Regresiones y clusters entre variables macroeconómicas del año 2021

Introducción a Ciencia de Datos

Antonio Huerta Montellano

$\quad$ ***Resumen**: En este trabajo vamos a analizar los índices tales como el índice de calidad de vida y Gini, así como el producto interno bruto (PIB), la tasa de inflación, la expectativa de vida y la población de distintos países con el objetivo de hallar características o cualidades que nos ayuden a entender mejor de manera macroeconómica qué caulidades podrían estar relacionadas. Así mismo, se hallaron la regresiones lineales del PIB con el logaritmo de la expectavida de vida, así como PIB y la tasa de inflación.*

Índice:

$\quad$ *Se puede dar click en la parte deseada*

Introducción

$\quad$ Se sabe que existen muchos parámetros e índices para evaluar distintos aspectos de un país tales como los económicos, políticos, calidad de vida, entre otros. Sin embargo, casi siempre se necesita más de un índice para poder determinar incluso solo una característica interés, es por esta razones que existen muchos índices y parámetros asociados a distintos aspectos de tan solo un ámbito, donde no es raro que cierto índices estén relacionados de manera general como la expectativa de vida y el producto interno bruto, razón por la que se llega a utilizar como medida del bienestar. Sin embargo, si bien ya se han obtenido resultados y conclusiones a partir del estudio de los datos o medelos tanto teóricos como empíricos, estos han sido bajo cierrtas condiciones y supuestos; es por esta razón que cabe preguntarse si dichos resultados continuan siendo válidos después de eventos tan drásticos a nivel mundial tales como la pandemia debida al virus de la *COVID-19* puesto que debido al impacto

económico que tuvo tales como un gran desempleo a nivel mundial, la banca rota de muchas empresas y el incremento excesivo de reportes de depresión en personas a lo largo del mundo se espera que valores tales como el índice de *calidad de vida*, *Gini*, la *expectativa de vida* e incluso el *PIB*. Es por este motivo que por medio de cluster estudiaremos las relaciones entre dichas variables además de la tasa de inflación y población; por otro lado, también se obtendrá la regresión entre la *expectativa de vida* y el logaritmo del *PIB*, y entre las demás variables que observemos una correlación significativa puesto que hay evidencia de correlación entre el *PIB* y el ingreso medio per cápita, esto servirá de base para usarse en otras investigaciones y contrastar con los resultados teóricos de otros años. Este trabajo se divide en esta introducción para después ofrecer un marco teórico de las variables y regresiones que se estudiarán; posteriormente se hará el respectivo análisis de los datos donde se dividirá en subsecciones que constan de la limpieza de datos, visualizaciones de los datos, construcción de las bases de datos, aplicación de métodos de machine learning; después, en otra sección, estará la interpretación de los datos; por último, las conclusiones.

Marco teórico

$\quad$ Si bien, hay variables que intuitivamente se espera que estén correlacionadas, hay evidencia empírica de que algunas de nuestras variables de interés están correlacionadas como es el caso del *PIB* y la expectativa de vida [1] [2] [3] cuya relación analítica está dada por la siguiente ecuación

$$ Expectativa \,\, de \,\, vida = \beta_0 + \beta_1 \log\left(PIB \right) + u_t, $$
datos

donde los errores u_t están identicamente distribuidos con una distribución normal estándar. A pesar de que esta relación se aproxime mejor para países con un *PIB* bajo, por lo general se toma para todos los países puesto que la correlación tiende a ser bastante significativa. Por otro lado, a pesar de que el *PIB* sea utilizado como una medida del bienestar, no necesariamente implica que un país con un *PIB* grande tenga un alto índice de calidad de vida. Sin embargo, cabe preutnarse si estas variables están correlacionadas significativamente para la mayoría de los países, en particular, veremos si existe alguna relación significativa entre esta y otras variables. Análogamente, las variables que estén suficientemente correlacionadas serán regresadas entre sí mientras no haya multicolinealidad, aunque hay variables que se esperan estar correlacioandas como el *PIB* y la *tasa de inflación*[2]. Sin embargo, también hay posibles relaciones intuitivas tales como una relación a nivel general entre la *expectativa de vida* y la *calidad de vida* a pesar de que haya países que sirvan de contraejemplo tales como China. Cabe mencionar que no se descarta que hayan variables correlacionadas que no tengan una relación causal o explícita más allá de los datos, pero que sirva para futuras comparativas en otros trabajos.

$\quad$ Por otra parte, se aplicarán técnicas de *clustering* [4] para detectar grupo representativos por medio del algoritmo llamado *K-means++* [5]. Este algoritmo está basado en el agoritmo usual del *K-means* con la diferencia de que este algoritmo resuelve la problemática *NP-duro k-means* donde el agoritmo usual arrojaba centros de clusters que satisfacían la condición de mínima distancia del algoritmo, pero que visualmente se apreciaba que los culsters están incorretamente definidos. Por otro

lado, en este trabajo no se usará un conjunto de etiquetas discretas, sino un conjunto contínuo el cual será mapeado a un gradiente de color que nos permita visualizar tendencias o grupos representativos a partir de aplicar un umbral para determinar el número de cluster necesarios para cada caso.

En este trabajo se omitirán los *outliers* para el correcto funionamiento de los algorítmos de clasificación regresión. El método que se utilizará será mediante el uso del cálculo de los cuartiles para filtrar fuera los datos que no estén dentro del segundo y tercer cuartil más 1.5 veces el rango intercuartil de los datos correspondientes [6].

Debido al mes en el que se está haciendo este proyecto, las bases de datos del año en curso, 2022, no han sido propiamente actualizadas, no están completas o las fuentes no son confiables; es por este motivo que para este análisis nos restringiremos a los datos del año 2021.

Análisis de datos</h3>

$\quad$ A continuación, importaremos cada una de las bases de datos para observar cosas generales como el tipo de datos que guardan, el nombre de las columnas o datos faltantes y se harán la limpieza de datos necesaria en caso de requerirse y dependiendo del caso.

In [1]:
```
import pandas as pd
import numpy as np
import math
import plotly.express as px
import plotly.graph_objects as go
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import plotly
from sklearn.impute import SimpleImputer
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn import metrics

print('Modulos importados')
```

```
Modulos importados
```

A continuación, modificaremos algunos parámetros que están por defecto, esto con el propósito de presentar los resultados de una manera más estática.

In [2]:
```
# Para modificar el tamaño de todos los gráficos.
from matplotlib import rcParams
rcParams['figure.figsize'] = 15,9
```

In [3]:
```
# Para que no nos molesten los mensajes de advertencia.
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
# Reduzcamos el número de línea a leer cuando se imprime el dataframe
pd.options.display.max_rows = 10
```

Limpieza de datos

$\quad$ La base de datos que necesitamos para el análisis por clustering se construirá con todas las bases de datos que tenemos, donde la columna *ID* será el nombre del país puesto que no hay dos países con el mismo nombre. Para facilitar el manejo de los dataframes, solo tomaremos las columnas de las variables que nos interesan y luego se determinará el criterio para rellenar los *NaN*. Por otro lado, se trabajará con copias de los dataframes puesto que se hará otro aparte para el análisis de la regresiones lineales propuestas. Además, con el propósito de no hacer engorroso el análisis, se preferirá tomar variables que engloben más datos o que sean únicas en el sentido de no estar correlacionadas con otras variables.

Primero, guaremos en distintas variables la ruta de los dataframes por buena práctica:

In [5]:
```
# Especificamos la ruta de los datos
pib_path = 'Data/GDP_2021.csv'
gini_path = 'Data/Gini_2022.csv'
inflacion_path = 'Data/Inflation_Rate_2021.csv'
e_vida_path = 'Data/Life_Expectancy_2021.csv'
c_vida_path = 'Data/Quality_life_2021.csv'
```

In [6]:
```
# Importamos las bases de datos
pib = pd.read_csv(pib_path)
gini = pd.read_csv(gini_path)
inflacion = pd.read_csv(inflacion_path)
e_vida = pd.read_csv(e_vida_path)
c_vida = pd.read_csv(c_vida_path)
```

Para hacer el proceso de limpieza de datos eficiente, se hará toda la limpieza de datos dataframe por dataframe. Primero, solo conservaremos las columnas de interés en dataframes que sean copias del original y luego aplicar la limpiezza correspondiente, esto para no modificar el dataframe original y por aplicar el conjunto de buenas prácticas. El subíndice *c* denotará que es la copia directa del dataframe.

In [7]:
```
# Creamos una copia de cada dataframe
pib_c = pib.copy()
gini_c = gini.copy()
inflacion_c = inflacion.copy()
e_vida_c = e_vida.copy()
c_vida_c = c_vida.copy()
```

Ahora, veremos cada uno de los dataframes para determinar con qué columnas nos quedamos para el análisis tanto para clustering como por regresión lineal.

Producto interno bruto per cápita[7]

Veamos el contenido del dataframe correspondiente.

In [8]:
```
pib_c.head()
```

Out[8]:

	Rank	Name	GDP (IMF '19)	GDP (UN '16)	GDP Per Capita	Population 2021
0	1	United States	22.20 Tn	18.62 Tn	$66,678	332,915,073
1	2	China	15.47 Tn	11.22 Tn	$10,710	1,444,216,107
2	3	Japan	5.50 Tn	4.94 Tn	$43,597	126,050,804
3	4	Germany	4.16 Tn	3.48 Tn	$49,548	83,900,473
4	5	India	3.26 Tn	2.26 Tn	$2,338	1,393,409,038

Veamos cuántos páises hay en la base datos para tener el número de referencia para más adelante que se haga el *merge* de las bases de datos.

In [9]:
```
# Dimensiones del dataframe
pib_c.shape
```

Out[9]: (211, 6)

Tal como se mencionó antes, nos interesa el *PIB per cápita*, pero también conservaremos los datos de la población por país.

In [10]:
```
# Asignamos solo las columnas que necesitamos
pib_c = pib_c[['Name','GDP Per Capita', 'Population 2021']]
pib_c.head()
```

Out[10]:

	Name	GDP Per Capita	Population 2021
0	United States	$66,678	332,915,073
1	China	$10,710	1,444,216,107
2	Japan	$43,597	126,050,804
3	Germany	$49,548	83,900,473

Name	GDP Per Capita	Population 2021

Para evitar problemas con los métodos, algoritmos y funciones se cambiarán los nombres de los dataframes por aquellos que no tengan espacios en blanco, ni caracteres especiales buscando la intuitividad.

In [11]:
```
# Creamos una lista con los nuevos nombres que asignaremos a las columnas del
pib_c_names = ['Pais','PIB_pc', 'Poblacion']
# Asignamos la lista anterior al parámetro que contiene el nombre de las colu
pib_c.columns = pib_c_names
pib_c.head()
```

Out[11]:

	Pais	PIB_pc	Poblacion
0	United States	$66,678	332,915,073
1	China	$10,710	1,444,216,107
2	Japan	$43,597	126,050,804
3	Germany	$49,548	83,900,473
4	India	$2,338	1,393,409,038

En la columna del *PIB per cápita* contiene el caracter especial *$*, por lo que tenemos que removerlo ya que necesitamos que sea la columna del tipo flotante. Por otro lado, también tenemos que eliminar las comas en las columnas para después convertir los tipos de datos de las columnas al adecuado.

In [12]:
```
# Quitamos el caracter $ de la serie correspondiente al PIB per cápita
pib_aux_1 = pib_c['PIB_pc'].str.replace("$","")
# Quitamos el caracter ',' de la serie correspondiente al PIB per cápita
pib_aux_2 = pib_aux_1.str.replace(",","")
# Quitamos el caracter ',' de la serie correspondiente al PIB per cápita
pib_aux_3 = pib_c['Poblacion'].str.replace(",","")
# Asignamos a la serie del PIB_pc el valor de la lista pib_aux_2 para replaza
pib_c['PIB_pc'] = pib_aux_2
# Asignamos a la serie del Población el valor de la lista pib_aux_3 para repl
pib_c['Poblacion'] = pib_aux_3
pib_c.head()
```

Out[12]:

	Pais	PIB_pc	Poblacion
0	United States	66678	332915073
1	China	10710	1444216107
2	Japan	43597	126050804
3	Germany	49548	83900473
4	India	2338	1393409038

Veamos el tipo de datos que contienen.

In [13]:
```
# Nos muestra el tipo de datos que contiene cada columna donde 'objetc' tambi
print(pib_c.dtypes)
```

```
Pais         object
PIB_pc       object
Poblacion    object
dtype: object
```

Para asegurarnos que no haya problemas con los datos, convirtamos el tipo de todas las columnas a tipo entero o flotante.

In [14]:
```
# Transformamos los strings a variables 'int' o 'float' en caso de que se pue
pib_c['PIB_pc'] = pd.to_numeric(pib_c['PIB_pc'])
pib_c['Poblacion'] = pd.to_numeric(pib_c['Poblacion'])
```

Veamos los tipos de datos que almacena cada columna con el propósito de encontrar irregularidades.

In [15]:
```
print(pib_c.dtypes)
```

```
Pais         object
PIB_pc        int64
Poblacion     int64
dtype: object
```

Por último, verefiquemos que no haya datos tipo *NaN* o nulos.

In [16]:
```
# isna() determinar si algún registro tiene algún valor NaN y en caso de ser
# con sum(), se hace una suma de todos los valores 'True' obtenidos con el mé
# Análogamente con isnull(), donde este método nos muestra sí hay valores nul
print("PIB_pc nan = " + str(pib_c['PIB_pc'].isna().sum()) + "\n" + "PIB_nc nu
print("Población nan = " + str(pib_c['Poblacion'].isna().sum()) +  "\n" + "Po
print("País nan = " + str(pib_c['Pais'].isna().sum()) +  "\n" + "Pais nulls =
```

```
PIB_pc nan = 0
PIB_nc nulls = 0
Población nan = 0
Población nulls = 0
País nan = 0
Pais nulls = 0
```

Como no tenemos datos faltante ya tenemos este dataframe limpio.

Índice Gini[8]<

Se hará la limpieza de datos, análoga al dataframe anterior.

In [17]:
```
gini_c.tail(5)
```

Out[17]:

	country	giniWB	yearWB	giniCIA	yearCIA	pop2022

	country	giniWB	yearWB	giniCIA	yearCIA	pop2022
171	Hong Kong	NaN	NaN	53.9	2016.0	7604.299
172	Cambodia	NaN	NaN	37.9	2008.0	17168.639
173	Taiwan	NaN	NaN	33.6	2014.0	23888.595
174	Saudi Arabia	NaN	NaN	45.9	2013.0	35844.909

In [18]:
```
gini_c.shape
```

Out[18]: (176, 6)

Debido a que solo nos interesa la población del 2021 para este análisis, borraremos la columna que contiene la población de cada país del año 2022. Por otro lado, se tomará el promedio de los índices gini *WB* y *CÍA* para acompletar los datos entre sí puesto que hay datos *NaN* que en la otra columna no están y viceversa, también para tener una variable que tome en cuenta ambas variables.

In [19]:
```
gini_c['Gini'] = gini_c[['giniWB','giniCIA']].mean(axis = 1, skipna = True)
gini_c.tail()
```

Out[19]:

	country	giniWB	yearWB	giniCIA	yearCIA	pop2022	Gini
171	Hong Kong	NaN	NaN	53.9	2016.0	7604.299	53.9
172	Cambodia	NaN	NaN	37.9	2008.0	17168.639	37.9
173	Taiwan	NaN	NaN	33.6	2014.0	23888.595	33.6
174	Saudi Arabia	NaN	NaN	45.9	2013.0	35844.909	45.9
175	Afghanistan	NaN	NaN	29.4	2008.0	40754.388	29.4

Ahora, solo nos quedaremos con las columnas de interes y cambiar los nombres.

In [20]:
```
gini_c = gini_c[['country','Gini']]
gini_names = ['Pais','Gini']
gini_c.columns = gini_names
gini_c.tail()
```

Out[20]:

	Pais	Gini
171	Hong Kong	53.9
172	Cambodia	37.9
173	Taiwan	33.6
174	Saudi Arabia	45.9
175	Afghanistan	29.4

Ahora, nos aseguraremos que los datos sean del tipo correcto y que no haya datos faltantes tal y como se hizo en el caso anterior.

In [21]:
```
gini_c['Gini'] = pd.to_numeric(gini_c['Gini'])
print(gini_c.dtypes)
print("\nGini nan = " + str(gini_c['Gini'].isna().sum()) + "\n" + "Gini nulls
print("País nan = " + str(gini_c['Pais'].isna().sum()) +  "\n" + "Pais nulls
```

```
Pais     object
Gini    float64
dtype: object

Gini nan = 0
Gini nulls = 0
País nan = 0
Pais nulls = 0
```

Por lo tanto, ahora tenemos este dataframe limpio.

Tasa de inflación[9]

Repitamos el proceso para la variable de la tasa de inflación.

In [22]:
```
inflacion_c.head()
```

Out[22]:

	country	lastPerc	previousPerc	dateReference
0	Venezuela	9586.0	14291.00	12/19
1	Zimbabwe	676.0	540.00	03/20
2	Sudan	71.4	64.28	02/20
3	Argentina	46.9	50.30	03/20
4	South Sudan	36.4	69.00	01/20

In [23]:
```
inflacion_c.shape
```

Out[23]: (181, 4)

En este caso, nos quedaremos con el último valor de la tasa de inflación, es decir, la columna llamada *lastPerc*.

In [24]:
```
inflacion_c = inflacion[['country','lastPerc']]
inflacion_names = ['Pais', 'Inflacion']
inflacion_c.columns = inflacion_names
inflacion_c.head()
```

Out[24]:

	Pais	Inflacion
0	Venezuela	9586.0
1	Zimbabwe	676.0
2	Sudan	71.4

	Pais	Inflacion
3	Argentina	46.9
4	South Sudan	36.4

Verifiquemos el tipo de dato que contendrán las columnas.

In [25]:
```
inflacion_c['Inflacion'] = pd.to_numeric(inflacion_c['Inflacion'])
print(inflacion_c.dtypes)
print("\nInflación nan = " + str(inflacion_c['Inflacion'].isna().sum()) + "\n
print("País nan = " + str(inflacion_c['Pais'].isna().sum()) +  "\n" + "Pais n
```

```
Pais          object
Inflacion    float64
dtype: object

Inflación nan = 0
Inflación nulls = 0
País nan = 0
Pais nulls = 0
```

Como todos los datos están correctos, procedamos a las visualizaciones

Expectativa de vida[10]

La variable que toma en cuenta las demás es la expectativa de vida de ambos sexos.

In [26]:
```
e_vida_c.head()
```

Out[26]:

	Rank	Country	Life Expectancy \n (both sexes)	Females \n Life Expectancy	Males \n Life Expectancy
0	1	Hong Kong	85.29	88.17	82.38
1	2	Japan	85.03	88.09	81.91
2	3	Macao	84.68	87.62	81.73
3	4	Switzerland	84.25	86.02	82.42
4	5	Singapore	84.07	86.15	82.06

In [27]:
```
e_vida_c.shape
```

Out[27]: (202, 5)

In [28]:
```
e_vida_c = e_vida_c[['Country', 'Life Expectancy \n (both sexes)']]
e_life_names = ['Pais', 'Expectativa_vida']
e_vida_c.columns = e_life_names
e_vida_c.head()
```

Out[28]:

	Pais	Expectativa_vida

	Pais	Expectativa_vida
0	Hong Kong	85.29
1	Japan	85.03
2	Macao	84.68
3	Switzerland	84.25

In [29]:
```
e_vida_c['Expectativa_vida'] = pd.to_numeric(e_vida_c['Expectativa_vida'])
print(e_vida_c.dtypes)
print("\nExpectativa de vida nan = " + str(e_vida_c['Expectativa_vida'].isna(
print("País nan = " + str(e_vida_c['Pais'].isna().sum()) +  "\n" + "Pais null
```

```
Pais                 object
Expectativa_vida    float64
dtype: object

Expectativa de vida nan = 0
Expectativa de vida nulls = 0
País nan = 0
Pais nulls = 0
```

Como todo está correcto, procedamos a graficar.

Índice de calidad de vida[11]

Análogo al caso del índice *MPI*, el índice de calidad de vida toma en cuenta distintas variables con distintos pesos; es por esta razón que solo nos quedaremos con la variable que denote el índice de calidad de vida.

In [30]:
```
c_vida_c.head()
```

Out[30]:

	Rank	Country	Quality of Life Index	Purchasing Power Index	Safety Index	Health Care Index	Cost of Living Index	Property Price to Income Ratio	Traffic Commute Time Index	Pollution Index	C
0	1	Switzerland	190.82	110.96	78.65	74.47	131.75	8.42	28.73	20.09	
1	2	Denmark	190.01	94.73	73.28	79.96	91.67	6.66	28.69	20.40	
2	3	Netherlands	183.31	83.89	72.78	75.76	78.64	7.35	27.81	25.28	
3	4	Finland	182.79	89.05	72.99	76.40	77.46	8.64	28.96	11.86	
4	5	Austria	182.37	78.23	74.77	78.40	75.49	10.40	25.68	19.20	

In [31]:
```
c_vida_c.shape
```

Out[31]: (83, 11)

In [32]:
```
c_vida_c = c_vida_c[['Country', 'Quality of Life Index']]
c_life_names = ['Pais', 'Calidad_de_vida']
c_vida_c.columns = c_life_names
c_vida_c.head()
```

Out[32]:

	Pais	Calidad_de_vida
0	Switzerland	190.82
1	Denmark	190.01
2	Netherlands	183.31
3	Finland	182.79
4	Austria	182.37

In [33]:
```
c_vida_c['Calidad_de_vida'] = pd.to_numeric(c_vida_c['Calidad_de_vida'])
print(c_vida_c.dtypes)
print("\nCalidad de vida nan = " + str(c_vida_c['Calidad_de_vida'].isna().sum
print("País nan = " + str(c_vida_c['Pais'].isna().sum()) +  "\n" + "Pais null
```

```
Pais                object
Calidad_de_vida    float64
dtype: object

Calidad de vida nan = 0
Calidad de vida nulls = 0
País nan = 0
Pais nulls = 0
```

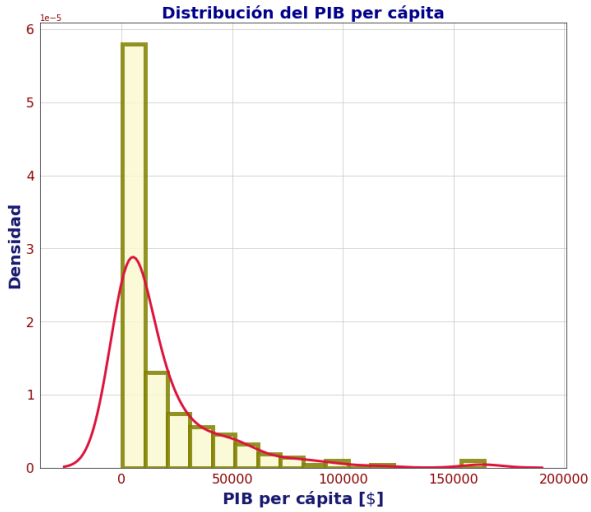
Como todo está en orden, procedamos a hacer las visualizaciones. Primero, veamos cómo están distribuidos los datos para notar anomalías generales, aunque después se analizarán las anomalías tales como los *outliers* mediante diagramas de caja o boxplots.

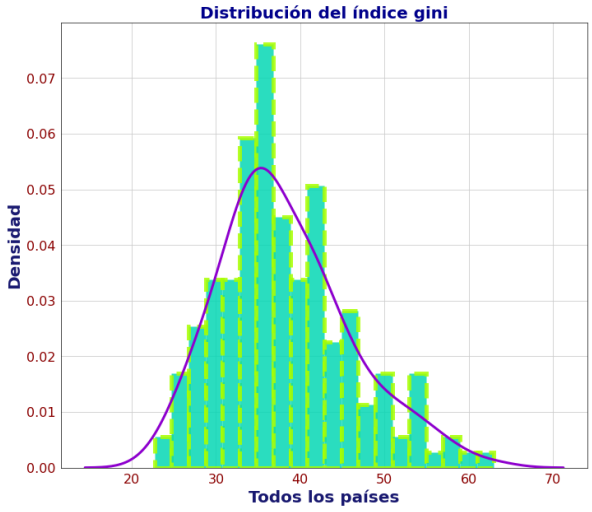
In [96]:

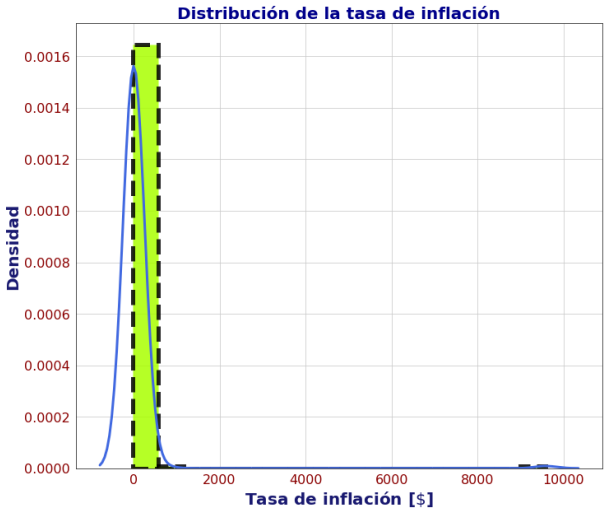
```
sns.set_style('white')
f,axes = plt.subplots(3,2, figsize=(25,35))
f.suptitle("Distribución de las varables", fontsize = 45, color = 'darkcyan',

g_1 = sns.distplot(pib_c['PIB_pc'], bins=16,
             hist_kws={'color':'lightgoldenrodyellow', 'edgecolor':'olive',
                       'linewidth':5, 'linestyle':'solid',
                       'alpha':0.85},
             kde_kws={"color": "#DC143C", "linewidth": 3},
             ax=axes[0,0])
g_1.set_title("Distribución del PIB per cápita", fontsize = 20, fontweight =
g_1.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_1.set_xlabel('PIB per cápita [$\$]', fontsize = 20, fontweight = 'bold', c
g_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_1.grid()

g_2 = sns.distplot(gini_c['Gini'], bins=20,
             hist_kws={'color':'#04D8B4', 'edgecolor':'#aaff00',
                       'linewidth':5, 'linestyle':'--',
                       'alpha':0.85},
             kde_kws={"color": "#8e00ce", "linewidth": 3},
             ax=axes[0,1])
g_2.set_title("Distribución del índice gini", fontsize = 20, fontweight = 'bo
g_2.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_2.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_2.tick_params(labelsize = 16, labelcolor = 'darkred')
g_2.grid()

g_3 = sns.distplot(inflacion_c['Inflacion'], bins=16,
             hist_kws={'color':'#aaff00', 'edgecolor':'black',
                       'linewidth':5, 'linestyle':'dashed',
                       'alpha':0.85},
             kde_kws={"color": "royalblue", "linewidth": 3},
             ax = axes[1,0])
g_3.set_title("Distribución de la tasa de inflación", fontsize = 20, fontweig
g_3.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_3.set_xlabel('Tasa de inflación [$\$]', fontsize = 20, fontweight = 'bold'
g_3.tick_params(labelsize = 16, labelcolor = 'darkred')
g_3.grid()

g_4 = sns.distplot(e_vida_c['Expectativa_vida'], bins = 13,
             hist_kws={'color':'lightgreen', 'edgecolor':'#8e00ce',
                       'linewidth':3, 'linestyle':'dashdot',
                       'alpha':0.85},
             kde_kws={"color": "navy", "linewidth": 5},
             ax = axes[1,1])
g_4.set_title("Distribución de la expectativa de vida", fontsize = 20, fontwe
g_4.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_4.set_xlabel('Expectativa de vida [años]', fontsize = 20, fontweight = 'bol
g_4.tick_params(labelsize = 16, labelcolor = 'darkred')
g_4.grid()

g_5 = sns.distplot(c_vida_c['Calidad_de_vida'], bins = 13,
             hist_kws={'color':'paleturquoise', 'edgecolor':'#DC143C',
                       'linewidth':3, 'linestyle':(0, (3, 5, 1, 5)),
                       'alpha':0.8},
             kde_kws={"color": "#8e00ce", "linewidth": 4},
             ax = axes[2,0])
```

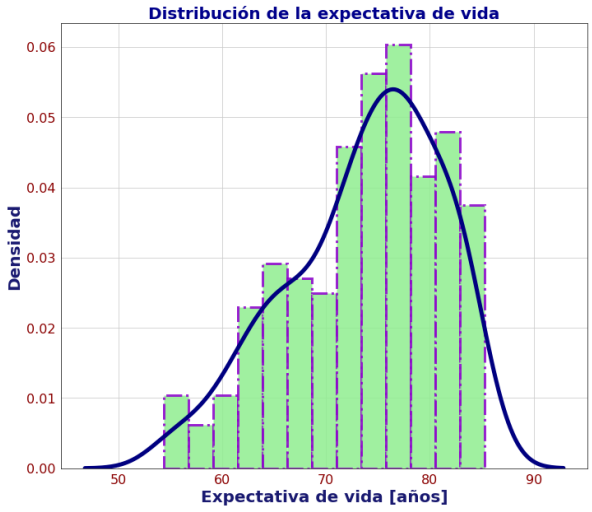
```
g_5.set_title("Distribución índice de calidad de vida", fontsize = 20, fontwe
g_5.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_5.set_xlabel('Índice de calidad de vida', fontsize = 20, fontweight = 'bold
g_5.tick_params(labelsize = 16, labelcolor = 'darkred')
g_5.grid()

g_6 = sns.distplot(pib_c['Poblacion'], bins=20,
             hist_kws={'color':'lightpink', 'edgecolor':'magenta',
                       'linewidth':5, 'linestyle':(0, (3, 5, 1, 5)),
                       'alpha':0.85},
             kde_kws={"color": "darkgoldenrod", "linewidth": 3},
             ax = axes[2,1])
g_6.set_title("Distribución de la población", fontsize = 20, fontweight = 'bo
g_6.set_ylabel('Densidad', fontsize = 20, fontweight = 'bold', color = 'midni
g_6.set_xlabel('Población', fontsize = 20, fontweight = 'bold', color = 'midn
g_6.tick_params(labelsize = 16, labelcolor = 'darkred')
g_6.grid()

# Controlemos el espacio entre los gráficos
plt.subplots_adjust(wspace=0.18, hspace=0.26)
plt.show()
```

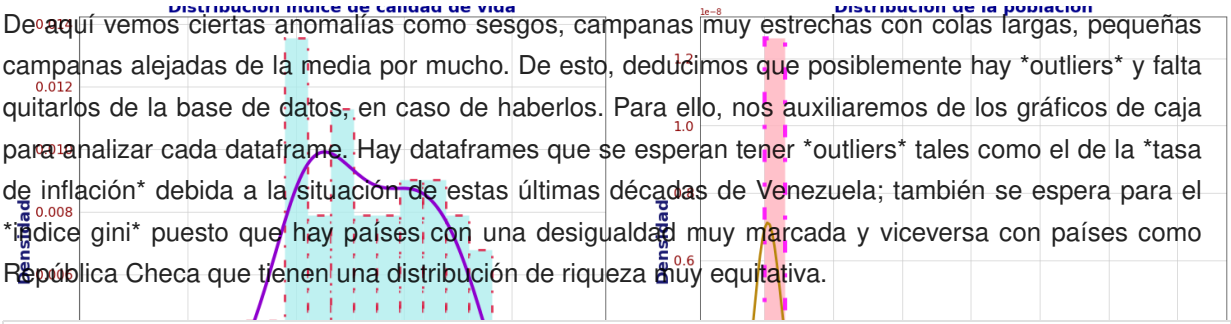
Distribución de las varables











De aquí vemos ciertas anomalías como sesgos, campanas muy estrechas con colas largas, pequeñas campanas alejadas de la media por mucho. De esto, deducimos que posiblemente hay *outliers* y falta quitarlos de la base de datos, en caso de haberlos. Para ello, nos auxiliaremos de los gráficos de caja para analizar cada dataframe. Hay dataframes que se esperan tener *outliers* tales como el de la *tasa de inflación* debida a la situación de estas últimas décadas de Venezuela; también se espera para el *índice gini* puesto que hay países con una desigualdad muy marcada y viceversa con países como República Checa que tienen una distribución de riqueza muy equitativa.

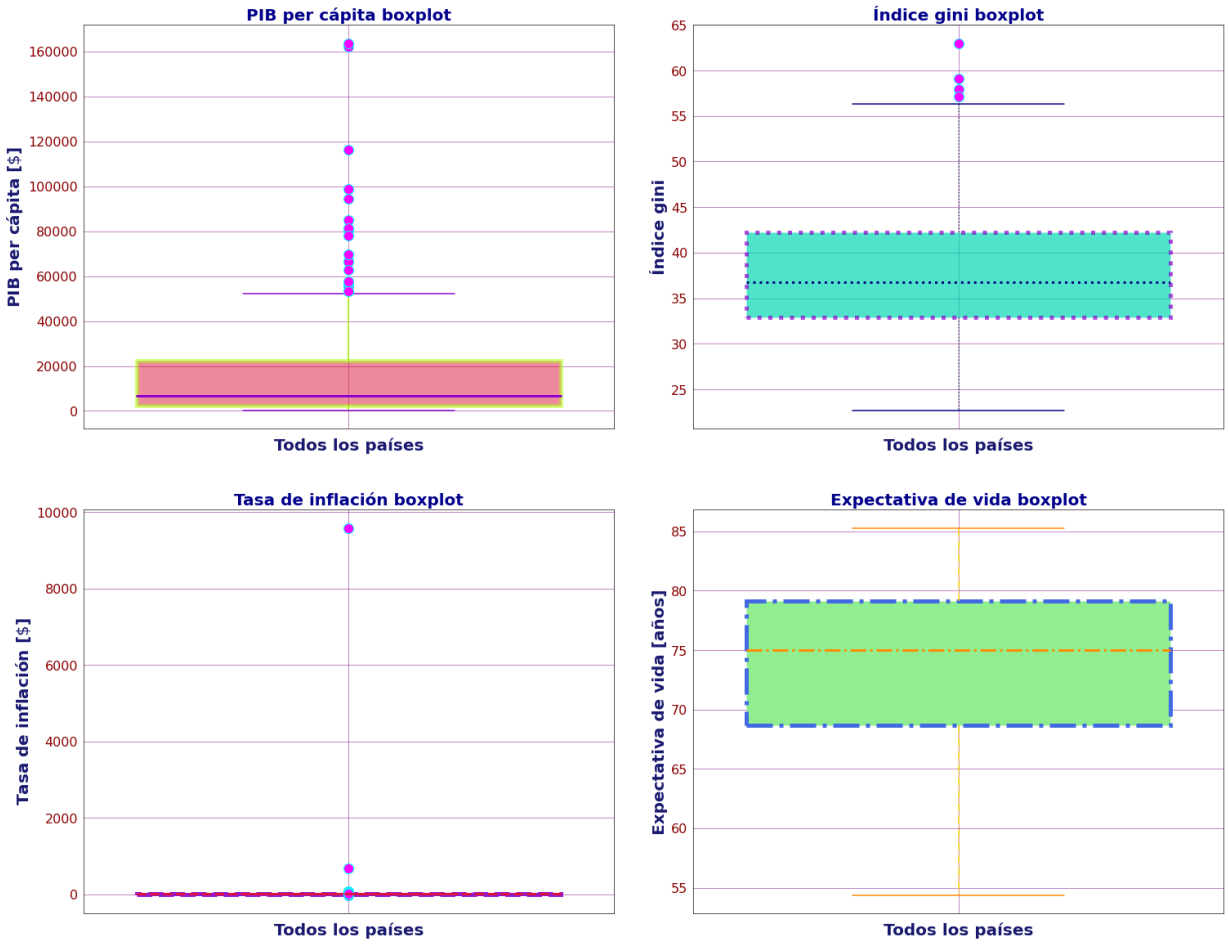
In [77]:
```
# Estilos para los boxplots usando los kwargs para guardar los estilos de los
kws_1 = {'boxprops':{'facecolor':'#DC143C','edgecolor':'#aaff00', 'linestyle'
    'medianprops':{'color':'#8e00ce', 'linewidth':3},
    'whiskerprops':{'color':'#aaff00'},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'#8e00ce'}}

kws_2 = {'boxprops':{'facecolor':'#04D8B4','edgecolor':'#8e00ce', 'linestyle'
    'medianprops':{'color':'navy', 'linestyle':'dotted', 'linewidth':3},
    'whiskerprops':{'color':'darkslategrey', 'linestyle':'dotted'},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'navy'}}

kws_3 = {'boxprops':{'facecolor':'#aaff00','edgecolor':'#8e00ce', 'linestyle'
    'medianprops':{'color':'#DC143C', 'linestyle':'dashed', 'linewidth':3},
    'whiskerprops':{'color':'#8e00ce', 'linestyle':'dashed'},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'#DC143C'}}

kws_4 = {'boxprops':{'facecolor':'lightgreen','edgecolor':'royalblue', 'lines
    'medianprops':{'color':'darkorange', 'linestyle':'dashdot', 'linewidth':3
    'whiskerprops':{'color':'gold', 'linestyle':'dashdot'},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'darkorange'}}

kws_5 = {'boxprops':{'facecolor':'paleturquoise','edgecolor':'goldenrod', 'li
    'medianprops':{'color':'darkcyan', 'linestyle':(0, (3, 5, 1, 5)), 'linewi
    'whiskerprops':{'color':'dodgerblue', 'linestyle':(0, (3, 5, 1, 5))},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'darkcyan'}}

kws_6 = {'boxprops':{'facecolor':'lightpink','edgecolor':'darkgoldenrod', 'li
    'medianprops':{'color':'slateblue', 'linestyle':(0, (3, 1, 1, 1)), 'linew
    'whiskerprops':{'color':'lime', 'linestyle':(0, (3, 1, 1, 1))},
    'flierprops' :{'marker':'o', 'markerfacecolor':'magenta', 'markersize':12
    'capprops':{'color':'slateblue'}}
```

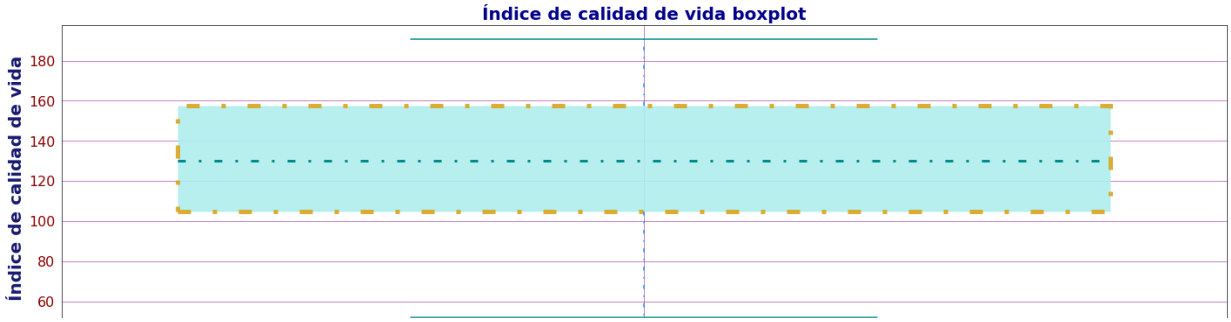
In [175…

```
sns.set_style('white')
f,axes = plt.subplots(2,2, figsize=(25,20))
f.suptitle("Boxplot de las varables", fontsize = 45, color = 'darkcyan', font
g_1 = sns.boxplot(y = 'PIB_pc', data = pib_c,
            palette = 'Set2',
            ax=axes[0,0],
            **kws_1)
g_1.set_title("PIB per cápita boxplot", fontsize = 20, fontweight = 'bold', c
g_1.set_ylabel('PIB per cápita [$\$]', fontsize = 20, fontweight = 'bold', c
g_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_2 = sns.boxplot(y = 'Gini', data = gini_c,
            palette = 'Set2',
            ax=axes[0,1],
            **kws_2)
g_2.set_title("Índice gini boxplot", fontsize = 20, fontweight = 'bold', colo
g_2.set_ylabel('Índice gini', fontsize = 20, fontweight = 'bold', color = 'mi
g_2.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_2.tick_params(labelsize = 16, labelcolor = 'darkred')
g_2.grid(color='purple', linestyle='-', alpha = 0.6)

g_3 = sns.boxplot(y = 'Inflacion', data = inflacion_c,
            palette = 'Set2',
            ax=axes[1,0],
            **kws_3)
g_3.set_title("Tasa de inflación boxplot", fontsize = 20, fontweight = 'bold'
g_3.set_ylabel('Tasa de inflación [$\$]', fontsize = 20, fontweight = 'bold'
g_3.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_3.tick_params(labelsize = 16, labelcolor = 'darkred')
g_3.grid(color='purple', linestyle='-', alpha = 0.6)

g_4 = sns.boxplot(y = 'Expectativa_vida', data = e_vida_c,
            palette = 'Set2',
            ax=axes[1,1],
            **kws_4)
g_4.set_title("Expectativa de vida boxplot", fontsize = 20, fontweight = 'bol
g_4.set_ylabel('Expectativa de vida [años]', fontsize = 20, fontweight = 'bol
g_4.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_4.tick_params(labelsize = 16, labelcolor = 'darkred')
g_4.grid(color='purple', linestyle='-', alpha = 0.6)

# Controlemos el espacio entre los gráficos
plt.subplots_adjust(wspace=0.15, hspace=0.2)

sns.set_style('white')
f1,axes1 = plt.subplots(2,1, figsize=(24,15))
g_5 = sns.boxplot(y = 'Calidad_de_vida', data = c_vida_c,
            palette = 'Set2',
            ax = axes1[0],
            **kws_5)
g_5.set_title("Índice de calidad de vida boxplot", fontsize = 20, fontweight
g_5.set_ylabel('Índice de calidad de vida', fontsize = 20, fontweight = 'bold
g_5.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_5.tick_params(labelsize = 16, labelcolor = 'darkred')
g_5.grid(color='purple', linestyle='-', alpha = 0.6)
```

```
g_6 = sns.boxplot(y = 'Poblacion', data = pib_c,
            palette = 'Set2',
            ax = axes1[1],
            **kws_6)
g_6.set_title("Población boxplot", fontsize = 20, fontweight = 'bold', color
g_6.set_ylabel('Población', fontsize = 20, fontweight = 'bold', color = 'midn
g_6.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', color
g_6.tick_params(labelsize = 16, labelcolor = 'darkred')
g_6.grid(color='purple', linestyle='-', alpha = 0.6)

# Controlemos el espacio entre los gráficos
plt.subplots_adjust(wspace=0.15, hspace=0.33)
plt.show()
```

Boxplot de las varables





Se aprecia que en los dataframes correspondientes al *PIB*, *tasa de inflación*, la *población* e *índice gini* hay outliers, donde en algunos se esperaba que ocurriera así. Los removeremos con el método descrito anteriormente. Sin embargo, debido a las regresiones que nos interesan, no se eliminarán los *outliers* de la población, además de que al pertenecer al mismo dataframe, nos podrían quedar pocos menos del 60% de los datos si aplicamos dos filtros con el método mencionado.

In [37]:
```
# Calcularemos el umbral superior e inferior auxiliares
pib_us,pib_ui = np.percentile(pib_c['PIB_pc'], [75 ,25])
# Calculamos el rango intercuatílico
iqr_pib = pib_us - pib_ui
# Calculamos los umbrales que ocuparemos para el filtrado
pib_us,pib_ui = pib_us + 1.5*iqr_pib,pib_ui - 1.5*iqr_pib

inflacion_us,inflacion_ui = np.percentile(inflacion_c['Inflacion'], [75 ,25])
iqr_inflacion = inflacion_us - inflacion_ui
inflacion_us,inflacion_ui = inflacion_us + 1.5*iqr_inflacion,inflacion_ui - 1

gini_us,gini_ui = np.percentile(gini_c['Gini'], [75 ,25])
iqr_gini = gini_us - gini_ui
gini_us,gini_ui = gini_us + 1.5*iqr_gini,gini_ui - 1.5*iqr_gini
```

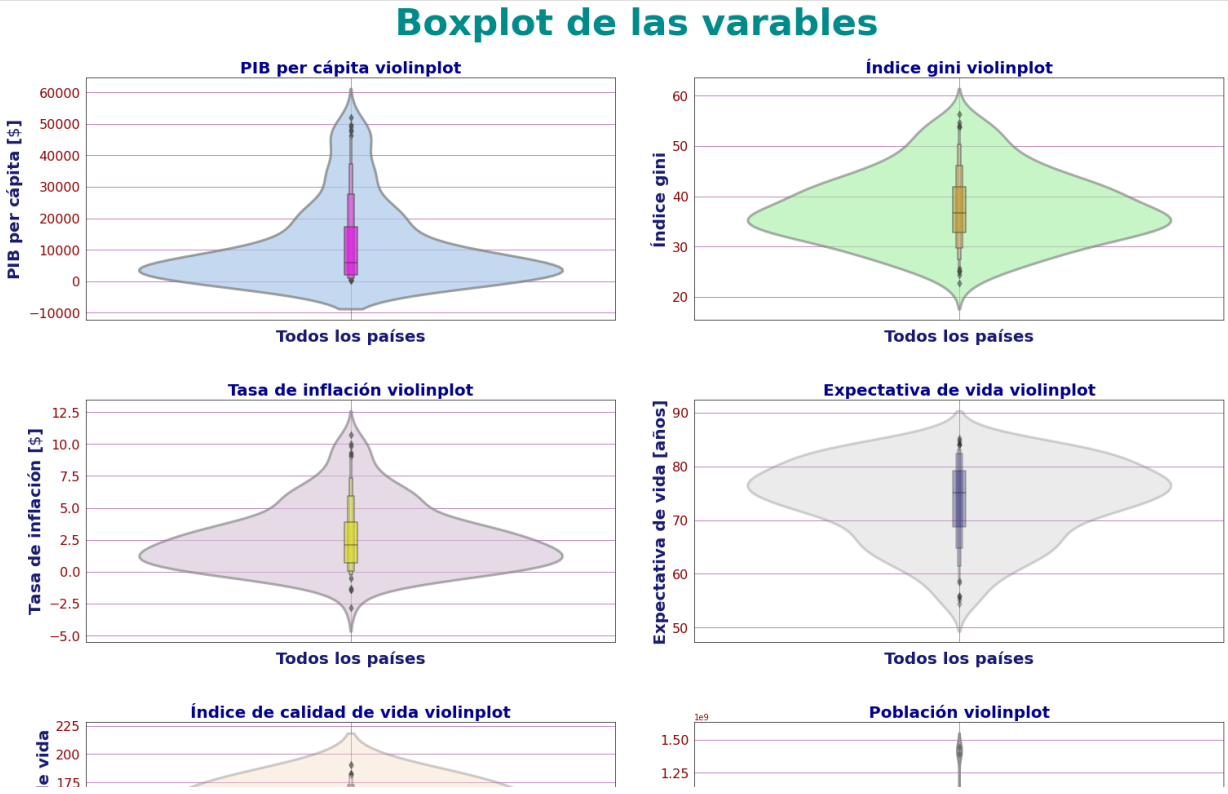
In [38]:
```
# Filtraremos los dataframes para que no contengan datos con outliers, se mos
condicion_1 = (pib_c['PIB_pc'] < pib_us) & (pib_c['PIB_pc'] > pib_ui)
pib_sin_out = pib_c[condicion_1]
# Asignaremos la columna "País" al nuevo dataframe
pib_sin_out['Pais'] = pib_c['Pais']
inflacion_sin_out = inflacion_c[(inflacion_c['Inflacion'] < inflacion_us) & (
inflacion_sin_out['Pais'] = inflacion_c['Pais']
gini_sin_out = gini_c[(gini_c['Gini'] < gini_us) & (gini_c['Gini'] > gini_ui)
gini_sin_out['Pais'] = gini_c['Pais']
```

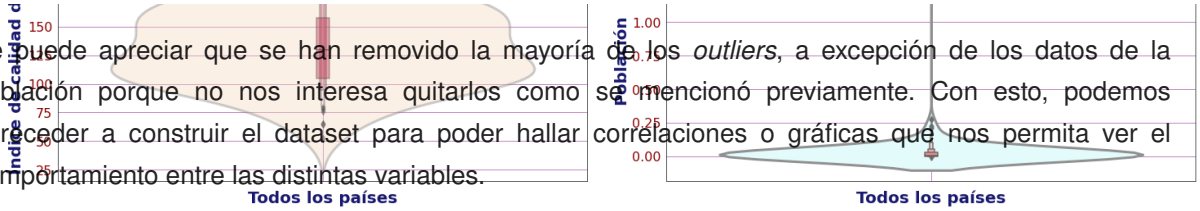
Veamos ahora con los violinplots que ya se filtraron la mayoría de los *outliers*.

In [176…

```
sns.set_style('white')
f,axes = plt.subplots(3,2, figsize=(25,20))
f.suptitle("Boxplot de las varables", fontsize = 45, color = 'darkcyan', font


g_1_1 = sns.violinplot(y = 'PIB_pc', data = pib_sin_out,
               linewidth = 3, inner = None,
               palette = 'pastel', edgecolor = '#aaff00',
               ax=axes[0,0])
g_1_2 = sns.boxenplot(y = 'PIB_pc', data = pib_sin_out,
               color="magenta", width = 0.025,
               ax=axes[0,0])
plt.setp(g_1_1.collections, alpha=.7)
g_1_1.set_title("PIB per cápita violinplot", fontsize = 20, fontweight = 'bol
g_1_1.set_ylabel('PIB per cápita [$\$]', fontsize = 20, fontweight = 'bold',
g_1_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_1_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_1_1.grid(color='purple', linestyle='-', alpha = 0.6)


g_2_1 = sns.violinplot(y = 'Gini', data = gini_sin_out,
               linewidth = 3, inner = None,
               color = 'palegreen', edgecolor = '#aaff00',
               ax=axes[0,1])
g_2_2 = sns.boxenplot(y = 'Gini', data = gini_sin_out,
               color="goldenrod", width = 0.025,
               ax=axes[0,1])
plt.setp(g_2_1.collections, alpha=.6)
g_2_1.set_title("Índice gini violinplot", fontsize = 20, fontweight = 'bold',
g_2_1.set_ylabel('Índice gini', fontsize = 20, fontweight = 'bold', color = '
g_2_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_2_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_2_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_3_1 = sns.violinplot(y = 'Inflacion', data = inflacion_sin_out,
               linewidth = 3, inner = None,
               color = 'thistle', edgecolor = '#aaff00',
               ax=axes[1,0])
g_3_2 = sns.boxenplot(y = 'Inflacion', data = inflacion_sin_out,
               color="yellow", width = 0.025,
               ax=axes[1,0])
plt.setp(g_3_1.collections, alpha=.6)
g_3_1.set_title("Tasa de inflación violinplot", fontsize = 20, fontweight = '
g_3_1.set_ylabel('Tasa de inflación [$\$]', fontsize = 20, fontweight = 'bol
g_3_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_3_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_3_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_4_1 = sns.violinplot(y = 'Expectativa_vida', data = e_vida_c,
               linewidth = 3, inner = None,
               color = 'silver', edgecolor = '#aaff00',
               ax=axes[1,1])
g_4_2 = sns.boxenplot(y = 'Expectativa_vida', data = e_vida_c,
               color="navy", width = 0.025,
               ax=axes[1,1])
plt.setp(g_4_1.collections, alpha=.3)
g_4_1.set_title("Expectativa de vida violinplot", fontsize = 20, fontweight =
g_4_1.set_ylabel('Expectativa de vida [años]', fontsize = 20, fontweight = 'b
```

```
g_4_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_4_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_4_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_5_1 = sns.violinplot(y = 'Calidad_de_vida', data = c_vida_c,
                       linewidth = 3, inner = None,
                       color = 'peachpuff', edgecolor = '#aaff00',
                       ax=axes[2,0])
g_5_2 = sns.boxenplot(y = 'Calidad_de_vida', data = c_vida_c,
                       color="crimson", width = 0.025,
                       ax=axes[2,0])
plt.setp(g_5_1.collections, alpha=.4)
g_5_1.set_title("Índice de calidad de vida violinplot", fontsize = 20, fontwe
g_5_1.set_ylabel('Índice de calidad de vida', fontsize = 20, fontweight = 'bo
g_5_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_5_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_5_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_6_1 = sns.violinplot(y = 'Poblacion', data = pib_sin_out,
                       linewidth = 3, inner = None,
                       color = 'lightcyan',
                       ax=axes[2,1])
g_6_2 = sns.boxenplot(y = 'Poblacion', data = pib_sin_out,
                       color="lightcoral", width = 0.025,
                       ax=axes[2,1])
g_6_1.set_title("Población violinplot", fontsize = 20, fontweight = 'bold', c
g_6_1.set_ylabel('Población', fontsize = 20, fontweight = 'bold', color = 'mi
g_6_1.set_xlabel('Todos los países', fontsize = 20, fontweight = 'bold', colo
g_6_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_6_1.grid(color='purple', linestyle='-', alpha = 0.6)

# Controlemos el espacio entre los gráficos
plt.subplots_adjust(wspace=0.15, hspace=0.33)
plt.show()
```





Se puede apreciar que se han removido la mayoría de los *outliers*, a excepción de los datos de la población porque no nos interesa quitarlos como se mencionó previamente. Con esto, podemos proceder a construir el dataset para poder hallar correlaciones o gráficas que nos permita ver el comportamiento entre las distintas variables.

Transformación del dataframe

Primero, procederemos con la creación de la base de datos que usaremos para la regresión lineal del PIB con la expectativa de vida, pero también veremos si hay alguna clase de relación explícita hay entre el PIB y la inflación para valores pequeños. Para esto unamos los dataframes por medio de la columna *País*. También se agregará la variable del logarítmo natiural del *PIB* como se mencionó en la refrencia explicada en la parte teórica.

In [40]:
```
# Creamos la columna que contiene el valor del logaritmo del PIB y aunque no
pib_sin_out['log_PIB_pc'] = np.log(np.abs(pib_sin_out['PIB_pc']))
```

In [41]:
```
# Unimos las columnas con merge eliminando los registros que no hagan match e
df_aux_regresion = pd.merge(pib_sin_out, e_vida_c, on='Pais')
df_aux_regresion = pd.merge(df_aux_regresion, inflacion_sin_out, on='Pais')
df_aux_regresion = pd.merge(df_aux_regresion, c_vida_c, on='Pais')
df_aux_regresion = pd.merge(df_aux_regresion, gini_sin_out, on='Pais')
df_aux_regresion.drop('PIB_pc', axis=1, inplace=True)
df_aux_regresion.head()
```

Out[41]:

	Pais	Poblacion	log_PIB_pc	Expectativa_vida	Inflacion	Calidad_de_vida	Gini
0	China	1444216107	9.278933	77.47	4.30	103.15	38.50
1	Japan	126050804	10.682744	85.03	0.40	162.32	32.90
2	Germany	83900473	10.810697	81.88	1.40	176.76	31.90
3	India	1393409038	7.757051	70.42	5.91	104.52	35.70
4	United Kingdom	68207116	10.666977	81.77	1.50	158.99	34.95

Veamos las características del dataframe

In [42]:
```
df_aux_regresion.describe()
```

Out[42]:

	Poblacion	log_PIB_pc	Expectativa_vida	Inflacion	Calidad_de_vida	Gini
count	5.900000e+01	59.000000	59.000000	59.000000	59.000000	59.000000
mean	9.364514e+07	9.428027	77.858136	2.587119	125.357627	35.739831
std	2.575006e+08	0.985516	4.123835	2.602889	29.617973	6.460622
min	1.215584e+06	7.134094	67.470000	-1.300000	65.270000	24.400000
25%	7.797106e+06	8.744434	75.450000	0.610000	103.015000	31.850000

	Poblacion	log_PIB_pc	Expectativa_vida	Inflacion	Calidad_de_vida	Gini
50%	2.149731e+07	9.529303	77.710000	1.910000	121.650000	35.350000
75%	6.681665e+07	10.158569	81.640000	3.880000	150.205000	39.525000

De esto, podemos observar que la limpieza de datos fue un éxito y que no hay irregularidades en la descripción estadística de los distintos datos.

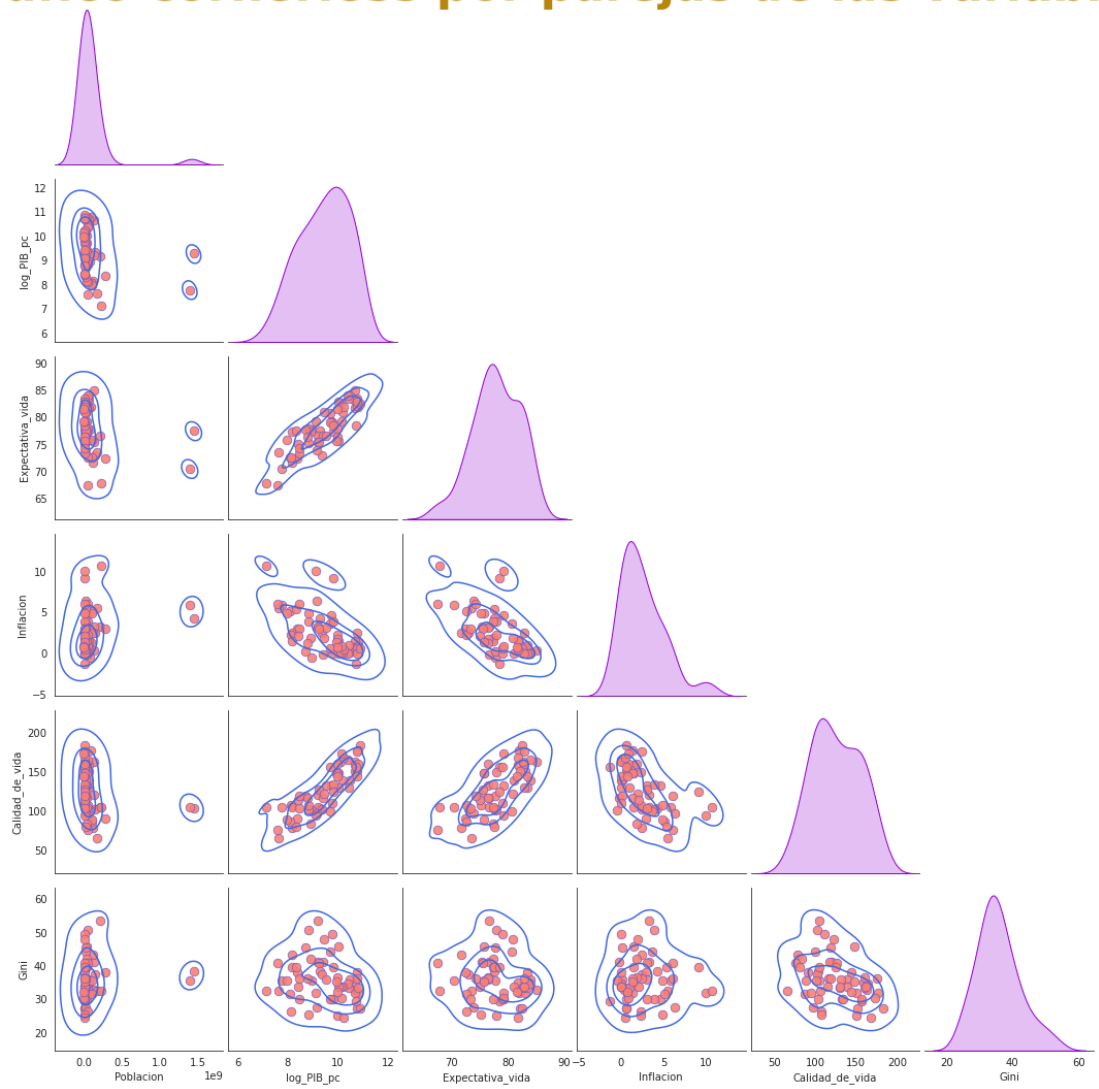
In [43]:
```
print(df_aux_regresion.shape)
```

```
(59, 7)
```

Vemos que a pesar de ya no contar con 211 datos, aún contamos con suficientes datos. Sin embargo, después haremos una nueva versión de este dataframe donde se rellenarán los datos faltantes por el promedio de los valores de la columna puesto que esto no afecta al valor de los estimadores, pero para ello tenemos que obtener las visualizaciones de las relaciones entre las variables para proponer un moderlo.

In [189…
```
pp = sns.pairplot(df_aux_regresion,  diag_kind='kde', markers="o", corner=Tru
                  plot_kws={"color": 'salmon', 'edgecolor':'slateblue', "s":7
                  grid_kws=dict(diag_sharey=False),
                  diag_kws=dict(fill=True, color='darkviolet'))
pp.fig.suptitle("Gráfico cornerless por parejas de las variables", fontsize =
pp.map_lower(sns.kdeplot, levels=4, color="royalblue")
pp.fig.set_size_inches(15,15)
plt.show()
```

Gráfico cornerless por parejas de las variables

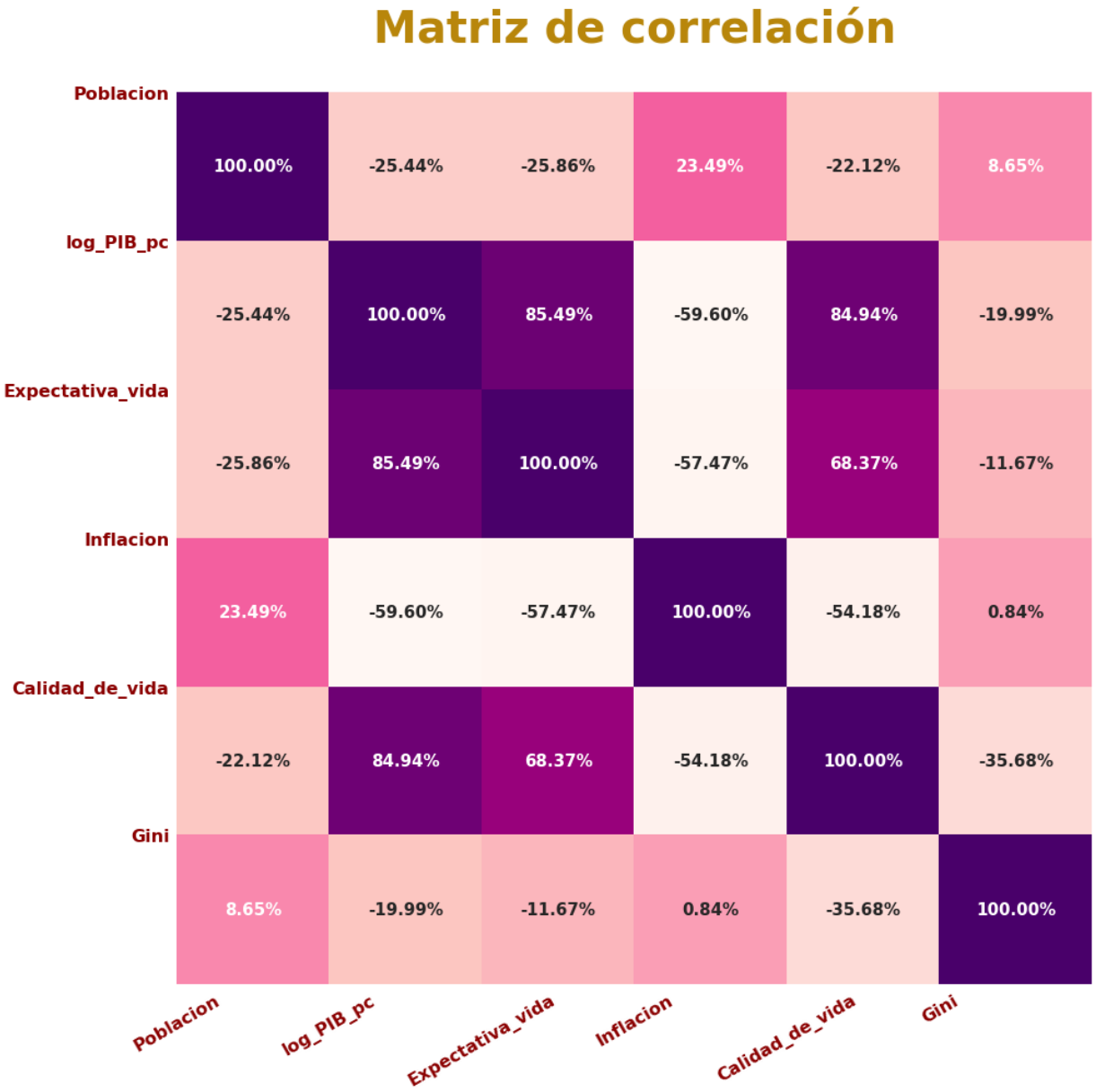


Se puede apreciar que hay variables que se tienden a agrupar, pero este análisis se hará detalladamente hasta quue hagamos la regresión con el método de K-means++. Por otro lado, vemos que visualmente se insinua que hay correlaciones entre distintas variables, pero ahora consideremos la matriz de covrianzas donde nuestro umbral para tomar en cuenta las variables para una regresión es que satisfagan $R^2 \ge 0.5$.

In [53]:
```
matriz_corr = df_aux_regresion.corr()
```

In [100…

```
plt.figure(figsize=(15, 15))
g = sns.heatmap(matriz_corr, annot = True, cmap = 'RdPu',
            annot_kws={"size": 15, 'fontweight':'bold'},
            cbar_kws={'label': 'Correlación'},
            fmt=".2%", cbar=False)
plt.title('Matriz de correlación', fontsize = 40,
          fontweight = 'bold', y = 1.05,
          color = 'darkgoldenrod')
plt.xticks(range(df_aux_regresion.select_dtypes(['number']).shape[1]),
           df_aux_regresion.select_dtypes(['number']).columns,
           fontsize = 16, fontweight = 'bold',
           rotation = 30, color = 'darkred')
plt.yticks(range(df_aux_regresion.select_dtypes(['number']).shape[1]),
           df_aux_regresion.select_dtypes(['number']).columns,
           fontsize = 16, fontweight = 'bold',
           rotation = 0, color = 'darkred')
plt.show()
```



Tal como vemos en el gráfico y la matriz de varianzas, solo tenemos evidencia de una relación fuerte

entre el logaritmo del PIB y la expectativa de vida con la calidad de vida. Por otro lado, también se muestra una relación, aunque no tan fuerte, entre el logarítmo del PIB y la tasa de inflación. Sin embargo, como todas estas variables están correlacionadas entre sí, no podemos hacer una regresión múltiple entre estas puesto que hay evidencia de multicolinealidad por lo que se harán las regresiones por separado haciéndolas entre las pareja que presenten estén más correlacionadas entre sí. La multicolinealidad está dada por nuestro criterio de selección en base a un $R^2 \ge 0.5$.

Ahora, creemos un dataframe adecuado con más datos que contenga a estas tres varaibles. En todos los casos, no nos interesa eliminar observaciones, por lo que no se hará la unión de los dataframes con el método *merge*, sino con el método *concat* puesto que las filas que no estén en común en ambos dataframes las rellena con valores *NaN*.

Transformación del dataframe

Tal y como se mencionó anteriormente, se usará el parámetro *outer* dentro del método *merge*, pero de econometría sabemos que llenar los datos faltantes con el promedio no afecta el valor de los estimadores. Sin embargo, esto solo nos funcionará si lo hacemos con los logaritmos de las observaciones puesto que la regresión lineal se hará sobre estos tal y como sugieren los artículos y bibliografía.

In [165…
```
df_regresion_aux_1 = pd.merge(pib_sin_out, inflacion_sin_out, how = "outer",
df_regresion_aux_2 = pd.merge(df_regresion_aux_1, e_vida_c, how = "outer", on
df_regresion = pd.merge(df_regresion_aux_2, c_vida_c, how = "outer", on = ['P
df_regresion.head()
```

Out[165…

	Pais	PIB_pc	Poblacion	log_PIB_pc	Inflacion	Expectativa_vida	Calidad_de_vida
0	China	10710.0	1.444216e+09	9.278933	4.30	77.47	103.15
1	Japan	43597.0	1.260508e+08	10.682744	0.40	85.03	162.32
2	Germany	49548.0	8.390047e+07	10.810697	1.40	81.88	176.76
3	India	2338.0	1.393409e+09	7.757051	5.91	70.42	104.52
4	United Kingdom	42915.0	6.820712e+07	10.666977	1.50	81.77	158.99

Quitemos la columna del PIB per cápita y población puesto que solo necesitamos la del logaritmo.

In [168…
```
df_regresion.drop('PIB_pc', axis=1, inplace=True)
df_regresion.drop('Poblacion', axis=1, inplace=True)
df_regresion.head()
```

Out[168…

	Pais	log_PIB_pc	Inflacion	Expectativa_vida	Calidad_de_vida

	Pais	log_PIB_pc	Inflacion	Expectativa_vida	Calidad_de_vida
0	China	9.278933	4.30	77.47	103.15
1	Japan	10.682744	0.40	85.03	162.32
2	Germany	10.810697	1.40	81.88	176.76
3	India	7.757051	5.91	70.42	104.52

Verifiquemos que si se hayan conservado los registros.

In [169…
```
df_regresion.shape
```

Out[169… `(238, 5)`

Claramente es un número mayor al número de filas obtenidas en el dataframe utilizado para la matriz de varianzas, implica que no se borraron los registros. Veamos columna por columna los valores faltante o nulos.

In [170…
```
print("\nlog_PIB per cápita nan = " + str(df_regresion['log_PIB_pc'].isna().s
print("\nInflación nan = " + str(df_regresion['Inflacion'].isna().sum()) +  "
print("\nExpectativa de vida nan = " + str(df_regresion['Expectativa_vida'].i
print("\nCalidad de vida nan = " + str(df_regresion['Calidad_de_vida'].isna()
```

```
log_PIB per cápita nan = 45
log_PIB_pc nulls = 45

Inflación nan = 75
Inflación nulls = 75

Expectativa de vida nan = 36
Expectativa de vida nulls = 36

Calidad de vida nan = 155
Calidad de vida nulls = 155
```

Entonces, al tener *NaN* debemos rellenarlos con el promedio tal cómo se dijo anteriormente.

In [171…
```
promedio_pib = df_regresion['log_PIB_pc'].mean(skipna = True)
promedio_expectativa = df_regresion['Expectativa_vida'].mean(skipna = True)
promedio_inflacion = df_regresion['Inflacion'].mean(skipna = True)
promedio_calidad = df_regresion['Calidad_de_vida'].mean(skipna = True)

df_regresion['log_PIB_pc'].fillna(value = promedio_pib, inplace = True)
df_regresion['Expectativa_vida'].fillna(value = promedio_expectativa, inplace
df_regresion['Inflacion'].fillna(value = promedio_inflacion, inplace = True)
df_regresion['Calidad_de_vida'].fillna(value = promedio_calidad, inplace = Tr
print("Valores faltantes del dataframe: " + str(df_regresion.isna().sum().sum
```

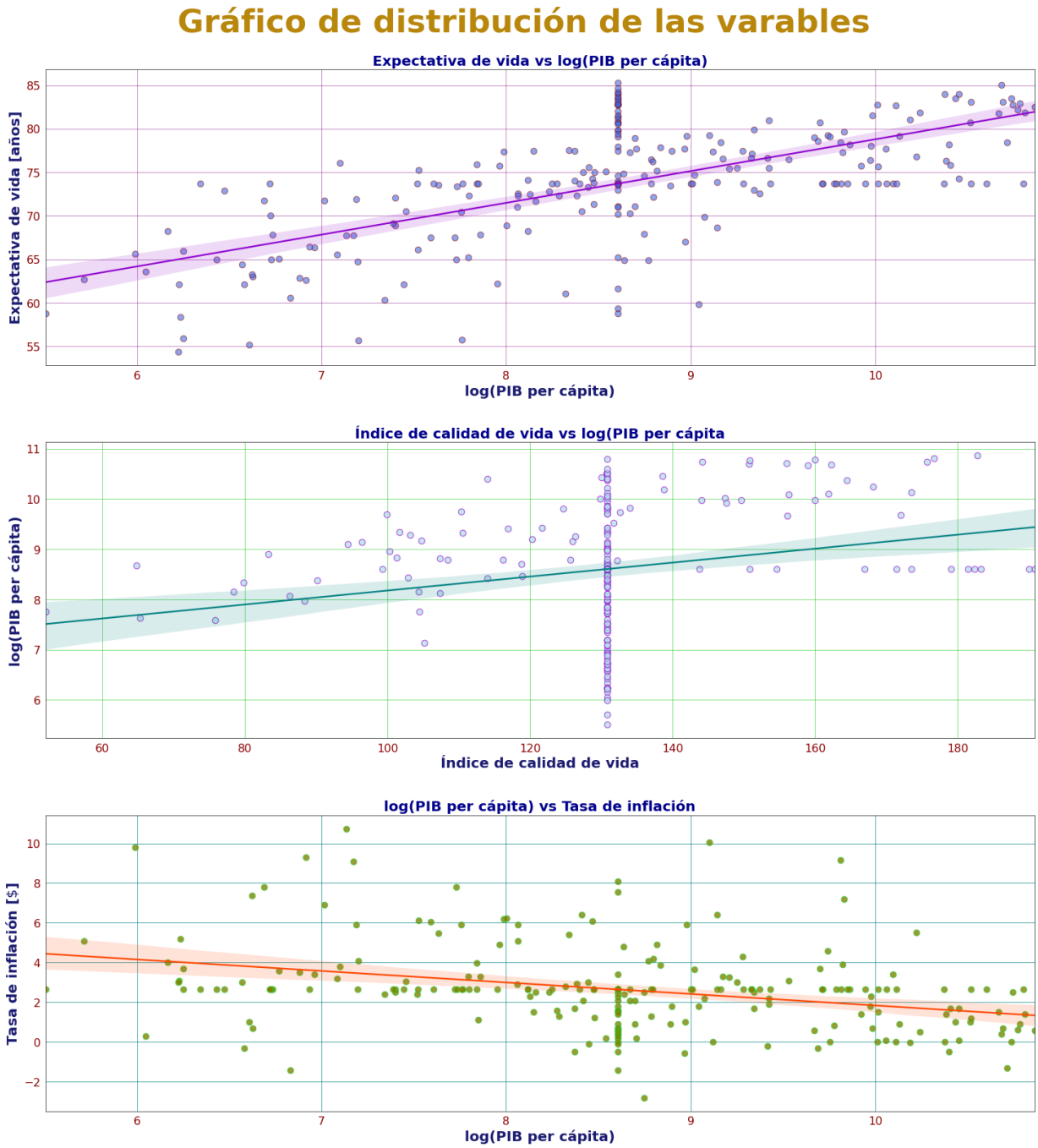
```
Valores faltantes del dataframe: 0
```

Vemos que en todo el dataframe ya no hay valores faltantes. Por lo que ahora mostraremos con más detalle las gráficas con su respectava recta de regresión. Cabe mencionar que tal y cómo se especificó anteriormente, se harán las regresiones entre las variables que presenten más correlación entre sí. Si nos fijamos en la matriz de covarianzas, veremos que todas las variables presentan por separado su

mayor correlación con respecto al logaritmo del PIB per cápita, por lo que son las regresiones que nos itneresan

In [208…

```
sns.set_style('white')
f,axes = plt.subplots(3,1, figsize=(25,27))
f.suptitle("Gráfico de distribución de las varables", fontsize = 45, color =

g_1 = sns.regplot(x = 'log_PIB_pc', y = 'Expectativa_vida', data = df_regresi
            scatter_kws={"color": 'royalblue', 'edgecolor':'darkred', "s":75,
            line_kws={"color": "#8e00ce"},
            ax=axes[0])
g_1.set_title("Expectativa de vida vs log(PIB per cápita)", fontsize = 20, fo
g_1.set_ylabel('Expectativa de vida [años]', fontsize = 20, fontweight = 'bol
g_1.set_xlabel('log(PIB per cápita)', fontsize = 20, fontweight = 'bold', col
g_1.tick_params(labelsize = 16, labelcolor = 'darkred')
g_1.grid(color='purple', linestyle='-', alpha = 0.6)

g_2 = sns.regplot(x = 'Calidad_de_vida', y = 'log_PIB_pc', data = df_regresio
            scatter_kws={"color": 'powderblue', 'edgecolor':'darkviolet', "s"
            line_kws={"color": "teal"},
            ax=axes[1])
g_2.set_title("Índice de calidad de vida vs log(PIB per cápita", fontsize = 2
g_2.set_ylabel('log(PIB per cápita)', fontsize = 20, fontweight = 'bold', col
g_2.set_xlabel('Índice de calidad de vida', fontsize = 20, fontweight = 'bold
g_2.tick_params(labelsize = 16, labelcolor = 'darkred')
g_2.grid(color='limegreen', linestyle='-', alpha = 0.8)

g_3 = sns.regplot(x = 'log_PIB_pc', y = 'Inflacion', data = df_regresion,
            scatter_kws={"color": 'olive', 'edgecolor':'limegreen', "s":75, "
            line_kws={"color": "orangered"},
            ax=axes[2])
g_3.set_title("log(PIB per cápita) vs Tasa de inflación", fontsize = 20, font
g_3.set_ylabel('Tasa de inflación [$\$]', fontsize = 20, fontweight = 'bold'
g_3.set_xlabel('log(PIB per cápita)', fontsize = 20, fontweight = 'bold', col
g_3.tick_params(labelsize = 16, labelcolor = 'darkred')
g_3.grid(color='teal', linestyle='-', alpha = 0.9)

# Controlemos el espacio entre los gráficos
plt.subplots_adjust(wspace=0.18, hspace=0.26)
plt.show()
```



Se puede ver que por el método de *fill* de datos se acumulan varias observaciones a lo largo de una línea recta en el eje horizontal, siendo esto por usar la media para rellenar las observaciones faltantes.

Por otro lado, en la siguiente sección usaremos algoritmos de machine learning para obtener las regresiones lineales señaladas y los clusters auziliándonos de etiquetas que crearemos estratégicamente.

Bibliografía:

I. Renta como indicador de bienestar
II. Life expectancy vs. GDP per capita, 1543 to 2018
III. Patterns in the relationship between life expectancy and gross domestic product in Russia in 2005–15: a cross-sectional analysis
IV. K-means++
V. Measuring Global Inequality: Median Income, GDP per Capita, and the Gini Index
VI. The Relationship between Income, Consumption and GDP: A Time Series, Cross-Country Analysis
VII. Clustering Algorithms for Economic Policy Guidance
VIII. Quality of Life Index by Country 2021
IX. Gini Coefficient by Country up to 2019
X. GDP Ranked by Country 2021
XI. Why Does Inflation Increase With GDP Growth?
XII. Feature Engineering – How to Detect and Remove Outliers (with Python Code)
XIII. Economic Growth and Inflation
XIV. A Reassessment of the Relationship between GDP and Life Satisfaction
XV. Life Expectancy of the World Population 2021
XVI. Inflation Rate by Country 2021

In [ ]: