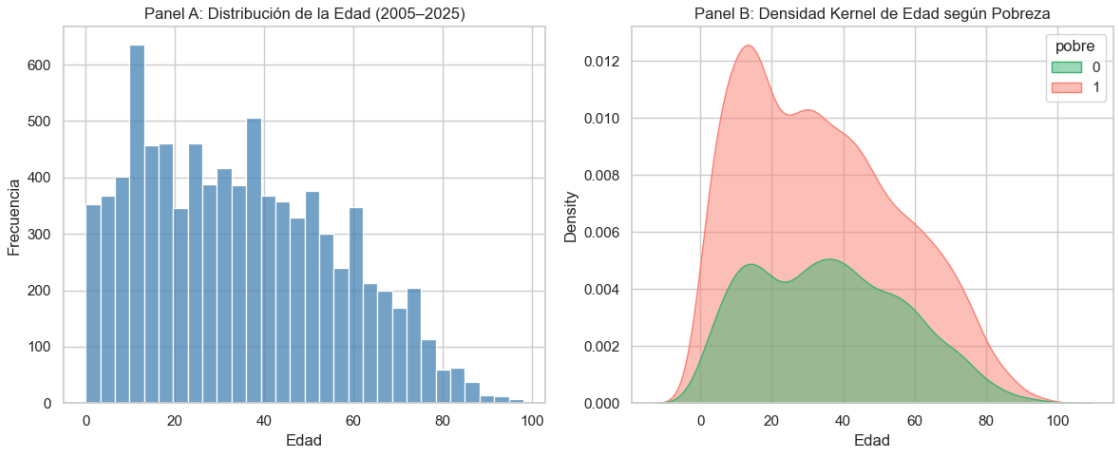
	itf_2025	linea_pobreza	pobre
0	25,200.00	1,278,119.50	1
1	25,200.00	1,278,119.50	1
2	25,200.00	1,278,119.50	1

[3]:
```
# === 3) Revisión general ===
print("Años disponibles:", eph["ano4"].unique())
print("\nValores faltantes por variable (top 10):")
print(eph.isna().mean().sort_values(ascending=False).head(10))
```

```
Años disponibles: [2005 2025]

Valores faltantes por variable (top 10):
horastrab       1.00
educ            0.38
pp03g           0.35
pp03c           0.35
pp07h           0.35
p47t            0.07
p21             0.04
ano4            0.00
codusu          0.00
linea_pobreza   0.00
dtype: float64
```

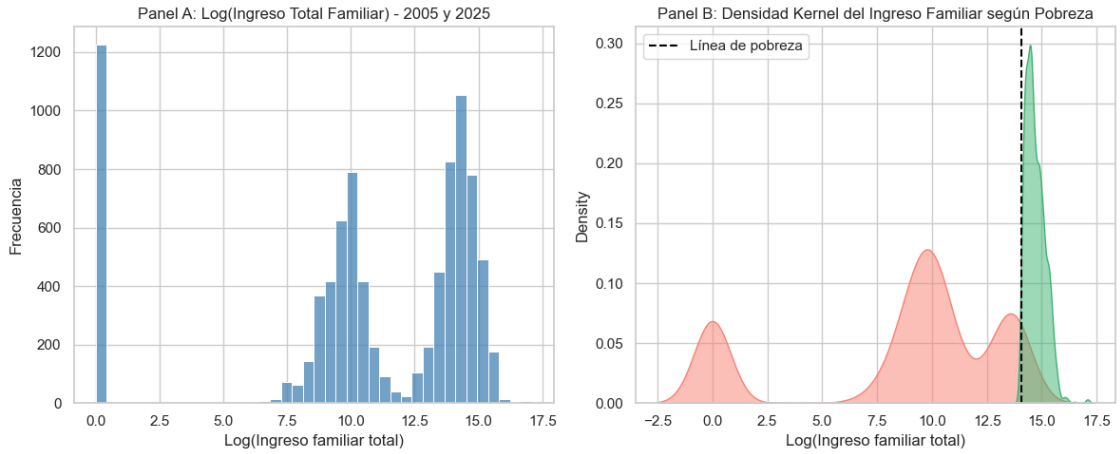
[4]:
```
# === 4) Visualización: Edad ===
plt.figure(figsize=(12,5))

# Panel A: Histograma de edad
plt.subplot(1,2,1)
sns.histplot(eph["ch06"], bins=30, color="steelblue", kde=False)
plt.title("Panel A: Distribución de la Edad (2005-2025)")
plt.xlabel("Edad")
plt.ylabel("Frecuencia")

# Panel B: Densidad Kernel por pobreza
plt.subplot(1,2,2)
sns.kdeplot(data=eph, x="ch06", hue="pobre", fill=True,
            palette={0:"mediumseagreen",1:"salmon"}, alpha=0.5)
plt.title("Panel B: Densidad Kernel de Edad según Pobreza")
plt.xlabel("Edad")

plt.tight_layout()
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[5]:
```
# === 5) Visualización: Ingreso familiar ===
plt.figure(figsize=(12,5))

# Panel A: Histograma
plt.subplot(1,2,1)
sns.histplot(np.log1p(eph["itf_2025"]), bins=40, color="steelblue", kde=False)
plt.title("Panel A: Log(Ingreso Total Familiar) - 2005 y 2025")
plt.xlabel("Log(Ingreso familiar total)")
plt.ylabel("Frecuencia")

# Panel B: Kernel por pobreza
plt.subplot(1,2,2)
sns.kdeplot(data=eph, x=np.log1p(eph["itf_2025"]), hue="pobre", fill=True,
            palette={0:"mediumseagreen",1:"salmon"}, alpha=0.5)
plt.axvline(np.log1p(eph["linea_pobreza"].iloc[0]), color="black", ls="--",␣
 ↪lw=1.5, label="Línea de pobreza")
plt.legend()
plt.title("Panel B: Densidad Kernel del Ingreso Familiar según Pobreza")
plt.xlabel("Log(Ingreso familiar total)")

plt.tight_layout()
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[6]:
```
# === 6) Descriptivos básicos ===
tabla_desc = eph[["ch06","educ","itf_2025","horastrab"]].describe().T
print("Tabla descriptiva general:\n")
print(tabla_desc)
```

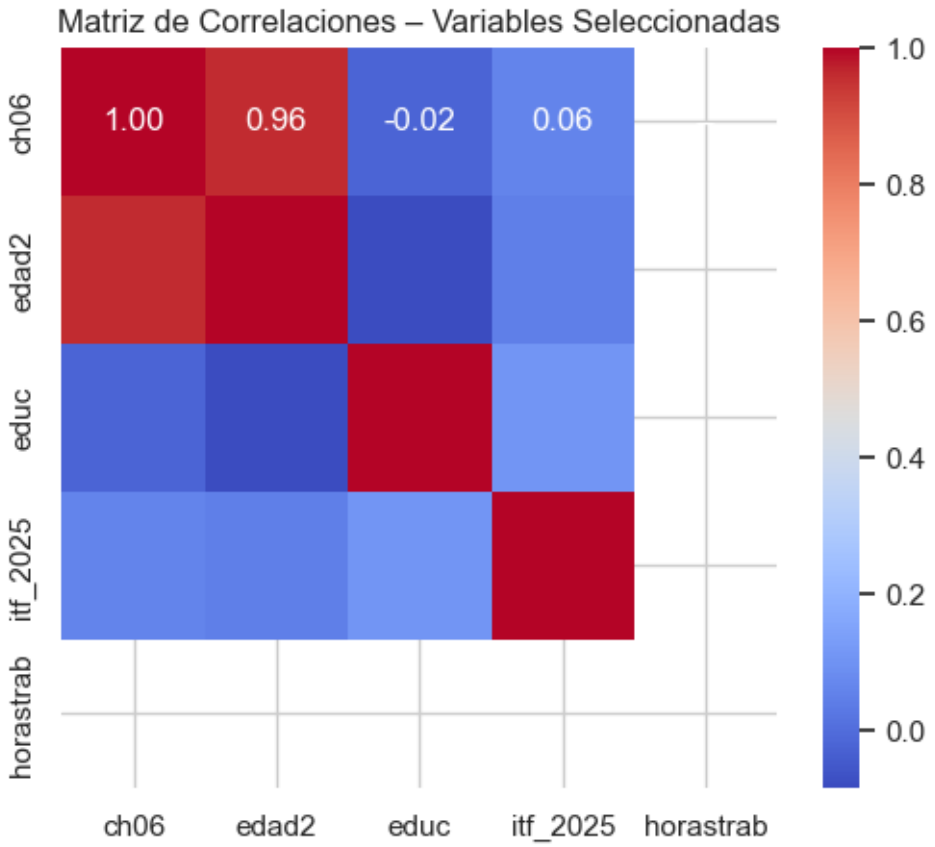
```
Tabla descriptiva general:

              count       mean          std  min       25%       50%  \
ch06      8,588.00      34.65        21.70 0.00     16.00     33.00
educ      5,359.00       7.89         4.71 0.00      6.00      9.00
itf_2025  8,588.00 975,431.49 1,525,597.37 0.00 10,500.00 59,850.00
horastrab     0.00        NaN          NaN  NaN       NaN       NaN

                  75%           max
ch06            51.00         98.00
educ            12.00         16.00
itf_2025  1,600,000.00 26,400,000.00
horastrab          NaN           NaN
```

[7]:
```
# === 7) Tabla resumen por año ===
tabla_1 = eph.groupby("ano4").agg(
    obs_totales = ("ano4", "count"),
    pobres = ("pobre", "sum"),
    no_pobres = ("pobre", lambda x: (x==0).sum()),
    edad_prom = ("ch06", "mean"),
    educ_prom = ("educ", "mean"),
    ingreso_prom = ("itf_2025", "mean")
).reset_index()

tabla_1["tasa_pobreza_%"] = 100 * tabla_1["pobres"] / tabla_1["obs_totales"]
print(tabla_1)
```

```
   ano4  obs_totales  pobres  no_pobres  edad_prom  educ_prom  ingreso_prom  \
0  2005         3229    3229          0      29.69        NaN     21,201.32
1  2025         5359    2732       2627      37.63       7.89  1,550,391.22

   tasa_pobreza_%
0          100.00
1           50.98
```

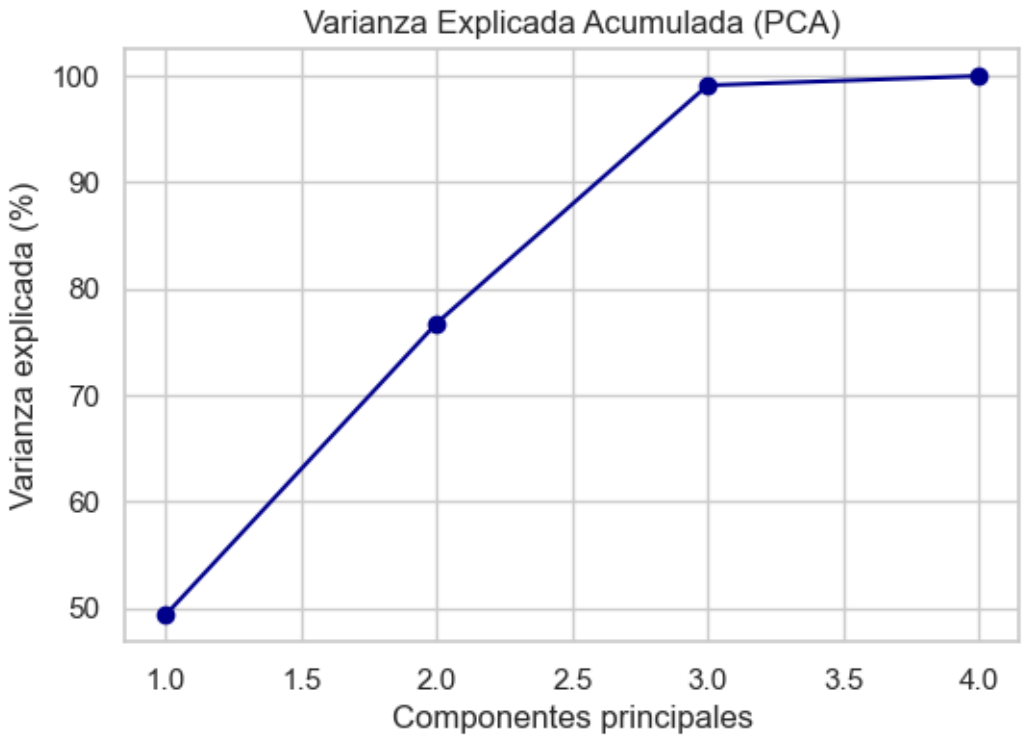
[8]:
```
# === 8) Matriz de correlaciones ===
vars_corr = ["ch06","edad2","educ","itf_2025","horastrab"]
corr = eph[vars_corr].corr()

plt.figure(figsize=(6,5))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Matriz de Correlaciones - Variables Seleccionadas")
plt.show()
```

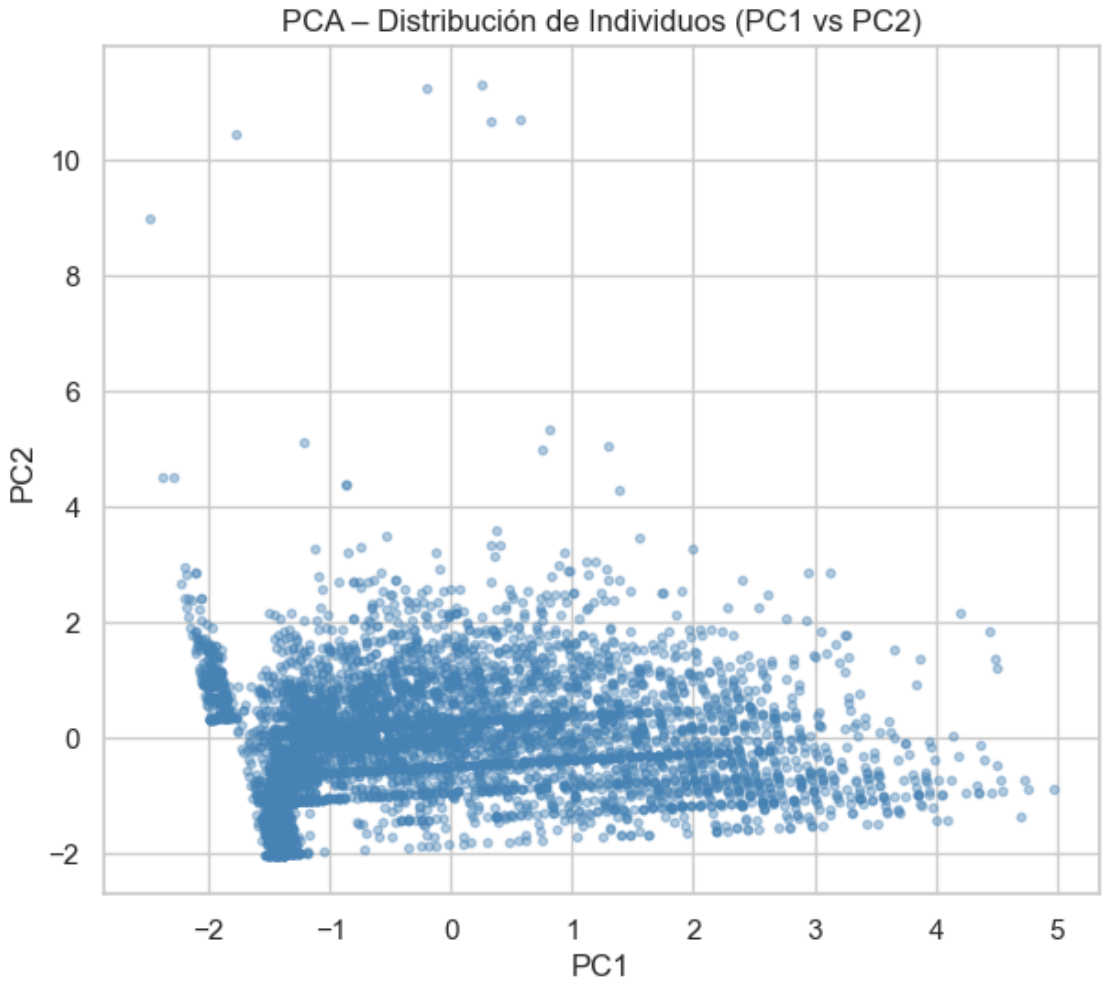
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\matrix.py:260: FutureWarning:
Format strings passed to MaskedConstant are ignored, but in future may error or
produce different behavior
  annotation = ("{:" + self.fmt + "}").format(val)
```



[9]:
```
# === 9) PCA (corregido) ===

# Variables sin tantos faltantes
vars_pca = ["ch06","edad2","educ","itf_2025"]

# Verificamos cuántas filas completas hay
print("Cantidad de filas con datos completos:", eph[vars_pca].dropna().shape[0])

# Eliminamos filas con NaN solo en estas columnas
X = eph[vars_pca].dropna()

# Escalamos las variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Ajustamos PCA
pca = PCA(n_components=4)
X_pca = pca.fit_transform(X_scaled)

# Scree Plot
plt.figure(figsize=(6,4))
plt.plot(range(1,5), np.cumsum(pca.explained_variance_ratio_)*100, marker='o',␣
 ↪color='darkblue')
plt.title("Varianza Explicada Acumulada (PCA)")
plt.xlabel("Componentes principales")
plt.ylabel("Varianza explicada (%)")
plt.grid(True)
plt.show()

# Biplot: PC1 vs PC2
plt.figure(figsize=(7,6))
plt.scatter(X_pca[:,0], X_pca[:,1], s=10, alpha=0.4, color='steelblue')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("PCA - Distribución de Individuos (PC1 vs PC2)")
plt.show()
```

```
Cantidad de filas con datos completos: 5359
```


Varianza Explicada Acumulada (PCA)
Varianza explicada (%)
100
90
80
70
60
50
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Componentes principales



[10]:
```
# === 10) Loadings de PCA (ajustado a 4 variables) ===
loadings = pd.DataFrame(
    pca.components_.T,
    index=vars_pca,
    columns=[f"PC{i+1}" for i in range(pca.n_components_)]
)

print("Cargas (loadings) de las variables en los componentes principales:\n")
display(loadings.round(3))
```

```
Cargas (loadings) de las variables en los componentes principales:
```

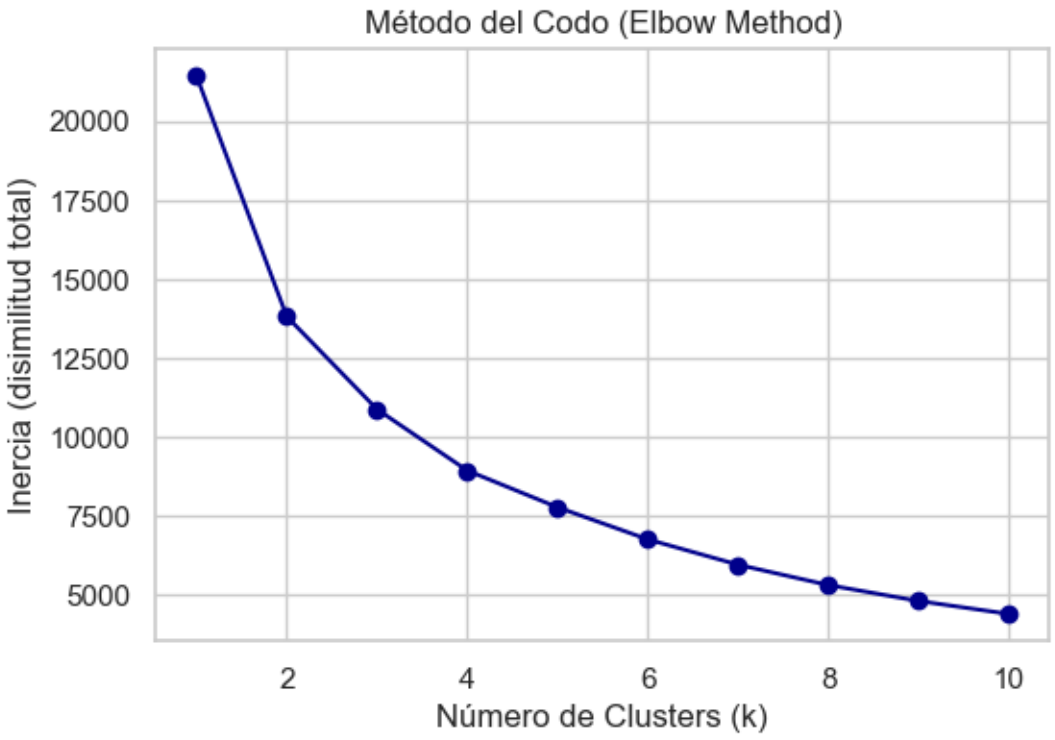
	PC1	PC2	PC3	PC4
ch06	0.70	0.10	-0.02	-0.70
edad2	0.70	0.05	0.01	0.71
educ	-0.08	0.70	-0.71	0.04

```
itf_2025 -0.07 0.70  0.71  0.01
```

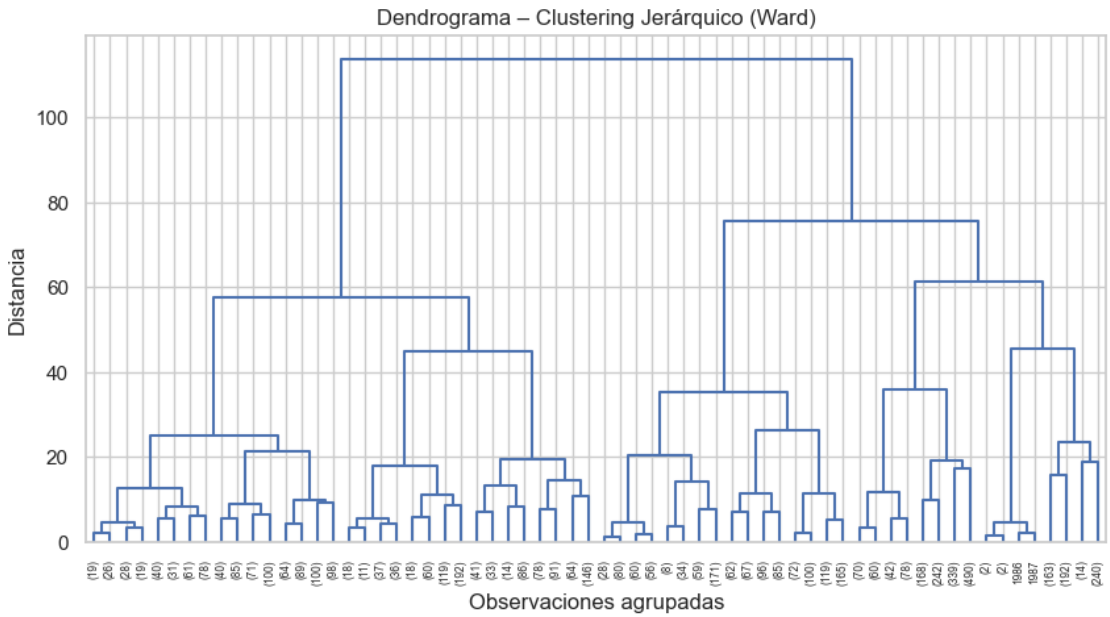
[ ]:
```python
# === 11) K-Means (con las mismas 4 variables del PCA) ===
from sklearn.cluster import KMeans

# Usamos las mismas variables que en el PCA
X = eph[["ch06","edad2","educ","itf_2025"]].dropna()

# Escalamos nuevamente (igual que en el PCA)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Ajustamos K-Means para distintos k
kmeans_results = {}
for k in [2, 3, 4]:
    kmeans = KMeans(n_clusters=k, n_init=20, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    eph[f"cluster_{k}"] = np.nan
    eph.loc[X.index, f"cluster_{k}"] = labels
    kmeans_results[k] = kmeans.inertia_

# Visualización para k=3 (ejemplo principal)
pca_plot = PCA(n_components=2)
coords = pca_plot.fit_transform(X_scaled)

plt.figure(figsize=(7,5))
sns.scatterplot(x=coords[:,0], y=coords[:,1],
                hue=eph.loc[X.index,"cluster_3"].astype(int),
                palette="Set2", s=25, alpha=0.7)
plt.title("Clustering (K=3) - Espacio PCA reducido (2D)")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.legend(title="Cluster")
plt.show()
```

[12]:
```python
# === 12) Método del Codo ===
ks = range(1, 11)
inertias = []

for k in ks:
    km = KMeans(n_clusters=k, n_init=20, random_state=42)
    km.fit(X_scaled)
    inertias.append(km.inertia_)

plt.figure(figsize=(6,4))
plt.plot(ks, inertias, marker='o', color='darkblue')
plt.title("Método del Codo (Elbow Method)")
```

```
plt.xlabel("Número de Clusters (k)")
plt.ylabel("Inercia (disimilitud total)")
plt.grid(True)
plt.show()
```



[13]:
```
# === 13) Clustering jerárquico ===
from scipy.cluster.hierarchy import linkage, dendrogram

Z = linkage(X_scaled, method='ward')

plt.figure(figsize=(10,5))
dendrogram(Z, truncate_mode='level', p=5, color_threshold=0)
plt.title("Dendrograma - Clustering Jerárquico (Ward)")
plt.xlabel("Observaciones agrupadas")
plt.ylabel("Distancia")
plt.show()
```



[14]:
```
# === 14) Guardar base con resultados ===
ruta_salida = os.path.join(DATOS, "base_TP2_resultados.csv")
eph.to_csv(ruta_salida, index=False, encoding="utf-8-sig")

print("  Resultados guardados en:", ruta_salida)
```

```
 Resultados guardados en: C:\Users\econz\OneDrive\Documentos\Lecturas
libros\Cursos UBA\Programación\Grupo4_UBA_2025\TP2\base_TP2_resultados.csv
```

[15]:
```
# === Exportar notebook a PDF ===
!jupyter nbconvert --to pdf "Program_TP2_Grupo4.ipynb"
```

```
[NbConvertApp] Converting notebook Program_TP2_Grupo4.ipynb to pdf
[NbConvertApp] Support files will be in Program_TP2_Grupo4_files\
[NbConvertApp] Making directory .\Program_TP2_Grupo4_files
[NbConvertApp] Writing 62353 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | b had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 464827 bytes to Program_TP2_Grupo4.pdf
```

[ ]: