World Indicators

Introduction

The analysis period spans from 1990 to 2024 on a global scale, but focuses on countries that allow for relevant comparisons, particularly to situate Chile within the Latin American context.

The aim is to contrast opinions with objective data on Chile, especially when it is described as a leading country in the region. Likewise, it examines the demands for greater state funding—based on the argument that Chile has sufficient resources—versus those who claim that resources are limited and advocate for private sector involvement to meet social demands.

In [204…
```
#Libraries to use
import pandas as pd
from pandas_datareader import wb
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict
from matplotlib.patches import Patch
import warnings
```

Países del Banco Mundial

In [206…
```
countries = wb.get_countries()
countries.head(2)
```

Out[206…

	iso3c	iso2c	name	region	adminregion	incomeLevel	lendingType
0	ABW	AW	Aruba	Latin America & Caribbean		High income	Not classified
1	AFE	ZH	Africa Eastern and Southern	Aggregates		Aggregates	Aggregates

In [207…
```
# Contar países por regiones
conteo_paises_region = countries['region'].value_counts().reset_index()
conteo_paises_region.columns = ['region', 'número_de_países']
print(conteo_paises_region)
```

```
                      region  número_de_países
0                 Aggregates                79
1       Europe & Central Asia                58
2         Sub-Saharan Africa                 48
3  Latin America & Caribbean                 42
4         East Asia & Pacific                37
5  Middle East & North Africa                21
6                  South Asia                 8
7               North America                 3
```

In [208…
```
# Contar países por su clasificación
conteo_paises_clasificacion = countries['incomeLevel'].value_counts().reset_
print(conteo_paises_clasificacion)
```

```
           incomeLevel  count
0          High income     85
1           Aggregates     79
2  Upper middle income     54
3  Lower middle income     51
4           Low income     26
5       Not classified      1
```

Países

Aquí se detalla el país o los países utilizados en el análisis. Se crearan listas según se va avanzando el analisis y se requiera revisar algun conjunto de países, etc.

In [210…
```
#Primer grupo. Países que son parte de la OCDE

#Lista de países (actualizada a 2023).
oecd_countries = [
    'Australia', 'Austria', 'Belgium', 'Canada', 'Chile', 'Colombia', 'Costa
    'Czech Republic', 'Denmark', 'Estonia', 'Finland', 'France', 'Germany',
    'Hungary', 'Iceland', 'Ireland', 'Israel', 'Italy', 'Japan', 'Korea, Rep
    'Latvia', 'Lithuania', 'Luxembourg', 'Mexico', 'Netherlands', 'New Zeala
    'Norway', 'Poland', 'Portugal', 'Slovak Republic', 'Slovenia', 'Spain',
    'Sweden', 'Switzerland', 'Turkey', 'United Kingdom', 'United States']

#Segundo grupo. Países de América Central y del Sur

central_south_america = ['Belize', 'Costa Rica', 'El Salvador', 'Guatemala',
    'Honduras', 'Nicaragua', 'Panama', 'Antigua and Barbuda', 'Bahamas',
    'Barbados', 'Cuba', 'Dominica', 'Grenada', 'Haiti', 'Jamaica',
    'Dominican Republic', 'Saint Kitts and Nevis', 'Saint Vincent and the Gr
    'Saint Lucia', 'Trinidad and Tobago', 'Argentina', 'Bolivia', 'Brazil',
    'Colombia', 'Ecuador', 'Guyana', 'Paraguay', 'Peru', 'Suriname', 'Urugua
```

In [211…
```
type(central_south_america)
```

Out[211… `list`

Función para obtener los datos del indicador

In [213…

```python
def descargar_datos_wb(indicadores, paises, inicio, fin):
    dfs = []

    for nombre, indicador in indicadores.items():
        try:
            # Descargar datos para el indicador actual
            df = wb.download(
                indicator=indicador,
                country=paises,
                start=inicio,
                end=fin
            )

            df.reset_index(inplace=True)
            df['indicador'] = nombre  # Agregar columna con nombre del indic
            df.rename(columns={indicador: 'valor'}, inplace=True)

            dfs.append(df)

        except Exception as e:
            print(f"Error al descargar el indicador {nombre} ({indicador}):

    # Combinar todos los DataFrames
    if dfs:
        df_final = pd.concat(dfs, ignore_index=True)
        return df_final
    else:
        return pd.DataFrame()
```

Gross Domestic Product (GDP)

To analyze the relationship between GDP and fiscal spending, it is necessary to select the most appropriate indicator. For this purpose, the available indicators on the relevant portal will be reviewed to identify which one allows for a valid comparison with public expenditure. Additionally, the GDP measurement units will be assessed to choose the most suitable one for this analysis.

In [215…

```python
# Buscar indicadores relacionados con "gdp"
resultados = wb.search('gdp')

# Convertir a DataFrame
df_resultados = pd.DataFrame(resultados)

# Mostrar las primeras filas y también generar un archivo excel por una sola
#df_resultados.to_excel('GPD.xlsx')
df_resultados.head(3)
```

Out[215…

	id	name	unit	source	sourceNote	sourceOrganization
688	6.0.GDP_current	GDP (current $)		LAC Equity Lab	GDP is the sum of gross value added by all res...	b'World Development Indicators (World Bank)'
689	6.0.GDP_growth	GDP growth (annual %)		LAC Equity Lab	Annual percentage growth rate of GDP at market...	b'World Development Indicators (World Bank)'
690	6.0.GDP_usd	GDP (constant 2005 $)		LAC Equity Lab	GDP is the sum of gross value added by all res...	b'World Development Indicators (World Bank)'

In [216…
```
type(df_resultados)
```

Out[216… `pandas.core.frame.DataFrame`

The selection of the indicator **NY.GDP.MKTP.KD.ZG** refers to the Annual percentage growth rate of Gross Domestic Product (GDP) at market prices in constant local currency. The data, expressed as a percentage, will initiate an introductory research process that will strengthen the analysis due to the correct and timely choice of indicators, which will reflect the political and social economy of post-dictatorship Chile.

Comparative GDP Analysis: OECD Countries vs. Chile (1990-1993 vs. 2020-2023)

This study examines the evolution of Gross Domestic Product (GDP) in OECD member countries, comparing their performance with Chile's across two key periods: the early 1990s (1990-1993) and recent years (2020-2023). It is worth noting that some analyzed countries, including Chile, were not OECD members during the first period (Chile joined in 2010), allowing for an assessment of their economic trajectory both before and after joining the organization.

The analysis aims to identify:

- Comparative economic growth trends
- Chile's relative evolution compared to OECD countries
- Changes in growth patterns between both periods

In [219…
```
# Indicador seleccionado y los países del OCDE
indicadores = {
    'crecimiento del PIB': 'NY.GDP.MKTP.KD.ZG'
    }
```

```
paises = [
    'AUS', 'AUT', 'BEL', 'CAN', 'CHL', 'COL', 'CRI',
    'CZE', 'DNK', 'EST', 'FIN', 'FRA', 'DEU', 'GRC',
    'HUN', 'ISL', 'IRL', 'ISR', 'ITA', 'JPN', 'KOR',
    'LVA', 'LTU', 'LUX', 'MEX', 'NLD', 'NZL',
    'NOR', 'POL', 'PRT', 'SVK', 'SVN', 'ESP',
    'SWE', 'CHE', 'TUR', 'GBR', 'USA'
]

df_pib = descargar_datos_wb(indicadores, paises, '1990', '2023')
#df_pib.to_excel('pib.xlsx') #Exportar archivo para revisar la consistencia
df_pib.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1292 entries, 0 to 1291
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   country    1292 non-null   object
 1   year       1292 non-null   object
 2   valor      1285 non-null   float64
 3   indicador  1292 non-null   object
dtypes: float64(1), object(3)
memory usage: 40.5+ KB
```

```
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
```

In [221…

```
# Convertir 'year' a numérico (si aún no lo está)
df_pib['year'] = pd.to_numeric(df_pib['year'], errors='coerce')

# Filtrar y sumar valores para 1990-1993
df_90 = df_pib[(df_pib['year'] >= 1990) & (df_pib['year'] <= 1995)].groupby(
df_90 = df_90.rename(columns={'valor': 'sum_1990_1995'})  # Renombrar column

# Ordenar el dataframe por sum_1990_1995 de forma descendente y agregar rank
df_90 = df_90.sort_values('sum_1990_1995', ascending=False).reset_index(drop
df_90['ranking_1990_1995'] = df_90.index + 1  # +1 para que empiece en 1 en

# Filtrar y sumar valores para 2010-2015
df_20 = df_pib[(df_pib['year'] >= 2010) & (df_pib['year'] <= 2015)].groupby(
df_20 = df_20.rename(columns={'valor': 'sum_2010_2015'})  # Renombrar column

# Hacer un left join para mantener todas las filas de df_20
df_20 = pd.merge(
    df_20,
    df_90[['country', 'ranking_1990_1995']],  # Seleccionar solo las columna
    on='country',  # Columna común para unir
    how='left'     # Mantener todas las filas del dataframe izquierdo
)

# Filtrar y sumar valores para 2020-2023
df_2020 = df_pib[(df_pib['year'] >= 2020) & (df_pib['year'] <= 2023)].groupb
df_2020 = df_2020.rename(columns={'valor': 'sum_2020_2023'})  # Renombrar co
```

In [222…
```
df_90.head(2)
```

Out[222…

	country	sum_1990_1995	ranking_1990_1995
0	Korea, Rep.	52.614957	1
1	Chile	42.856943	2

In [223…
```
df_20.head(2)
```

Out[223…

	country	sum_2010_2015	ranking_1990_1995
0	Australia	15.931417	12
1	Austria	7.173292	14

In [224…
```
df_2020.head(2)
```

Out[224…

	country	sum_2020_2023
0	Australia	9.675955
1	Austria	2.800006

In [225…
```
#countries es el dataframe con los países del Banco Mundial.
#el cual tiene una variable llamada 'name' y que contiene los nombres de los
#y para complementar los dataframe anteriores: df_90 y df_20 con otros datos
#perentorio que alguna variable tenga el mismo nombre en todos los data df c
#sino se da el caso, se renombra alguna columna, la que es igual en todos lo
#name tiene los nombres de los países, pero en los otros data se llama esta
#por lo tanto, se le renombra con ese nombre 'country'

countries.rename(columns={'name': 'country'}, inplace=True)
```

In [226…
```
# Hacer un left join para mantener todas las filas de df_90
df_90_completo = pd.merge(
    df_90,
    countries[['country', 'region', 'incomeLevel']],  # Seleccionar solo las
    on='country',  # Columna común para unir
    how='left'     # Mantener todas las filas del dataframe izquierdo (df_90
)
df_90_completo.head(1)
```

Out[226…

	country	sum_1990_1995	ranking_1990_1995	region	incomeLevel
0	Korea, Rep.	52.614957	1	East Asia & Pacific	High income

In [227…
```
# Hacer un left join para mantener todas las filas de df_20
df_20_completo = pd.merge(
    df_20,
    countries[['country', 'region', 'incomeLevel']],  # Seleccionar solo las
    on='country',  # Columna común para unir
    how='left'     # Mantener todas las filas del dataframe izquierdo
```

```
)

df_20_completo.head(40)
```

Out[227…

	country	sum_2010_2015	ranking_1990_1995	region	incomeLevel
0	Australia	15.931417	12	East Asia & Pacific	High income
1	Austria	7.173292	14	Europe & Central Asia	High income
2	Belgium	8.402469	22	Europe & Central Asia	High income
3	Canada	13.832913	25	North America	High income
4	Chile	25.483988	2	Latin America & Caribbean	High income
5	Colombia	27.944112	6	Latin America & Caribbean	Upper middle income
6	Costa Rica	24.335052	5	Latin America & Caribbean	Upper middle income
7	Czechia	10.878536	33	Europe & Central Asia	High income
8	Denmark	7.663356	17	Europe & Central Asia	High income
9	Estonia	20.646183	36	Europe & Central Asia	High income
10	Finland	3.044536	31	Europe & Central Asia	High income
11	France	7.468081	23	Europe & Central Asia	High income
12	Germany	12.587471	13	Europe & Central Asia	High income
13	Greece	-25.609848	27	Europe & Central Asia	High income
14	Hungary	11.434913	35	Europe & Central Asia	High income
15	Iceland	10.752980	30	Europe & Central Asia	High income

	country	sum_2010_2015	ranking_1990_1995	region	incomeLevel
16	Ireland	39.076566	4	Europe & Central Asia	High income
17	Israel	24.166182	3	Middle East & North Africa	High income
18	Italy	-1.834415	26	Europe & Central Asia	High income
19	Japan	9.358411	19	East Asia & Pacific	High income
20	Korea, Rep.	22.069289	1	East Asia & Pacific	High income
21	Latvia	14.654743	37	Europe & Central Asia	High income
22	Lithuania	21.786006	38	Europe & Central Asia	High income
23	Luxembourg	14.519275	8	Europe & Central Asia	High income
24	Mexico	18.026779	16	Latin America & Caribbean	Upper middle income
25	Netherlands	5.817771	11	Europe & Central Asia	High income
26	New Zealand	16.223961	10	East Asia & Pacific	High income
27	Norway	9.532330	9	Europe & Central Asia	High income
28	Poland	18.972688	20	Europe & Central Asia	High income
29	Portugal	-2.679723	18	Europe & Central Asia	High income
30	Slovak Republic	19.510332	34	Europe & Central Asia	High income
31	Slovenia	3.184402	32	Europe & Central Asia	High income

	country	sum_2010_2015	ranking_1990_1995	region	incomeLevel
32	Spain	0.743118	21	Europe & Central Asia	High income
33	Sweden	16.343802	28	Europe & Central Asia	High income
34	Switzerland	12.022412	29	Europe & Central Asia	High income
35	Turkiye	43.925727	7	Europe & Central Asia	Upper middle income
36	United Kingdom	12.098123	24	Europe & Central Asia	High income
37	United States	14.135913	15	North America	High income

In [228…

```
# Hacer un left join para mantener todas las filas de df_20
df_2020_completo = pd.merge(
    df_2020,
    countries[['country', 'region', 'incomeLevel']],  # Seleccionar solo las
    on='country',  # Columna común para unir
    how='left'     # Mantener todas las filas del dataframe izquierdo
)

df_2020_completo.head(1)
```

Out[228…

	country	sum_2020_2023	region	incomeLevel
0	Australia	9.675955	East Asia & Pacific	High income

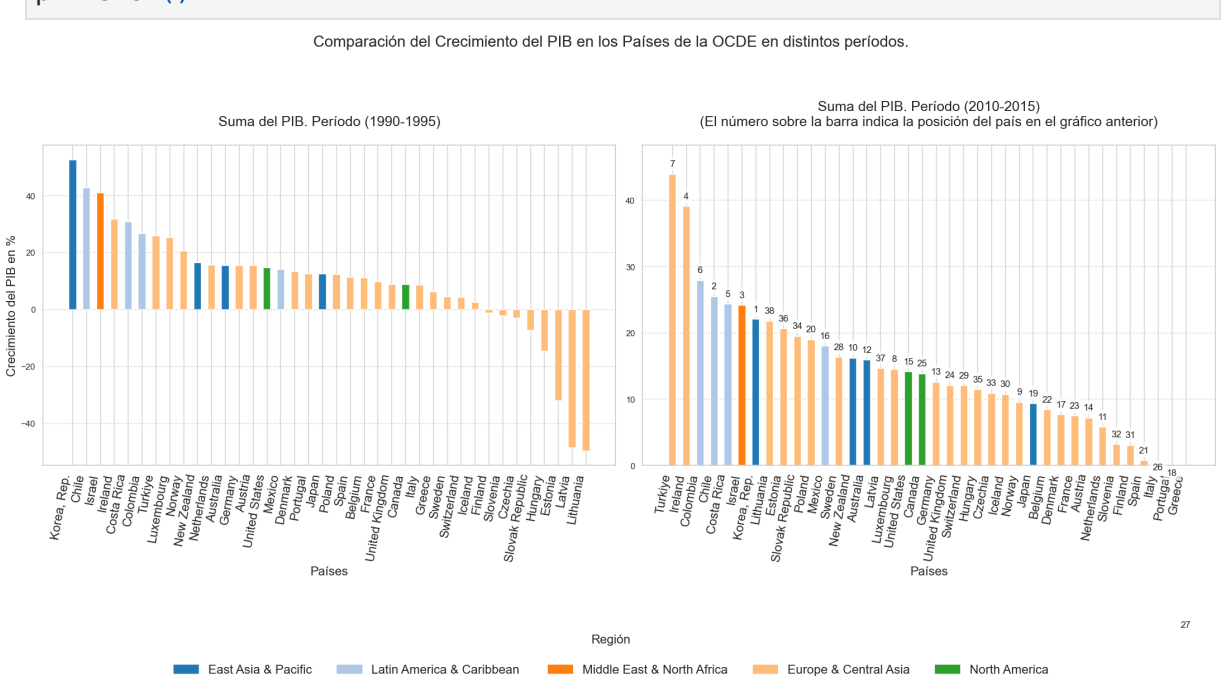
In [ ]:

In [229…

```
# Configuración de estilo y parámetros
bar_width = 0.7
space_between_bars = 0.3

# Configurar figura y subgráficos
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(22, 11))  # 1 fila, 2 columnas

# Añadir título principal centrado sobre ambos gráficos
fig.suptitle('Comparación del Crecimiento del PIB en los Países de la OCDE e
             fontsize=20, y=1.05)

# Ajustar diseño
plt.tight_layout()
fig.subplots_adjust(top=0.9, bottom=0.25)  # Ajustamos top para el título pr

# --- Definir colores únicos para todas las regiones ---
regiones_totales = pd.concat([df_90_completo['region'], df_20_completo['regi
```

```
colores = plt.cm.tab20.colors[:len(regiones_totales)]
color_por_region = dict(zip(regiones_totales, colores))

# --- Gráfico 1: 1990-1993 ---
df_sorted_90 = df_90_completo.sort_values('sum_1990_1995', ascending=False)
x_positions = np.arange(len(df_sorted_90['country'])) * (1 + space_between_b

bars1 = ax1.bar(
    x_positions,
    df_sorted_90['sum_1990_1995'],
    width=bar_width,
    color=[color_por_region[region] for region in df_sorted_90['region']]
)
ax1.set_title('Suma del PIB. Período (1990-1995)\n', fontsize=18)
ax1.set_xlabel('Países', fontsize=16)
ax1.set_ylabel('Crecimiento del PIB en %', fontsize=16)
ax1.set_xticks(x_positions)
ax1.set_xticklabels(df_sorted_90['country'], rotation=80, ha='right', fontsi
ax1.grid(axis='y', linestyle='--', alpha=0.7)

# --- Gráfico 2: 2020-2023 ---
df_sorted_20 = df_20_completo.sort_values('sum_2010_2015', ascending=False)
bars2 = ax2.bar(
    x_positions,
    df_sorted_20['sum_2010_2015'],
    width=bar_width,
    color=[color_por_region[region] for region in df_sorted_20['region']]
)

# Ajustar límites del eje Y para dar más espacio superior
max_valor = df_sorted_20['sum_2010_2015'].max()
ax2.set_ylim(0, max_valor * 1.10)  # Añade 10% de espacio extra arriba

# Resto de configuraciones del gráfico 2 (títulos, ejes, etc.)
ax2.set_title('Suma del PIB. Período (2010-2015)\n(El número sobre la barra
ax2.set_xlabel('Países', fontsize=16)
ax2.set_xticks(x_positions)
ax2.set_xticklabels(df_sorted_20['country'], rotation=80, ha='right', fontsi
ax2.grid(axis='y', linestyle='--', alpha=0.7)


# Agregar ranking de 1990-1993 sobre cada barra
for bar, valor, ranking in zip(bars2, df_sorted_20['sum_2010_2015'], df_sort
    ax2.text(
        bar.get_x() + bar.get_width()/2,
        valor + (0.02 * max(df_sorted_20['sum_2010_2015'])),
        str(int(ranking)),
        ha='center',
        va='bottom',
        fontsize=12,
        bbox=dict(facecolor='white', alpha=0.7, edgecolor='none')
    )

# --- Leyenda común debajo de ambos gráficos ---
handles = [plt.Rectangle((0, 0), 1, 1, color=color_por_region[region], label
           for region in regiones_totales]
```

```
# Ajustar diseño y añadir leyenda
plt.tight_layout()
fig.subplots_adjust(bottom=0.35)  # Aumentamos el espacio inferior para acom

# Crear leyenda más abajo
legend = fig.legend(
    handles=handles,
    title='Región\n',
    loc='lower center',
    bbox_to_anchor=(0.5, -0.01),  # Ajustamos posición vertical (más negativ
    ncol=min(len(regiones_totales), 5),
    frameon=False,
    prop={'size': 16},
    title_fontsize='16',
)

# Guardar figura asegurando que la leyenda no se corte
plt.savefig('comparacion_pib.png', bbox_extra_artists=(legend,), bbox_inches
plt.show()
```

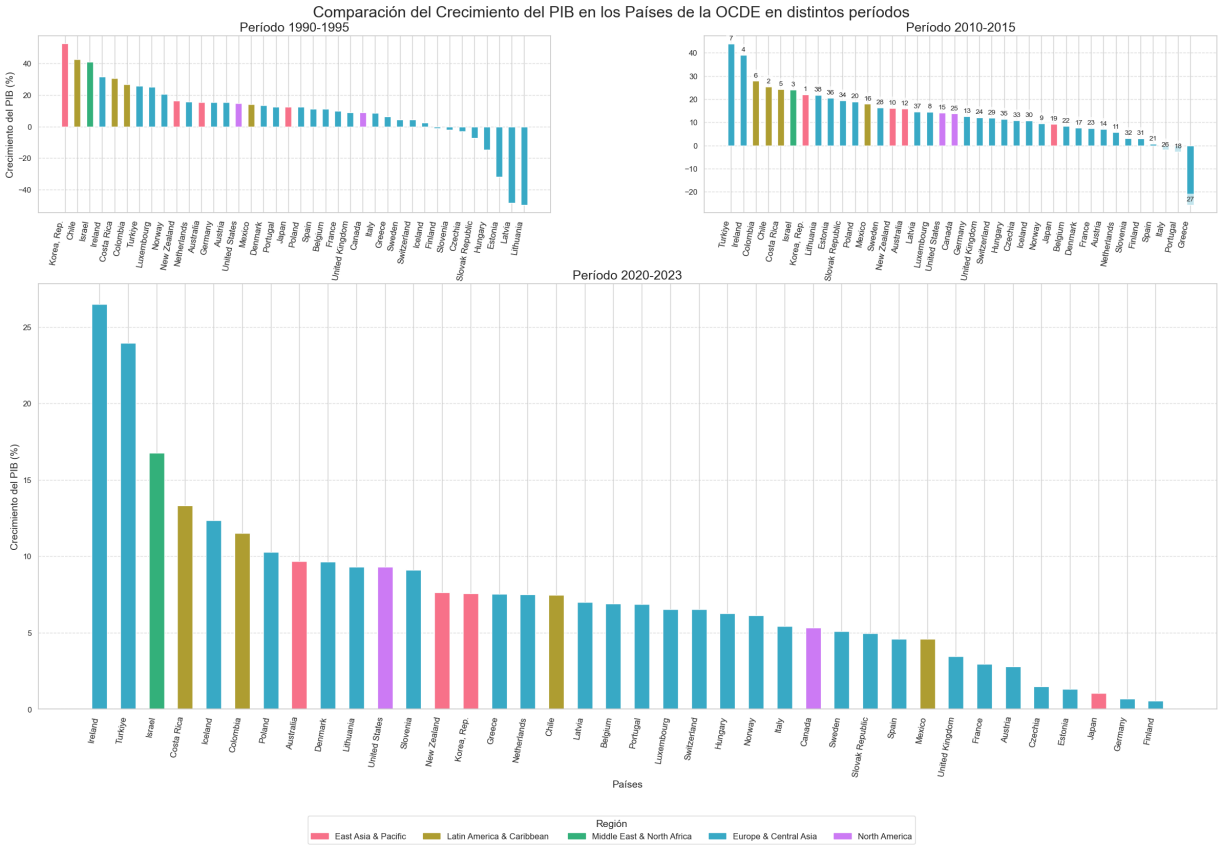


In [230…

```
# Configuración de estilo
sns.set_theme(style="whitegrid")

# Parámetros de visualización
bar_width = 0.7
space_between_bars = 0.3

# Crear figura con disposición personalizada
fig = plt.figure(figsize=(24, 16))

# Primer gráfico (arriba izquierda)
ax1 = plt.subplot2grid((3, 2), (0, 0), colspan=1)
# Segundo gráfico (arriba derecha)
ax2 = plt.subplot2grid((3, 2), (0, 1), colspan=1)
# Tercer gráfico (abajo centrado)
```

```
ax3 = plt.subplot2grid((3, 2), (1, 0), rowspan=2, colspan=2)

# Título principal
fig.suptitle('Comparación del Crecimiento del PIB en los Países de la OCDE e
             fontsize=22, y=0.98)

# --- Definir colores únicos para todas las regiones ---
regiones_totales = pd.concat([
    df_90_completo['region'].dropna(),
    df_20_completo['region'].dropna(),
    df_2020_completo['region'].dropna()
]).unique()

palette = sns.color_palette("husl", len(regiones_totales))
color_por_region = dict(zip(regiones_totales, palette))

# Posiciones x comunes (ajustar según necesidad)
x_positions = np.arange(len(df_90_completo['country'])) * (1 + space_betweer

# --- Gráfico 1: 1990-1995 ---
df_sorted_90 = df_90_completo.sort_values('sum_1990_1995', ascending=False)
bars1 = ax1.bar(
    x_positions,
    df_sorted_90['sum_1990_1995'],
    width=bar_width,
    color=[color_por_region.get(region, 'gray') for region in df_sorted_90['
)
ax1.set_title('Período 1990-1995', fontsize=18)
ax1.set_xlabel('Países', fontsize=14)
ax1.set_ylabel('Crecimiento del PIB (%)', fontsize=14)
ax1.set_xticks(x_positions)
ax1.set_xticklabels(df_sorted_90['country'], rotation=80, ha='right', fontsi
ax1.grid(axis='y', linestyle='--', alpha=0.7)

# --- Gráfico 2: 2010-2015 ---
df_sorted_20 = df_20_completo.sort_values('sum_2010_2015', ascending=False)
bars2 = ax2.bar(
    x_positions,
    df_sorted_20['sum_2010_2015'],
    width=bar_width,
    color=[color_por_region.get(region, 'gray') for region in df_sorted_20['
)
ax2.set_title('Período 2010-2015', fontsize=18)
ax2.set_xlabel('Países', fontsize=14)
ax2.set_xticks(x_positions)
ax2.set_xticklabels(df_sorted_20['country'], rotation=80, ha='right', fontsi
ax2.grid(axis='y', linestyle='--', alpha=0.7)

# Añadir rankings al segundo gráfico
for bar, valor, ranking in zip(bars2, df_sorted_20['sum_2010_2015'], df_sort
    ax2.text(
        bar.get_x() + bar.get_width()/2,
        valor + (0.02 * df_sorted_20['sum_2010_2015'].max()),
        str(int(ranking)),
        ha='center',
        va='bottom',
```

```
        fontsize=10,
        bbox=dict(facecolor='white', alpha=0.7, edgecolor='none')
    )

# --- Gráfico 3: 2020-2023 ---
df_sorted_2020 = df_2020_completo.sort_values('sum_2020_2023', ascending=Fal
bars3 = ax3.bar(
    x_positions,
    df_sorted_2020['sum_2020_2023'],
    width=bar_width,
    color=[color_por_region.get(region, 'gray') for region in df_sorted_2020
)
ax3.set_title('Período 2020-2023', fontsize=18)
ax3.set_xlabel('Países', fontsize=14)
ax3.set_ylabel('Crecimiento del PIB (%)', fontsize=14)
ax3.set_xticks(x_positions)
ax3.set_xticklabels(df_sorted_2020['country'], rotation=80, ha='right', font
ax3.grid(axis='y', linestyle='--', alpha=0.7)

# --- Leyenda común ---
handles = [Patch(color=color_por_region[region], label=region) for region in
legend = fig.legend(
    handles=handles,
    title='Región',
    loc='lower center',
    bbox_to_anchor=(0.5, -0.05),
    ncol=min(len(regiones_totales), 5),
    fontsize=12,
    title_fontsize=14
)

# Ajustar layout para hacer espacio
plt.tight_layout()
plt.subplots_adjust(bottom=0.12, hspace=0.4, wspace=0.3)  # Ajustar espacios

# Guardar figura
plt.savefig(
    'comparacion_pib_disposicion_personalizada.png',
    bbox_extra_artists=(legend,),
    bbox_inches='tight',
    dpi=300
)

plt.show()
```



Comparative GDP, year 2023: American Latin Countries vs. OCDE Countries

In [232…

```
# Indicador a explorar:
indicadores = {
    'crecimiento del PIB': 'NY.GDP.MKTP.KD.ZG'
    }

paises = ['BLZ', 'CRI', 'SLV', 'GTM', 'HND', 'NIC', 'PAN', 'ATG', 'BHS',
          'BRB', 'CUB', 'DMA', 'GRD', 'HTI', 'JAM', 'DOM', 'KNA', 'VCT',
          'LCA', 'TTO', 'ARG', 'BOL', 'BRA', 'CHL', 'COL', 'ECU', 'GUY',
          'PRY', 'PER', 'SUR', 'URY', 'VEN']

pib_america = descargar_datos_wb(indicadores, paises, '2023', '2023')
pib_america.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   country    32 non-null     object
 1   year       32 non-null     object
 2   valor      31 non-null     float64
 3   indicador  32 non-null     object
dtypes: float64(1), object(3)
memory usage: 1.1+ KB
```

```
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
```

In [233…
```
pib_america.to_excel('pib_america.xlsx')
```

In [234…
```
pib_america.head(2)
```

Out[234…

	country	year	valor	indicador
0	Argentina	2023	-1.611002	crecimiento del PIB
1	Antigua and Barbuda	2023	3.862012	crecimiento del PIB

In [235…
```
#Revisando los datos de los paíse de la OCDE
df_pib
```

Out[235…

	country	year	valor	indicador
0	Australia	2023	3.441992	crecimiento del PIB
1	Australia	2022	4.242386	crecimiento del PIB
2	Australia	2021	2.111168	crecimiento del PIB
3	Australia	2020	-0.119591	crecimiento del PIB
4	Australia	2019	2.171545	crecimiento del PIB
...	...	...	...	...
1287	United States	1994	4.029023	crecimiento del PIB
1288	United States	1993	2.751796	crecimiento del PIB
1289	United States	1992	3.522497	crecimiento del PIB
1290	United States	1991	-0.108313	crecimiento del PIB
1291	United States	1990	1.885966	crecimiento del PIB

1292 rows × 4 columns

In [236…
```
# Generar un data frame con solo el año "2023"
anio_seleccionado = '2023'

# Filtrar el DataFrame por el estado seleccionado
pib_OCDE = df_pib[df_pib['year'] == anio_seleccionado]
pib_OCDE.head(2)
```

Out[236…

	country	year	valor	indicador

In [237…
```
#Ahora se desea obtener el promedio del crecimiento del PIB de los países OC
OCDE_prom = pib_OCDE['valor'].mean()
round(OCDE_prom, 2)
```

Out[237… nan

In [238…
```python
# Mapeo de códigos ISO a nombres de países en inglés
iso_to_country = {
    'AUS': 'Australia',
    'AUT': 'Austria',
    'BEL': 'Belgium',
    'CAN': 'Canada',
    'CHL': 'Chile',
    'COL': 'Colombia',
    'CRI': 'Costa Rica',
    'CZE': 'Czech Republic',
    'DNK': 'Denmark',
    'EST': 'Estonia',
    'FIN': 'Finland',
    'FRA': 'France',
    'DEU': 'Germany',
    'GRC': 'Greece',
    'HUN': 'Hungary',
    'ISL': 'Iceland',
    'IRL': 'Ireland',
    'ISR': 'Israel',
    'ITA': 'Italy',
    'JPN': 'Japan',
    'KOR': 'South Korea',
    'LVA': 'Latvia',
    'LTU': 'Lithuania',
    'LUX': 'Luxembourg',
    'MEX': 'Mexico',
    'NLD': 'Netherlands',
    'NZL': 'New Zealand',
    'NOR': 'Norway',
    'POL': 'Poland',
    'PRT': 'Portugal',
    'SVK': 'Slovakia',
    'SVN': 'Slovenia',
    'ESP': 'Spain',
    'SWE': 'Sweden',
    'CHE': 'Switzerland',
    'TUR': 'Turkey',
    'GBR': 'United Kingdom',
    'USA': 'United States'
}

# Lista de códigos de países (la que proporcionaste)
paises = [
    'AUS', 'AUT', 'BEL', 'CAN', 'CHL', 'COL', 'CRI',
    'CZE', 'DNK', 'EST', 'FIN', 'FRA', 'DEU', 'GRC',
    'HUN', 'ISL', 'IRL', 'ISR', 'ITA', 'JPN', 'KOR',
    'LVA', 'LTU', 'LUX', 'MEX', 'NLD', 'NZL',
    'NOR', 'POL', 'PRT', 'SVK', 'SVN', 'ESP',
    'SWE', 'CHE', 'TUR', 'GBR', 'USA'
]

# Crear el DataFrame
```

```
OCDE = pd.DataFrame({
    'country': [iso_to_country[code] for code in paises],  # Nombres en ingl
    'entidad': 'OCDE'  # Valor constante
})

# Mostrar las primeras filas
OCDE.head()
```

Out[238…

	country	entidad
0	Australia	OCDE
1	Austria	OCDE
2	Belgium	OCDE
3	Canada	OCDE
4	Chile	OCDE

In [239…

```
# Hacer el left join
pib_america = pd.merge(
    pib_america,
    OCDE[['country', 'entidad']],  # Seleccionar solo las columnas necesaria
    on='country',
    how='left'
)
```

In [240…

```
pib_america.head(2)
```

Out[240…

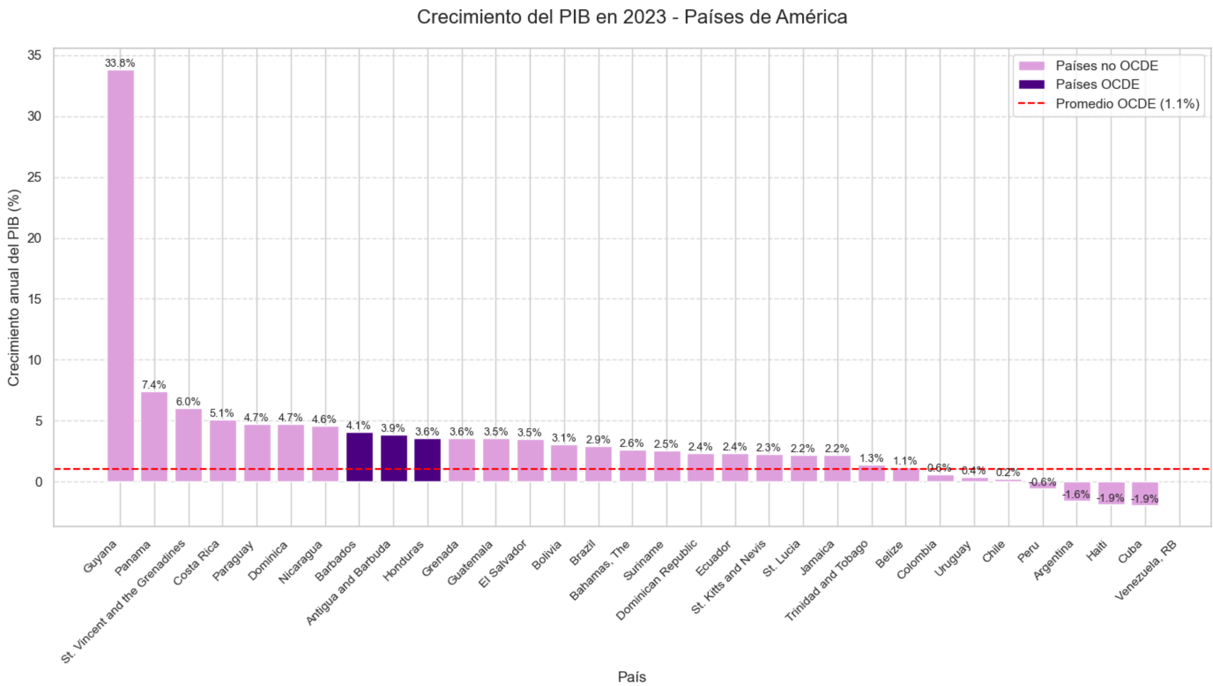
	country	year	valor	indicador	entidad
0	Argentina	2023	-1.611002	crecimiento del PIB	NaN
1	Antigua and Barbuda	2023	3.862012	crecimiento del PIB	NaN

In [241…

```
# 1. Definir colores basados en la columna 'entidad'
colors = ['indigo' if entidad == 'OCDE' else 'plum' for entidad in pib_ameri

# 2. Ordenar los valores para mejor visualización
pib_america = pib_america.sort_values('valor', ascending=False)

# 3. Crear el gráfico de barras vertical
plt.figure(figsize=(14, 8))
bars = plt.bar(
    pib_america['country'],
    pib_america['valor'],
    color=colors  # Usar la lista de colores personalizada
)

# --- Agregar línea del promedio OCDE ---
OCDE_prom = 1.05
plt.axhline(
    y=OCDE_prom,
    color='red',
    linestyle='--',
    linewidth=1.5,
```

```
    label=f'Promedio OCDE ({OCDE_prom:.1f}%)'
)

# 4. Personalizar el gráfico
plt.title('Crecimiento del PIB en 2023 - Países de América', fontsize=16, pa
plt.xlabel('País', fontsize=12)
plt.ylabel('Crecimiento anual del PIB (%)', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# 5. Añadir etiquetas de valor en las barras
for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width()/2.,
        height,
        f'{height:.1f}%',
        ha='center',
        va='bottom',
        fontsize=9
    )

# 6. Leyenda personalizada para los colores
from matplotlib.patches import Patch
legend_elements = [
    Patch(facecolor='plum', label='Países no OCDE'),
    Patch(facecolor='indigo', label='Países OCDE'),
    plt.Line2D([0], [0], color='red', linestyle='--', label=f'Promedio OCDE
]
plt.legend(handles=legend_elements, loc='upper right')

# 7. Ajustar diseño y mostrar
plt.tight_layout()
plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
```



In [ ]:

In [242…

```python
# Indicador a explorar:
indicadores = {
    'gasto_militar': 'MS.MIL.XPND.GD.ZS',
    'gasto_corriente_salud': 'SH.XPD.CHEX.GD.ZS',
    'gasto_publico_educacion': 'SE.XPD.TOTL.GD.ZS',
    'gasto_investigacion_desarrollo':'GB.XPD.RSDV.GD.ZS',
}

paises = ['CHL']

df_datos = descargar_datos_wb(indicadores, paises, '1990', '2024')
df_datos.info()
```

```
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   country    140 non-null    object
 1   year       140 non-null    object
 2   valor      101 non-null    float64
 3   indicador  140 non-null    object
dtypes: float64(1), object(3)
memory usage: 4.5+ KB
```

```
C:\Users\Paula\AppData\Local\Temp\ipykernel_9652\2667942121.py:7: FutureWarn
ing: errors='ignore' is deprecated and will raise in a future version. Use t
o_numeric without passing `errors` and catch exceptions explicitly instead
  df = wb.download(
```

In [243…

```
# Agrupar por 'indicador' y aplicar describe() a 'valor'
descripcion_por_indicador = df_datos.groupby('indicador')['valor'].describe(

descripcion_por_indicador
```

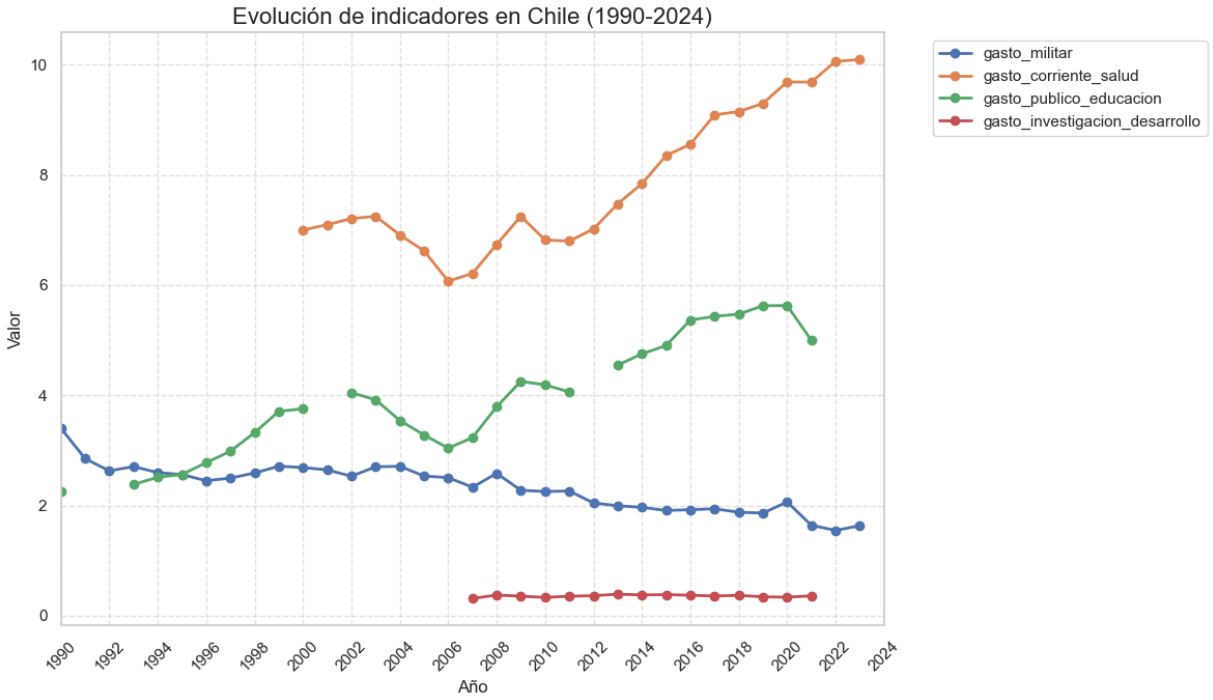
Out[243…

indicador	count	mean	std	min	25%
gasto_corriente_salud	24.0	7.845917	1.279334	6.072356	6.887610
gasto_investigacion_desarrollo	15.0	0.358137	0.021188	0.311430	0.347655
gasto_militar	34.0	2.336439	0.412634	1.548322	1.973986
gasto_publico_educacion	28.0	3.941245	1.044306	2.250160	3.184907

In [244…

```
plt.figure(figsize=(12, 7))
df_datos['year'] = df_datos['year'].astype(int)  # Antes de graficar

# Iterar por cada indicador único
for indicador in df_datos['indicador'].unique():
    subset = df_datos[df_datos['indicador'] == indicador]
    plt.plot(subset['year'], subset['valor'],
             label=indicador,
             marker='o',
             linewidth=2)

# Configuración del eje X (años)
years = df_datos['year'].unique()  # Obtener todos los años únicos
pares = [year for year in years if year % 2 == 0]  # Filtrar solo pares
plt.xticks(pares)  # Establecer ticks solo para años pares

# Asegurar orden cronológico (de menor a mayor)
plt.xlim(min(years), max(years))  # Límites del eje X

plt.title('Evolución de indicadores en Chile (1990-2024)', fontsize=16)
plt.xlabel('Año', fontsize=12)
plt.ylabel('Valor', fontsize=12)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.6)
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



NY.GDP.PCAP.CD GDP per capita (current US$) (2023)

In [246…

```
min_wave= pd.read_csv('sueldo_minimo_historico.csv')
min_wave
```

Out[246…

	ref_area.label	source.label	indicator.label	classif1.label	time	obs_va
0	Bulgaria	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2024	933.(
1	Bulgaria	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: $ PPA 2021	2024	1109.8
2	Bulgaria	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Dólar estadounidense	2024	516.(
3	Bulgaria	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2023	780.(
4	Bulgaria	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: $ PPA 2021	2023	980.3
...	...	...	...	...	...	
610	Polonia	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: $ PPA 2021	1996	247.3
611	Polonia	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Dólar estadounidense	1996	137.2
612	Polonia	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1995	305.(
613	Polonia	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: $ PPA 2021	1995	237.3
614	Polonia	ADM - Legislación Laboral	Salario mínimo nominal mensual bruto	Divisa: Dólar estadounidense	1995	125.7

615 rows × 8 columns

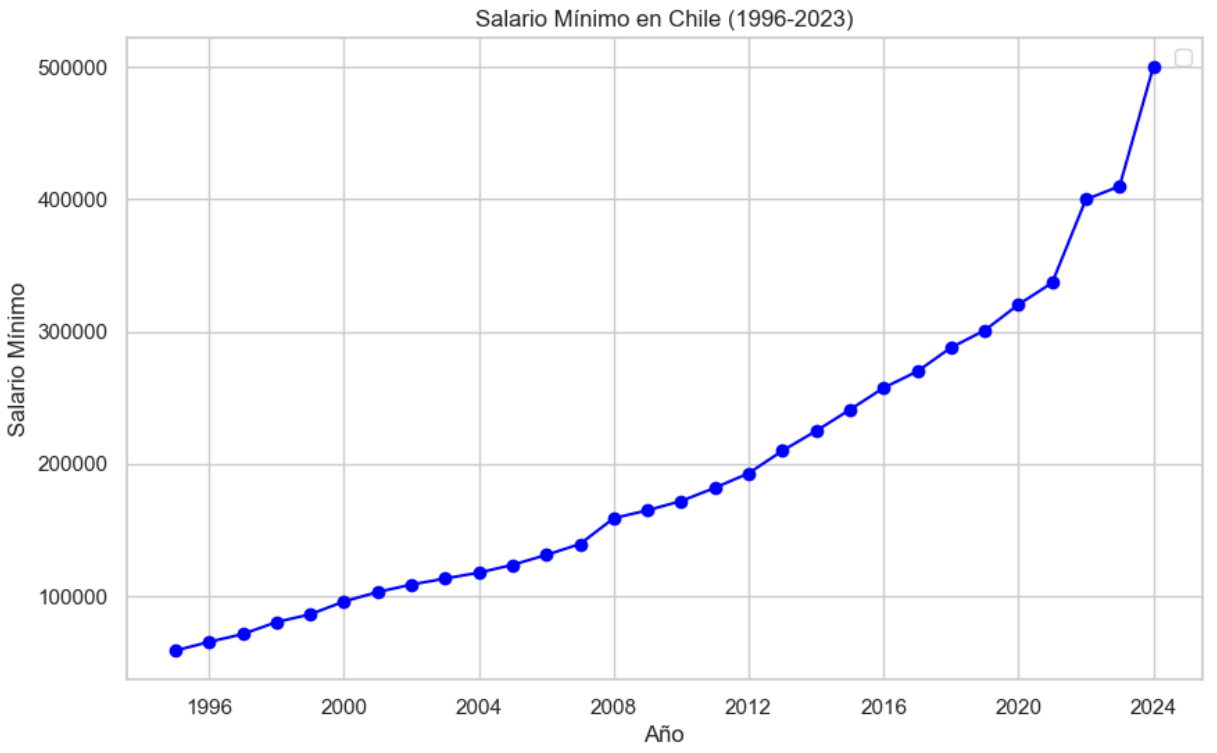
In [247…
```
min_wave.columns
```

Out[247…
```
Index(['ref_area.label', 'source.label', 'indicator.label', 'classif1.label',
       'time', 'obs_value', 'note_indicator.label', 'note_source.label'],
      dtype='object')
```

In [248…
```
# Filtrar: Chile, "Currency: Local currency" y Total
df_chile = min_wave[
    (min_wave['ref_area.label'] == 'Chile') &
    (min_wave['classif1.label'] == 'Divisa: Moneda local')
].copy()

df_chile['time'] = pd.to_datetime(df_chile['time'], format='%Y')
```

```
# Gráfico
plt.figure(figsize=(10, 6))
plt.plot(df_chile['time'], df_chile['obs_value'], marker='o', color='blue')
plt.title('Salario Mínimo en Chile (1996-2023)')
plt.xlabel('Año')
plt.ylabel('Salario Mínimo')
plt.grid(True)
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```



In [249…

```
df_chile
```

Out[249…

	ref_area.label	source.label	indicator.label	classif1.label	time	obs_val
78	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2024-01-01	500000
81	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2023-01-01	410000
84	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2022-01-01	400000
87	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2021-01-01	337000
90	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2020-01-01	320500
93	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2019-01-01	301000
96	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2018-01-01	288000
99	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2017-01-01	270000
102	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2016-01-01	257500
105	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2015-01-01	241000
108	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2014-01-01	225000
111	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2013-01-01	210000
114	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2012-01-01	193000
117	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2011-01-01	182000
120	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2010-01-01	172000
123	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2009-01-01	165000

	ref_area.label	source.label	indicator.label	classif1.label	time	obs_val
126	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2008-01-01	15900
129	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2007-01-01	13950
132	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2006-01-01	13125
135	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2005-01-01	12375
138	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2004-01-01	11782
141	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2003-01-01	11342
144	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2002-01-01	10882
147	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2001-01-01	10320
150	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	2000-01-01	9604
153	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1999-01-01	8633
156	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1998-01-01	8050
159	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1997-01-01	7140
162	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1996-01-01	6550

	ref_area.label	source.label	indicator.label	classif1.label	time	obs_val
165	Chile	OIT - SIALC Estimaciones	Salario mínimo nominal mensual bruto	Divisa: Moneda local	1995-01-01	5890

In [ ]: