

# Generar Mapas Coropléticos en Python\*

Aplicación de Geopandas en Perú

Facundo Cabral

Los mapas coropléticos (choropleth maps) son un gráfico importante en análisis de datos en temas económicos, políticos, sociales, entre otros. Esta aplicación muestra como realizar un gráfico de este estilo. Haremos una réplica de un gráfico presentado en el trabajo [Mujer Emprendedora: Indicadores de desempeño 2024](#) producto del trabajo de la Oficina General de Evaluación de Impacto y Estudios Económicos del [Ministerio de la Producción \(PRODUCE\)](#) del Perú.

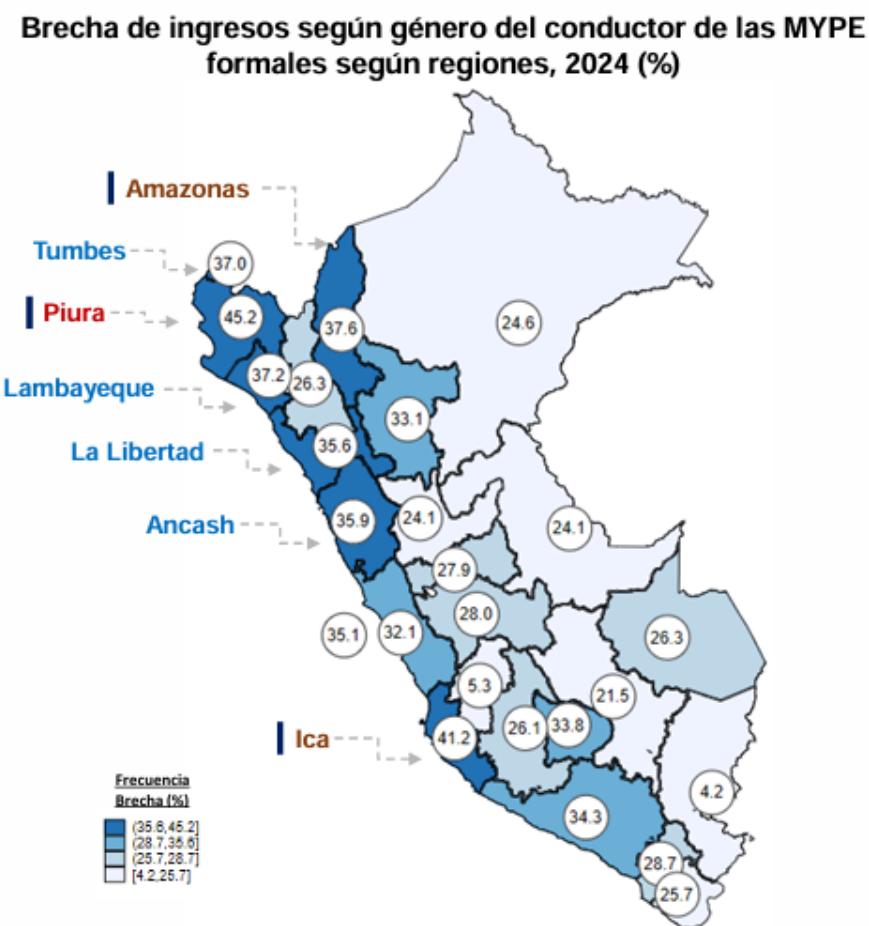


Figura 1: Gráfico del informe de PRODUCE

\* El proyecto completo en mi perfil de [Github](#)

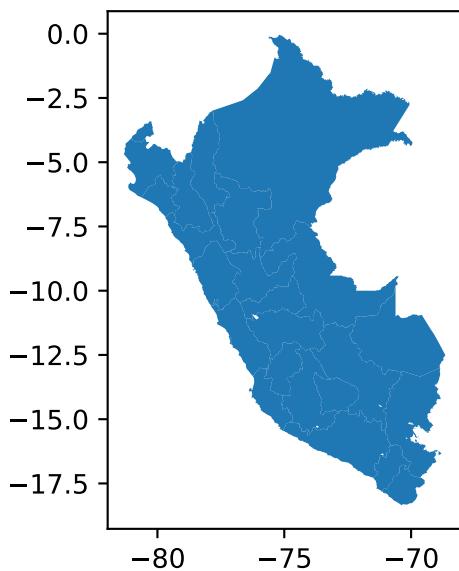
Primero importamos las librerías necesarias para la aplicación.

```
import pandas as pd          # Manejo de Data Frames
import geopandas as gpd       # Manejo de Data Frames espaciales
import matplotlib.pyplot as plt # Visualización | Principal
import matplotlib.colors as mcolors # Visualización | Complemento
import matplotlib.patches as mpatches # Visualización | Complemento
from shapely.geometry import Point # Cambio de coordenadas
```

A continuación leemos el shapefile y hacemos el ploteo. Este es el mapa base.

```
gdf = gpd.read_file("zip://peru_regions.zip")
gdf = gdf[['NOMBDEP','geometry']]
gdf.plot()
print(type(gdf))
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
```



La variable `gdf` no deja de ser un Data Frame, por lo que, para poder hacer una diferenciación entre las regiones, le agregamos variables numéricas y pintamos las áreas según los valores. Para ello, realizaremos la unión con otro data frame.

```
data = pd.read_csv("income_gap.csv", sep=";")
gdf = gdf.merge(data, left_on="NOMBDEP", right_on="NOMBDEP", how="left")
print(gdf.head(10))
```

	NOMBDEP	geometry	IncomeGap
0	AMAZONAS	POLYGON ((-77.81211 -2.98962, -77.81332 -2.990...	37.6
1	ANCASH	POLYGON ((-77.64692 -8.05086, -77.64669 -8.052...	35.9
2	APURIMAC	POLYGON ((-73.74632 -13.17456, -73.7457 -13.17...	33.8
3	AREQUIPA	POLYGON ((-71.98109 -14.64062, -71.98093 -14.6...	34.3
4	AYACUCHO	POLYGON ((-74.34843 -12.17503, -74.35 -12.1758...	26.1
5	CAJAMARCA	POLYGON ((-78.70034 -4.62769, -78.69817 -4.629...	26.3
6	CALLAO	POLYGON ((-77.13521 -11.81782, -77.13516 -11.8...	35.1
7	CUSCO	POLYGON ((-72.9728 -11.25189, -72.97134 -11.25...	21.5
8	HUANCAVELICA	POLYGON ((-74.57118 -11.9871, -74.57095 -11.98...	5.3
9	HUANUCO	POLYGON ((-76.00486 -8.30517, -76.00301 -8.305...	24.1

Antes de realizar el ploteo, debemos generar los parámetros necesarios para el gráfico, como son los intervalos de valores, los colores de sombreado y las etiquetas para la leyenda.

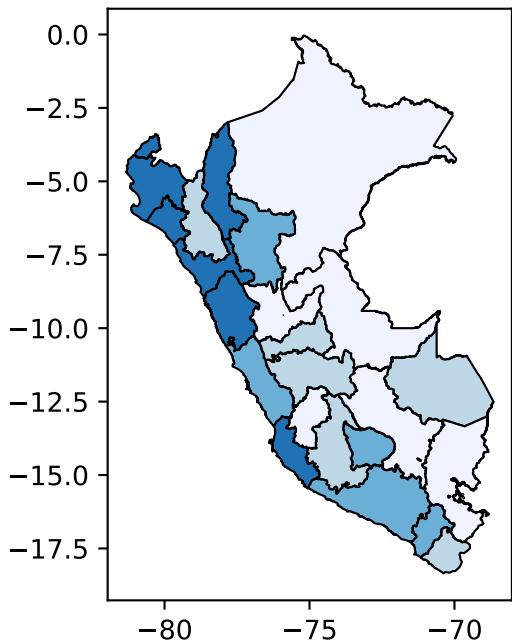
```
bins    = [4.2, 25.7, 28.7, 35.6, 45.2]
labels  = ["[4.2-25.7)", "[25.7-28.7)", "[28.7-35.6)", "[35.6-45.2]"]
colors  = ["#EFF3FF", "#BDD7E7", "#6BAED6", "#2171B5"]
cmap    = mcolors.ListedColormap(colors)
norm    = mcolors.BoundaryNorm(bins, cmap.N)

legend_patches = [
    mpatches.Patch(
        facecolor=colors[i],
        label=labels[i],
        edgecolor="black",
        linewidth=0.5
    )
    for i in range(len(labels))
]
legend_patches = legend_patches[::-1]
```

A continuación se hace el ploteo base coloreando las regiones según los valores de los intervalos.

```
fig, ax = plt.subplots(figsize=(4, 4))

gdf.plot(
    column    = "IncomeGap",
    cmap      = cmap,
    norm      = norm,
    linewidth = 0.8,
    edgecolor = "black",
    ax         = ax
)
```



Lo que continúa es el mismo ploteo con una serie de arreglos comentados.

```
fig, ax = plt.subplots(figsize=(10, 10))
gdf.plot(
    column      = "IncomeGap",
    cmap        = cmap,
    norm        = norm,
    linewidth   = 0.8,
    edgecolor   = "black",
    ax          = ax
)

ax.set_axis_off() # Quitar los ejes

ax.legend(
    handles=legend_patches,
    title="Income Gap (%)",
    loc="lower left",
    frameon=False,
    fontsize=10,
    title_fontsize=12
) # Añadir la leyenda de los intervalos

# Cálculo de los puntos medios de las regiones para las etiquetas
# Callao se desplaza dado que se sobrepone con la de Lima
gdf["label_point"] = gdf.geometry.representative_point()
gdf.loc[gdf["NOMBDEP"] == "CALLAO", "label_point"] = Point(-78, -11.94856)

ax.scatter(      # Graficar los círculos blancos
    gdf["label_point"].x,
    gdf["label_point"].y,
    s=750,
    facecolor="white",
    edgecolor="black",
    linewidth=1,
    zorder=3
)

for x, y, val in zip(      # Colocar los valores
    gdf["label_point"].x,
    gdf["label_point"].y,
    gdf["IncomeGap"]
):
    ax.text(
        x, y,
        f"{val:.1f}",
        ha="center",
        va="center",
        fontsize=10,
        color="black",
        zorder=4
)
```

