Tarea_N°4_Visualización_de_la_Información_Castillo_Castro_Cisternas_V

October 29, 2024

Universidad del Desarrollo Profesor: Fernando Becerra Saavedra

1 Visualización de la Información: Tarea N° 4

24 de Octubre de 2024

Nombre Estudiante(s): `Jaime Castillo Castro - Kurt Castro Ortega - Sofía Vits Contreras - Erick Cisterna Contreras`

Dataset: World Happiness Report- 2024

Fuente: https://www.kaggle.com/datasets/jainaru/world-happiness-report-2024-yearly-updated/data

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import geopandas as gpd
import matplotlib as mpl
from matplotlib import lines
from matplotlib import patches
from matplotlib.patheffects import withStroke
from matplotlib.patches import Rectangle
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from matplotlib.colors import LinearSegmentedColormap
from matplotlib.transforms import Affine2D
from matplotlib.collections import PathCollection
from mpl_toolkits.axisartist.grid_finder import DictFormatter
import mpl_toolkits.axisartist.floating_axes as floating_axes
from flexitext import flexitext
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from matplotlib.patches import Arrow
from matplotlib.patches import Circle
```

```python
from matplotlib import patches
from matplotlib.collections import LineCollection
from matplotlib.colors import LinearSegmentedColormap
import matplotlib.patches as mpatches
```

1.0.1 I. GENERALIDADES

1. Descripción general del dataset El **World Happiness Report 2024** es un conjunto de datos que evalúa el bienestar de los países a nivel global. Utiliza varios indicadores que intentan capturar la felicidad de los ciudadanos a través de múltiples dimensiones económicas, sociales y psicológicas.

2. Columnas Dataset Principal

1. **Country name**: El nombre del país evaluado.
2. **year**: El año en el que se tomaron las mediciones.
3. **Life Ladder**: Una medida de la felicidad subjetiva de los ciudadanos (en una escala de 0 a 10), conocida como el "escalera de la vida", donde las personas evalúan su vida actual.
4. **Log GDP per capita**: El PIB per cápita en una escala logarítmica. Representa la riqueza económica promedio de los ciudadanos en el país.
5. **Social support**: Un índice que mide el nivel de apoyo social, es decir, si las personas tienen alguien en quien confiar en momentos de necesidad.
6. **Healthy life expectancy at birth**: La esperanza de vida saludable al nacer, que mide cuántos años se espera que una persona viva en buena salud.
7. **Freedom to make life choices**: Índice que mide la percepción de las personas sobre su libertad para tomar decisiones importantes en su vida.
8. **Generosity**: Índice de generosidad, basado en la cantidad de donaciones y ayuda mutua dentro del país.
9. **Perceptions of corruption**: Mide la percepción pública sobre los niveles de corrupción en las instituciones del gobierno y los negocios.
10. **Positive affect**: La cantidad de experiencias positivas (como la risa y el disfrute) que una persona experimenta.
11. **Negative affect**: Mide el grado de experiencias negativas (como tristeza y enojo) que las personas experimentan.

3. Columnas dataset secundario

12. **Continent**: Esta columna proviene desde un dataset adicional que permite agrupar a los países por continente (por ejemplo, Asia, Europa, África, etc.).

4. Propósito del dataset El objetivo de este dataset es proporcionar una medida comprensiva de la felicidad global. Los indicadores económicos como el PIB per cápita y la esperanza de vida saludable se combinan con variables sociales y emocionales, como el apoyo social y la percepción de corrupción, para crear una imagen más completa del bienestar de los ciudadanos en cada país.

1.0.2 II. DESARROLLO

Paso 1: Carga de datos A continuación, se carga la base de datos World-happiness-report-updated_2024 y adicionalmente se carga una base de datos de los paises por continente, la cual

tiene por finalidad realizar un cruce de información y agregar una columna calculada en la base de datos inicial.

[2]:
```
# Cargar el dataset principal
ruta = "data/World-happiness-report-updated_2024.csv"
df = pd.read_csv(ruta, encoding='latin1')

# Cargar el dataset de países y continentes
ruta_paises = "data/Countries by continents.csv"
df_paises_continente = pd.read_csv(ruta_paises, sep=';')

# los nombres de las columnas coincidan para la combinacion
df_paises_continente = df_paises_continente.rename(columns={'Country': 'Country␣
↪name'})

# Combinar ambos datasets usando la columna 'Country name'
df_merged = pd.merge(df, df_paises_continente, on='Country name', how='left')

# Verificar las primeras filas para asegurarse de que el continente fue agregado
print(df_merged.head())
```

```
  Country name  year  Life Ladder  Log GDP per capita  Social support  \
0  Afghanistan  2008        3.724               7.350           0.451
1  Afghanistan  2009        4.402               7.509           0.552
2  Afghanistan  2010        4.758               7.614           0.539
3  Afghanistan  2011        3.832               7.581           0.521
4  Afghanistan  2012        3.783               7.661           0.521

   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                              50.5                         0.718       0.164
1                              50.8                         0.679       0.187
2                              51.1                         0.600       0.118
3                              51.4                         0.496       0.160
4                              51.7                         0.531       0.234

   Perceptions of corruption  Positive affect  Negative affect Continent
0                      0.882            0.414            0.258      Asia
1                      0.850            0.481            0.237      Asia
2                      0.707            0.517            0.275      Asia
3                      0.731            0.480            0.267      Asia
4                      0.776            0.614            0.268      Asia
```

Paso 2: Análisis exploratorio

[3]:
```
# Mostrar el tamaño del dataframe
df_merged.shape
```

[3]: (2363, 12)

[4]:
```python
# Separar variables numéricas
numerical_variables = df_merged.dtypes[df_merged.dtypes != 'object'].index
print('El número de variables numéricas es: ', len(numerical_variables))
print('Las variables numéricas son:', numerical_variables)
```

```
El número de variables numéricas es:  10
Las variables numéricas son: Index(['year', 'Life Ladder', 'Log GDP per capita',
'Social support',
       'Healthy life expectancy at birth', 'Freedom to make life choices',
       'Generosity', 'Perceptions of corruption', 'Positive affect',
       'Negative affect'],
      dtype='object')
```

[5]:
```python
# Separar variables categóricas
categorical_variables = df_merged.dtypes[df_merged.dtypes == 'object'].index
print('El número de variables categóricas es', len(categorical_variables))
print('Las variables categóricas son:', categorical_variables)
```

```
El número de variables categóricas es 2
Las variables categóricas son: Index(['Country name', 'Continent'],
dtype='object')
```

[6]:
```python
# Mostrar datos estadísticos de variables numéricas
df_merged.describe()
```

[6]:

	year	Life Ladder	Log GDP per capita	Social support
count	2363.000000	2363.000000	2335.000000	2350.000000
mean	2014.763860	5.483566	9.399671	0.809369
std	5.059436	1.125522	1.152069	0.121212
min	2005.000000	1.281000	5.527000	0.228000
25%	2011.000000	4.647000	8.506500	0.744000
50%	2015.000000	5.449000	9.503000	0.834500
75%	2019.000000	6.323500	10.392500	0.904000
max	2023.000000	8.019000	11.676000	0.987000

	Healthy life expectancy at birth	Freedom to make life choices
count	2300.000000	2327.000000
mean	63.401828	0.750282
std	6.842644	0.139357
min	6.720000	0.228000
25%	59.195000	0.661000
50%	65.100000	0.771000
75%	68.552500	0.862000
max	74.600000	0.985000

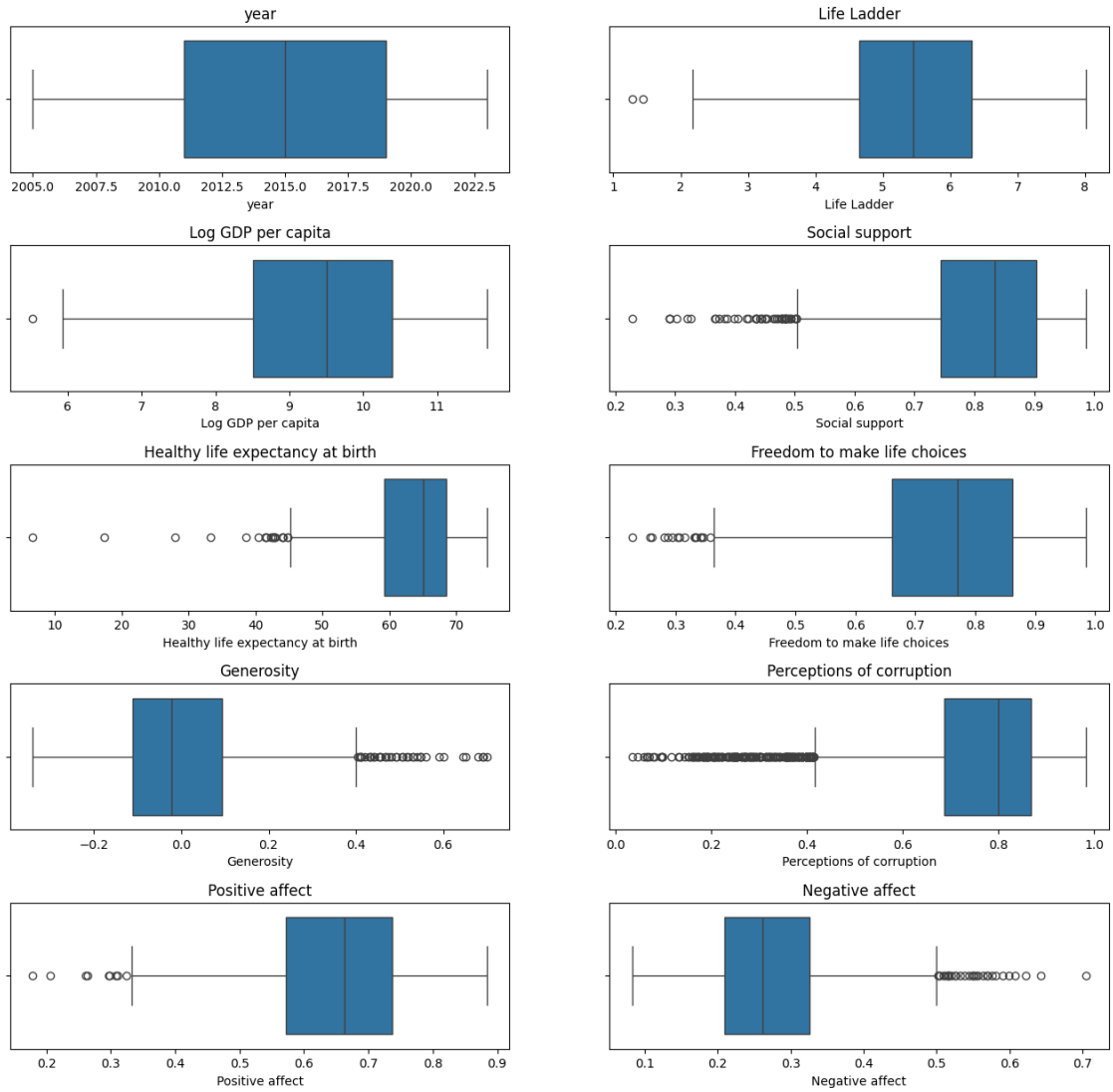
	Generosity	Perceptions of corruption	Positive affect
count	2282.000000	2238.000000	2339.000000
mean	0.000098	0.743971	0.651882

```
std       0.161388                    0.184865         0.106240
min      -0.340000                    0.035000         0.179000
25%      -0.112000                    0.687000         0.572000
50%      -0.022000                    0.798500         0.663000
75%       0.093750                    0.867750         0.737000
max       0.700000                    0.983000         0.884000

       Negative affect
count      2347.000000
mean          0.273151
std           0.087131
min           0.083000
25%           0.209000
50%           0.262000
75%           0.326000
max           0.705000
```

[7]:
```
# Mostrar datos estadísticos de variables categóricas
df_merged.describe(include=object)
```

[7]:
```
        Country name Continent
count           2363      2363
unique           165         6
top        Argentina    Europe
freq              18       718
```

[8]:
```
# Revisión de existencia de Outliers en variables numéricas
cols_num = ['year', 'Life Ladder', 'Log GDP per capita', 'Social support',␣
 ↪'Healthy life expectancy at birth',
            'Freedom to make life choices', 'Generosity', 'Perceptions of␣
 ↪corruption', 'Positive affect', 'Negative affect']

# Crear la figura con 5 filas y 2 columnas
fig, ax = plt.subplots(nrows=5, ncols=2, figsize=(16, 15))  # Tamaño ajustado␣
 ↪para mejor visualización
fig.subplots_adjust(hspace=0.5)  # Ajuste del espacio vertical entre gráficos

# Recorrer las variables numéricas y graficar cada boxplot
for i, col in enumerate(cols_num):
    row = i // 2  # Índice de la fila
    col_idx = i % 2  # Índice de la columna
    sns.boxplot(x=col, data=df_merged, ax=ax[row, col_idx])  # Graficar en el␣
 ↪eje correspondiente
    ax[row, col_idx].set_title(col)  # Establecer título de cada boxplot

# Mostrar el gráfico
plt.show()
```



Se observa presencia de outliers en todas las variables, excepto la correspondiente a los años. Life Ladder y Log GDP per cápita son las variables que presentan menor cantidad de valores atípicos, mientras que Perceptions of corruption la que presenta una mayor cantidad.

En este ensayo nos enfocaremos en la variable Life Ladder, considerando el resto de las variables cuando analicemos su influencia sobre la percepción de la felicidad de la la población de un país.

En este punto realizamos un análisis detallado de los boxplot graficados.

1. **Year (Año)**:

- El rango de los datos va de 2005 a 2022, donde se encuentran distribuidos los datos mayormente en los años 2011 a 2018. No se ven outliers.

2. **Life Ladder (Escalera de vida)**:

- Mide la percepción de las personas sobre su bienestar general. La mediana está alrededor

de 5.5, lo que significa que, en general, las personas evalúan su felicidad en ese nivel en una escala de 0 a 10. Hay algunos datos atípicos en el extremo izquierdo, lo que sugiere que en algunos países las personas se sienten mucho menos felices que en otros países.

3. **Log GDP per capita (Logaritmo del PIB per cápita)**:

- Representa el bienestar económico. Un PIB per cápita más alto generalmente está asociado con mayor bienestar. La mediana está alrededor de 9.5, lo que sugiere que la mayoría de los países tienen un nivel económico moderado. No hay muchos datos atípicos, lo que indica que la mayoría de los países tienen valores similares en esta medida.

4. **Social support (Apoyo social)**:

- Mide la percepción de las personas sobre el apoyo que reciben de su entorno. La mediana está en 0.8, lo que sugiere que la mayoría de las personas sienten que tienen un buen apoyo social. Los outliers en el extremo izquierdo, sugiere que personas de algunos países sienten que tienen muy poco apoyo social.

5. **Healthy life expectancy at birth (Esperanza de vida saludable al nacer)**:

- Representa cuántos años una persona puede esperar vivir con buena salud. La mediana está alrededor de 60 años, con algunos valores atípicos más bajos. Lo que indica que en general, las personas estiman llegar con buena salud hasta un poco antes de ser adultos mayores.

6. **Freedom to make life choices (Libertad para tomar decisiones)**:

- Mide la percepción de las personas sobre su libertad para tomar decisiones. La mediana está alrededor de 0.75, lo que sugiere que la mayoría de las personas sienten que tienen una cantidad considerable de libertad para tomar decisiones importantes. Hay datos atípicos en el extremo izquierdo, lo que significa que algunas personas sienten muy poca libertad.

7. **Generosity (Generosidad)**:

- Mide la disposición de las personas a ser generosas. La mediana es cercana a cero, lo que indica que, en promedio, la generosidad no es un factor relevante. Existen muchos valores atípicos al lado derecho del boxplot, lo que significa que algunos países son mucho más generosos que otros.

8. **Perceptions of corruption (Percepciones de corrupción)**:

- Mide cómo perciben las personas los niveles de corrupción en sus países. La mediana está cerca de 0.75, lo que indica que muchas personas consideran que la corrupción es un problema moderado o serio. Existen muchos valores atípicos en la parte izquierda, lo que sugiere que algunos países perciben niveles muy bajos de corrupción.

9. **Positive affect (Afecto positivo)**:

- Mide el grado de emociones positivas que experimentan las personas, como felicidad o risas. La mediana está alrededor de 0.7, lo que indica que la mayoría de las personas experimentan emociones positivas regularmente. Existen algunos valores atípicos en la parte inferior, lo que sugiere que algunas personas experimentan niveles muy bajos de emociones positivas.

10. **Negative affect (Afecto negativo)**:

- Mide el grado de emociones negativas, como preocupación o tristeza. La mediana está alrededor de 0.2 y 0.3, lo que indica que la mayoría de las personas no experimentan emociones

negativas de manera frecuente. Hay muchos valores atípicos en la parte superior, lo que indica que algunas personas experimentan emociones negativas con mayor frecuencia que en otros países.

[9]:
```
# Análisis de la existencia de valores nulos
valores_nulos = df_merged.isnull().sum()
print("Valores faltantes iniciales por columna:\n", valores_nulos)
```

```
Valores faltantes iniciales por columna:
 Country name                          0
year                                   0
Life Ladder                            0
Log GDP per capita                    28
Social support                        13
Healthy life expectancy at birth      63
Freedom to make life choices          36
Generosity                            81
Perceptions of corruption            125
Positive affect                       24
Negative affect                       16
Continent                              0
dtype: int64
```

Paso 4: Limpieza de datos

1. El proceso de limpieza de datos comenzará utilizando el criterio de eliminar aquellas filas (países) donde la cantidad de valores faltantes superen el 50%.

[10]:
```
# Eliminar filas con más del 50% de valores faltantes

param = df.shape[1] * 0.5

# Identificar las filas que tienen más del 50% de valores faltantes

filas_elim = df[df.isnull().sum(axis=1) > param]

# Mostrar las filas que tienen más del 50% de valores faltantes
print(filas_elim)
df_cleaned = df_merged.dropna(thresh=param)
```

```
              Country name  year  Life Ladder  Log GDP per capita  \
135                Bahrain  2014        6.165              10.802
1123                Kuwait  2014        6.180              10.926
1721                 Qatar  2015        6.375              11.532
2197  United Arab Emirates  2014        6.540              11.072
2307               Vietnam  2017        5.175               9.111

      Social support  Healthy life expectancy at birth  \
135              NaN                             65.96
```

```
1123             NaN                              70.00
1721             NaN                              66.40
2197             NaN                              65.34
2307             NaN                              65.10

      Freedom to make life choices  Generosity  Perceptions of corruption  \
135                            NaN         NaN                        NaN
1123                           NaN         NaN                        NaN
1721                           NaN         NaN                        NaN
2197                           NaN         NaN                        NaN
2307                           NaN         NaN                        NaN

      Positive affect  Negative affect
135               NaN              NaN
1123              NaN              NaN
1721              NaN              NaN
2197              NaN              NaN
2307              NaN              NaN
```

2. Para los valores faltantes que aún existen en el dataframe, serán reemplazados con la mediana respectiva. Para esto, se considerarán 2 criterios de imputación de datos; el primero nos indica que se reemplazarán los datos vacíos con la mediana del país respectivo. Por otro lado, el segundo nos indica que en caso de no encontrar datos de la columna respectiva y tampoco del país, se completará con la mediana del continente.

[11]:
```python
# Función para rellenar los valores faltantes por país y continente
def rellenar_faltantes(df, column):
    # Rellenar primero con la mediana por país
    df[column] = df.groupby('Country name')[column].transform(lambda x: x.
 ↪fillna(x.median()))

    # Rellenar los que aún estén faltantes con la mediana por continente
    df[column] = df.groupby('Continent')[column].transform(lambda x: x.fillna(x.
 ↪median()))

    return df

# Columnas consideradas para la  imputacion de datos
cols_a_rellenar = ['Log GDP per capita', 'Social support', 'Healthy life␣
 ↪expectancy at birth',
                   'Freedom to make life choices', 'Generosity', 'Perceptions␣
 ↪of corruption',
                   'Positive affect', 'Negative affect']

# Aplicar la función a todas las columnas numéricas con valores faltantes
for column in cols_a_rellenar:
    df_cleaned = rellenar_faltantes(df_cleaned, column)
```

3. A continuación, se verifica la existencia de valores duplicados y en el caso de existir, se procede a su eliminación.

[12]:
```
# Verificar y eliminar filas duplicadas
filas_duplicadas = df_cleaned.duplicated().sum()
print(f"Número de filas duplicadas: {filas_duplicadas}")

# Eliminar duplicados
df_cleaned = df_cleaned.drop_duplicates()
```

```
Número de filas duplicadas: 0
```

4. Se verifica la existencia de valores nulos luego de haber aplicado los pasos anteriores.

[13]:
```
# Revisar valores faltantes después de la limpieza
valores_nulos_post = df_cleaned.isnull().sum()
print("Valores faltantes después de la limpieza:\n", valores_nulos_post)
```

```
Valores faltantes después de la limpieza:
 Country name                        0
year                                0
Life Ladder                         0
Log GDP per capita                  0
Social support                      0
Healthy life expectancy at birth    0
Freedom to make life choices        0
Generosity                          0
Perceptions of corruption           0
Positive affect                     0
Negative affect                     0
Continent                           0
dtype: int64
```

Paso 5: Almacenamiento de dataframe limpio En este punto del análisis guardamos una copia limpia del dataset analizado. Debido a que se realizará un proceso de estandarización y normalización se guardarán copias de los datos sin dicho tipo de procesamiento, para poder utilizarlos sin problemas para elaborar los gráficos finales.

[14]:
```
# Guardar el dataset limpio
df_cleaned.to_csv("dataset_limpio.csv", index=False)
df_heatmap = df_cleaned.copy() # Este dataframe se utilizará más adelante para␣
↪la creación de visualizaciones
df_excel = df_cleaned.copy() # Este dataframe se utilizará más adelante para la␣
↪creación de visualizaciones
```

Paso 6: Carga de datos limpios A continuación, se carga el dataset limpio generado en la etapa `N° 1 de Exploración y Limpieza`.

[15]:
```python
# Cargar el dataset principal
ruta = "dataset_limpio.csv"
df_cleaned = pd.read_csv(ruta, encoding='latin1')
df_cleaned.head()
```

[15]:
```
  Country name  year  Life Ladder  Log GDP per capita  Social support  \
0  Afghanistan  2008        3.724               7.350           0.451
1  Afghanistan  2009        4.402               7.509           0.552
2  Afghanistan  2010        4.758               7.614           0.539
3  Afghanistan  2011        3.832               7.581           0.521
4  Afghanistan  2012        3.783               7.661           0.521

   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                              50.5                         0.718       0.164
1                              50.8                         0.679       0.187
2                              51.1                         0.600       0.118
3                              51.4                         0.496       0.160
4                              51.7                         0.531       0.234

   Perceptions of corruption  Positive affect  Negative affect Continent
0                      0.882            0.414            0.258      Asia
1                      0.850            0.481            0.237      Asia
2                      0.707            0.517            0.275      Asia
3                      0.731            0.480            0.267      Asia
4                      0.776            0.614            0.268      Asia
```

Paso 7: Aplicación de Feature Engineering En este paso se combinan distintas variables para identificar posibles interacciones entre ellas.

1. **GDP_social_interaction**:
   - **Descripción**: Representa la interacción entre el PIB per cápita (Log GDP per capita) y el soporte social. Mide cómo la combinación de la riqueza económica y el apoyo social puede influir en otros indicadores de bienestar.
2. **Life_health_interaction**:
   - **Descripción**: Muestra la interacción entre la expectativa de vida al nacer y el índice de satisfacción con la vida (Life Ladder). Esta variable combina la salud con el bienestar subjetivo para entender su relación.
3. **Support_GDP_ratio**:
   - **Descripción**: Proporción entre el soporte social y el PIB per cápita. Evalúa el nivel de soporte social en relación con la riqueza económica de cada país o región.
4. **Generosity_corruption_ratio**:
   - **Descripción**: Proporción entre la generosidad y la percepción de la corrupción. Permite analizar si los niveles de generosidad en un país están relacionados con la percepción de la corrupción.
5. **GDP_category**:
   - **Descripción**: Categorización del PIB per cápita en tres niveles: "Bajo", "Medio" y "Alto". Facilita la comparación entre países con diferentes niveles de riqueza.
6. **Life_Ladder_category**:

- **Descripción**: Categorización del índice de satisfacción con la vida (Life Ladder) en tres niveles: "Bajo", "Medio" y "Alto". Facilita el análisis de la satisfacción con la vida entre diferentes grupos.

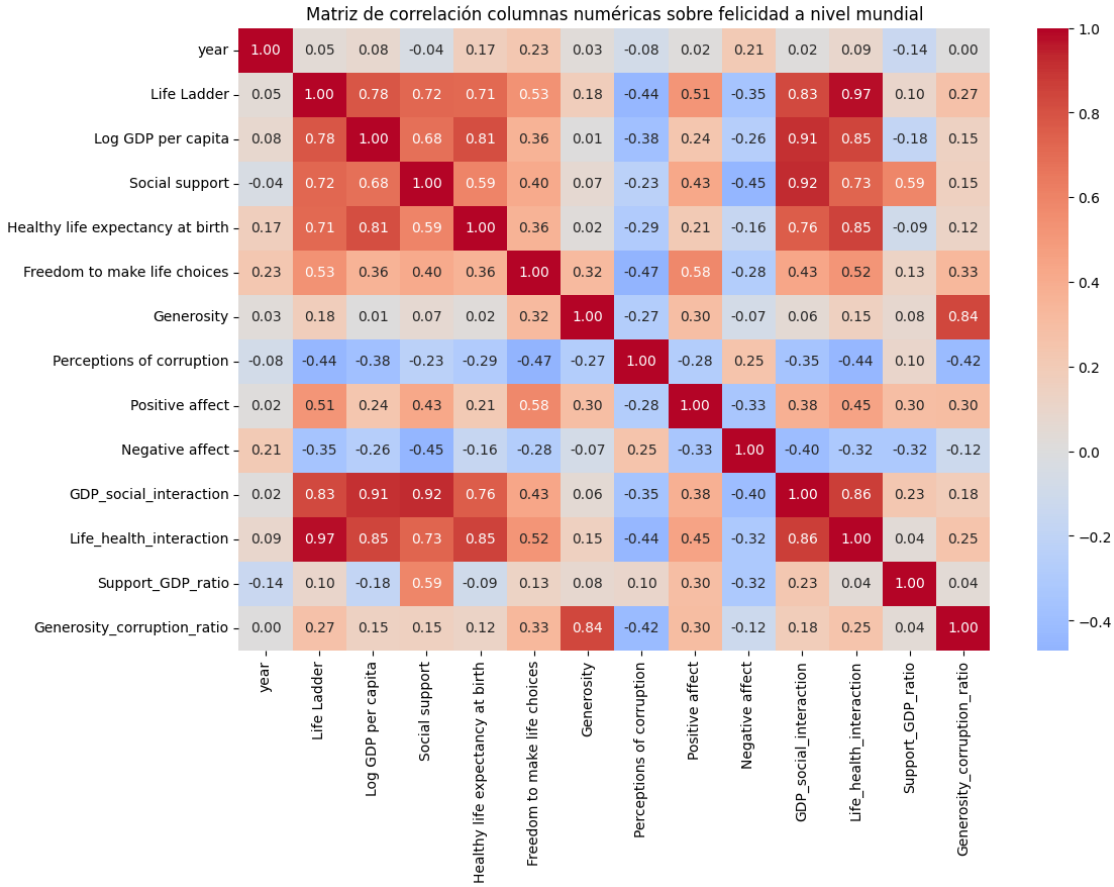
[16]:
```
# 1. Interacciones entre variables numéricas
# Crear la interacción entre "Log GDP per capita" y "Social support"
df_cleaned['GDP_social_interaction'] = df_cleaned['Log GDP per capita'] *␣
↪df_cleaned['Social support']

# Crear la interacción entre "Healthy life expectancy at birth" y "Life Ladder"
df_cleaned['Life_health_interaction'] = df_cleaned['Healthy life expectancy at␣
↪birth'] * df_cleaned['Life Ladder']

# 2. Ratios o proporciones
# Crear una columna de proporción entre "Social support" y "Log GDP per capita"
df_cleaned['Support_GDP_ratio'] = df_cleaned['Social support'] /␣
↪df_cleaned['Log GDP per capita']

# Crear una columna de proporción entre "Generosity" y "Perceptions of␣
↪corruption"
df_cleaned['Generosity_corruption_ratio'] = df_cleaned['Generosity'] /␣
↪df_cleaned['Perceptions of corruption']

# 3. Discretización o binning de variables
# Crear categorías para "Log GDP per capita" en niveles "Bajo", "Medio" y "Alto"
df_cleaned['GDP_category'] = pd.cut(df_cleaned['Log GDP per capita'],
                                    bins=[5, 8, 10, 12],
                                    labels=["Bajo", "Medio", "Alto"])

# Crear categorías para "Life Ladder" en niveles "Bajo", "Medio" y "Alto"
df_cleaned['Life_Ladder_category'] = pd.cut(df_cleaned['Life Ladder'],
                                            bins=[0, 4, 6, 10],
                                            labels=["Bajo", "Medio", "Alto"])

# Guardar el conjunto de datos con las nuevas columnas en un archivo CSV␣
↪(opcional)
df_cleaned.to_csv('dataset_con_nuevas_columnas.csv', index=False)

# Mostrar las primeras filas del dataset con las nuevas columnas
df_cleaned.head()
```

[16]:

	Country name	year	Life Ladder	Log GDP per capita	Social support	\
0	Afghanistan	2008	3.724	7.350	0.451	
1	Afghanistan	2009	4.402	7.509	0.552	
2	Afghanistan	2010	4.758	7.614	0.539	
3	Afghanistan	2011	3.832	7.581	0.521	
4	Afghanistan	2012	3.783	7.661	0.521	

```
   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                              50.5                         0.718       0.164
1                              50.8                         0.679       0.187
2                              51.1                         0.600       0.118
3                              51.4                         0.496       0.160
4                              51.7                         0.531       0.234

   Perceptions of corruption  Positive affect  Negative affect Continent  \
0                      0.882            0.414            0.258      Asia
1                      0.850            0.481            0.237      Asia
2                      0.707            0.517            0.275      Asia
3                      0.731            0.480            0.267      Asia
4                      0.776            0.614            0.268      Asia

   GDP_social_interaction  Life_health_interaction  Support_GDP_ratio  \
0               3.314850                 188.0620           0.061361
1               4.144968                 223.6216           0.073512
2               4.103946                 243.1338           0.070791
3               3.949701                 196.9648           0.068724
4               3.991381                 195.5811           0.068007

   Generosity_corruption_ratio GDP_category Life_Ladder_category
0                     0.185941         Bajo                 Bajo
1                     0.220000         Bajo                Medio
2                     0.166902         Bajo                Medio
3                     0.218878         Bajo                 Bajo
4                     0.301546         Bajo                 Bajo
```

[17]:
```python
# Se muestra la matriz de correlación de los valores numéricos del dataset
correlation_matrix = df_cleaned.corr(numeric_only=True)
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", center=0, fmt=".
 ↪2f")
plt.title("Matriz de correlación columnas numéricas sobre felicidad a nivel␣
 ↪mundial")
plt.show()
```



Las variables originales que presentan mayor correlación con Life Ladder son Log GDP per capita, Social support, Healthy life expectancy at birth.

En el caso de las variables combinadas, la mayor correlación con Life Ladder es con Life health interaction y GDP social interaction. El primer caso es esperable porque se trata de una combinación de Life Ladder con Healthy life expectancy at birth, y en el segundo caso se trata de la combinación de dos variables que ya tienen una correlación clara con Life Ladder.

Por lo tanto, no se utilizarán las variables combinadas en los gráficos finales debido a que la información aportada por ellas ya se puede obtener a partir de las variables originales.

Paso 8: Normalización de datos Se realiza el proceso de normalización de las columnas numéricas del dataframe, excluyendo la columna "year" mediante el método **Min-Max Scaling** de `sklearn`, que ajusta los valores al rango 0-1. La decisión de aplicar la normalización en este conjunto de datos se basa en las diferencias significativas en la escala y rango de las variables numéricas. Aglunas variables como el "Log GDP per capita" y "Life Ladder", tienen rangos muchos más amplios que otras, como "Generosity" o "Perceptions of corruption".

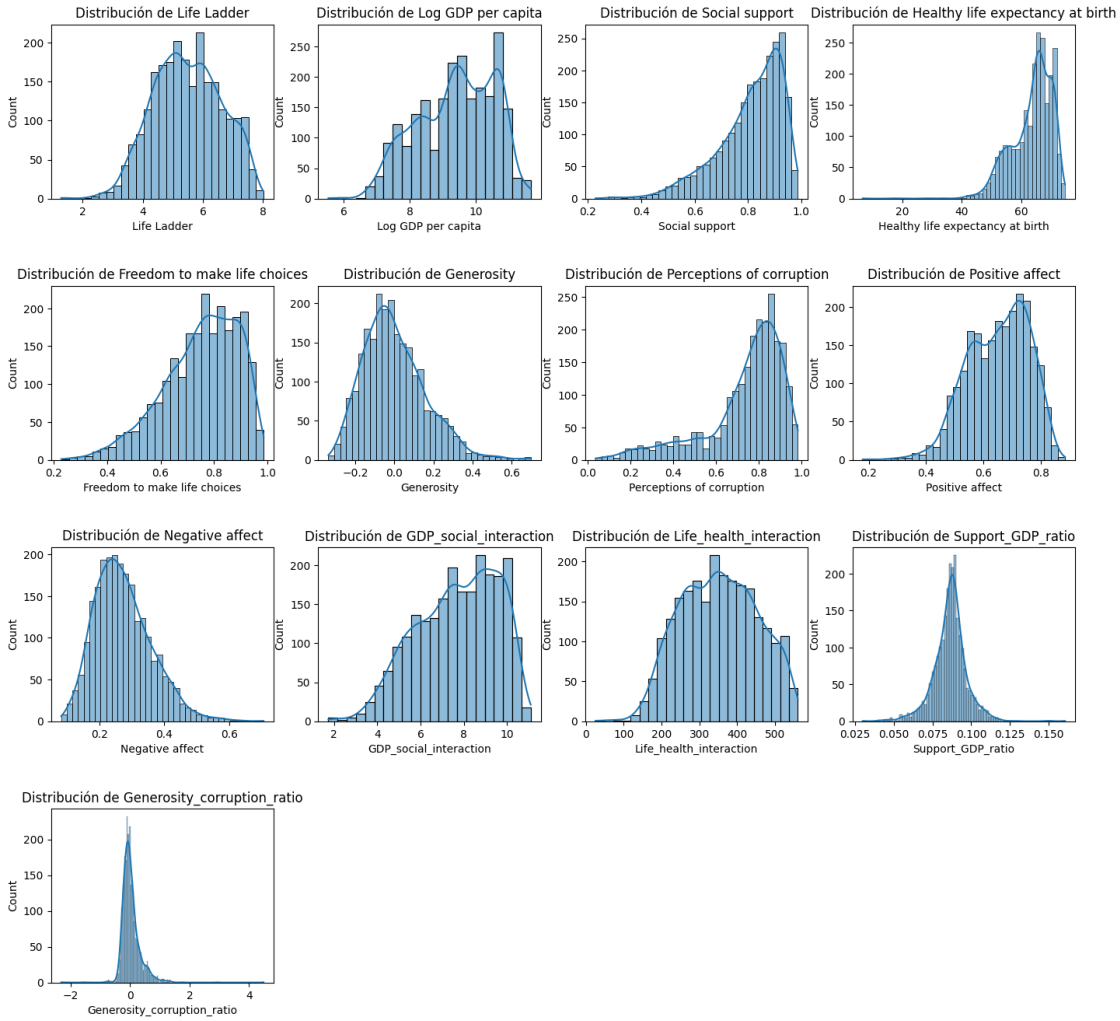
[18]:
```
# Inicializar el Min-Max Scaler
scaler = MinMaxScaler()
```

```python
# Excluir la columna "year" de las columnas numéricas a normalizar
columnas_normalizadas = df_cleaned.select_dtypes(include=['float64', 'int64']).
 ↪columns.drop('year')

# Aplicar la normalización solo a las columnas seleccionadas
normalizada_data = scaler.fit_transform(df_cleaned[columnas_normalizadas])

# Convertir los datos normalizados
normalizada_df = pd.DataFrame(normalizada_data, columns=columnas_normalizadas)

# Mantener las columnas no numéricas y la columna "year"
columnas_no_numericas = df_cleaned.select_dtypes(exclude=['float64', 'int64']).
 ↪columns

# Combinar el DataFrame de columnas no numéricas con el DataFrame normalizado
df_normalized = pd.concat([df_cleaned[columnas_no_numericas],␣
 ↪df_cleaned['year'], normalizada_df], axis=1)

# Guardar el conjunto de datos normalizado en un archivo CSV (opcional)
df_normalized.to_csv('dataset_normalizado.csv', index=False)

# Mostrar el dataset normalizado
print(df_normalized.head())
```

```
  Country name Continent GDP_category Life_Ladder_category  year  Life Ladder  \
0  Afghanistan      Asia         Bajo                 Bajo  2008     0.362570
1  Afghanistan      Asia         Bajo                Medio  2009     0.463194
2  Afghanistan      Asia         Bajo                Medio  2010     0.516028
3  Afghanistan      Asia         Bajo                 Bajo  2011     0.378599
4  Afghanistan      Asia         Bajo                 Bajo  2012     0.371327

   Log GDP per capita  Social support  Healthy life expectancy at birth  \
0            0.296471        0.293808                          0.644962
1            0.322329        0.426877                          0.649381
2            0.339405        0.409750                          0.653801
3            0.334038        0.386034                          0.658220
4            0.347048        0.386034                          0.662640

   Freedom to make life choices  Generosity  Perceptions of corruption  \
0                      0.647292    0.484615                   0.893460
1                      0.595773    0.506731                   0.859705
2                      0.491413    0.440385                   0.708861
3                      0.354029    0.480769                   0.734177
4                      0.400264    0.551923                   0.781646

   Positive affect  Negative affect  GDP_social_interaction  \
```

```
0         0.333333         0.281350                0.168299
1         0.428369         0.247588                0.257120
2         0.479433         0.308682                0.252731
3         0.426950         0.295820                0.236227
4         0.617021         0.297428                0.240686

   Life_health_interaction  Support_GDP_ratio  Generosity_corruption_ratio
0                 0.304090           0.240918                     0.367753
1                 0.370496           0.333801                     0.372762
2                 0.406935           0.313001                     0.364952
3                 0.320715           0.297207                     0.372597
4                 0.318131           0.291722                     0.384757
```

[19]:
```
### SE MUESTRA DISTRIBUCIÓN DE VARIABLES ANTES DE NORMALIZACIÓN ####

# Seleccionar todas las columnas numéricas, excluyendo "year"
cols_num = df_cleaned.select_dtypes(include=['float64', 'int64']).columns.
 ↪drop('year')

# Establecer el número de filas y columnas (7 filas, 2 columnas)
nrows = 4
ncols = 4

# Crear la figura con 4 filas y 4 columnas
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=(17, 4 * nrows))
fig.subplots_adjust(hspace=0.5)  # Ajustar el espacio entre gráficos

# Recorrer las variables numéricas y graficar cada histograma
for i, col in enumerate(cols_num):
    row = i // ncols  # Índice de la fila
    col_idx = i % ncols  # Índice de la columna
    sns.histplot(df_cleaned[col], kde=True, ax=ax[row, col_idx])  # Histogramas␣
 ↪con curva de densidad (KDE)
    ax[row, col_idx].set_title(f'Distribución de {col}')

# Eliminar cualquier gráfico vacío si hay menos de 14 variables
if len(cols_num) < nrows * ncols:
    for i in range(len(cols_num), nrows * ncols):
        fig.delaxes(ax.flatten()[i])

# Mostrar el gráfico
plt.show()
```



En estos gráficos se observa la distribución de los valores sin normalizar. Las variables que presentan distribuciones asimétricas son Social support, Healthy life expectancy at birth, Freedom to make life choices, Perceptions of corruption.

Life ladder tiene una distribución con dos valores centrales, por lo tanto, se podría deducir que en este caso existe una mezcla de distribuciones gaussianas.

Log GDP per cápita también podría considerarse como una mezcla de distribuciones gaussianas.

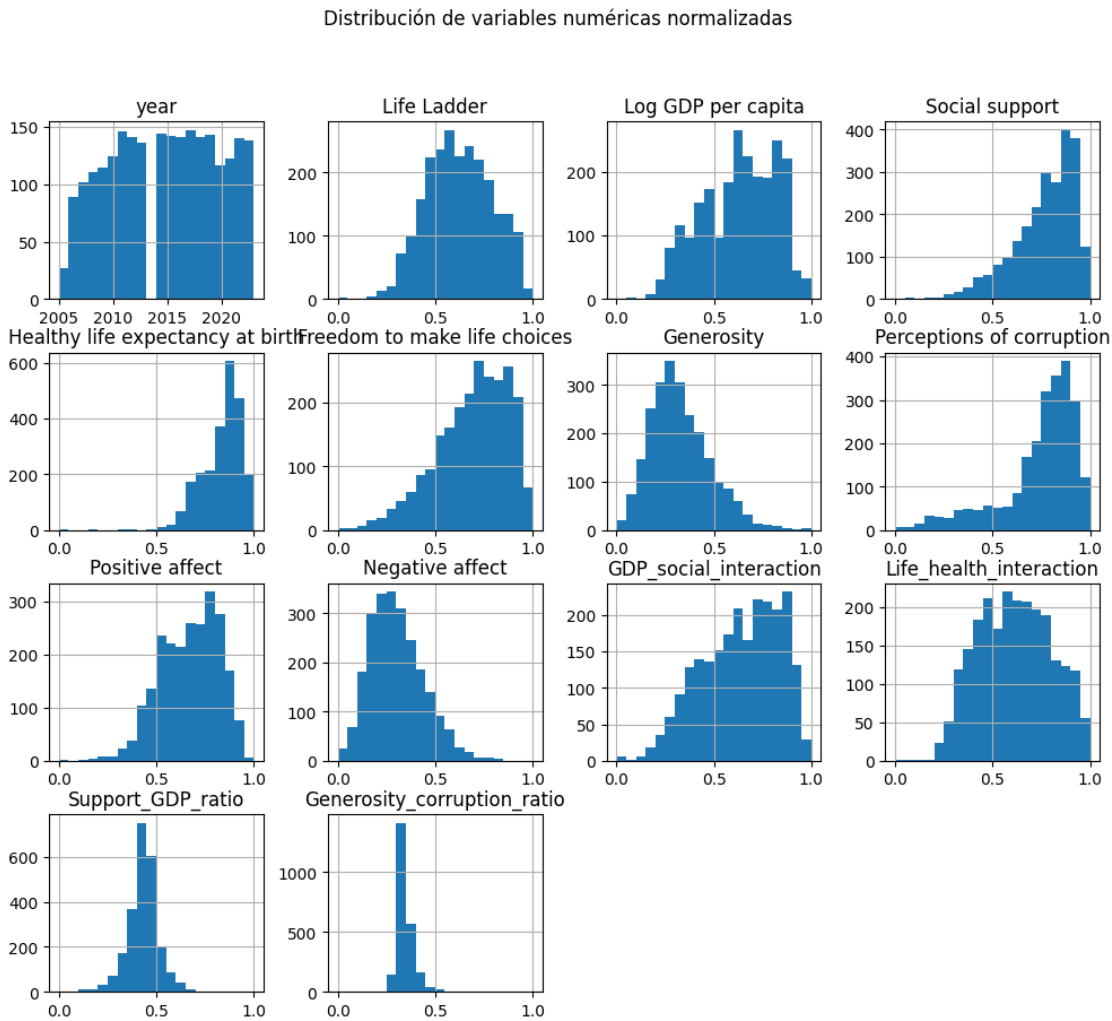
[20]:
```
### SE MUESTRA UNA SERIE DE GRÁFICOS PARA VISUALIZAR EL COMPORTAMIENTO DE LAS␣
↪VARIABLES ####

df = df_normalized

# 1. Estadísticas descriptivas
print(df.describe())
```

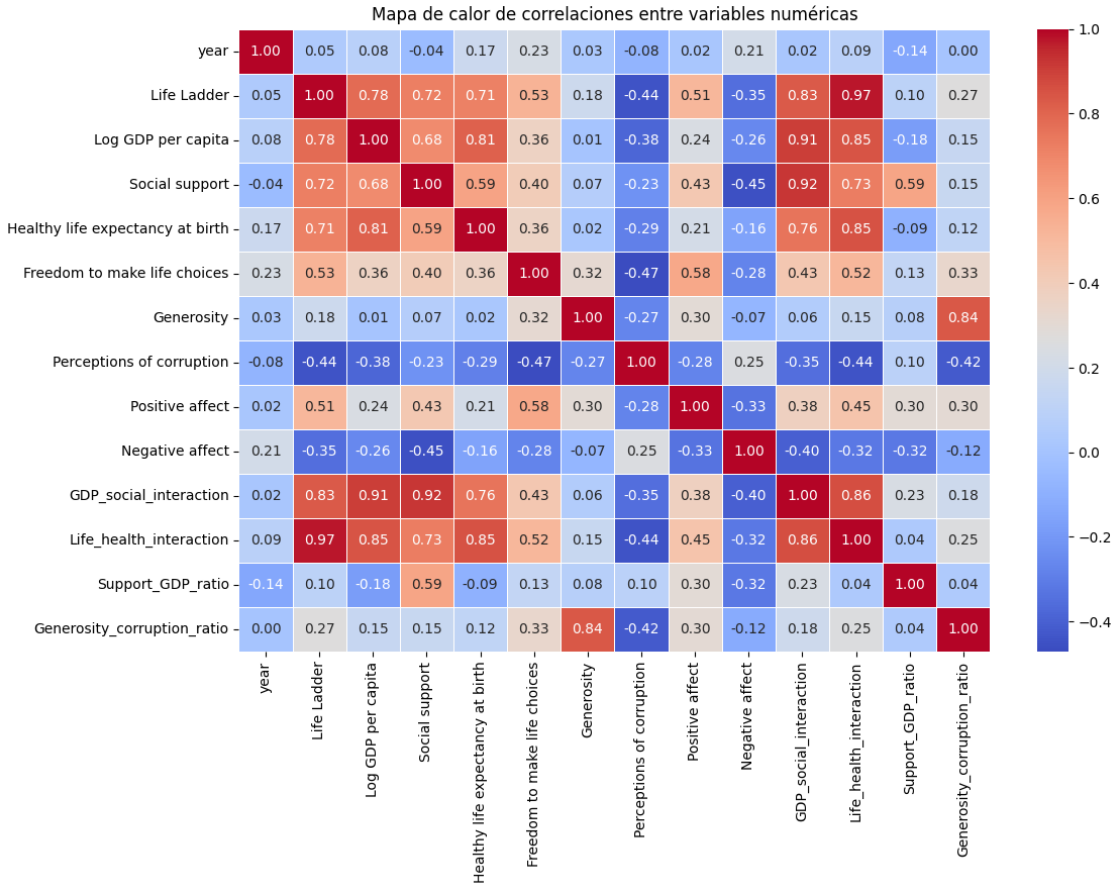
```python
# 2. Histograma de las variables numéricas
df.hist(bins=20, figsize=(12, 10))
plt.suptitle('Distribución de variables numéricas normalizadas', size=12)
plt.show()

# 3. Mapa de calor para visualizar las correlaciones entre las variables␣
↪numéricas
plt.figure(figsize=(12, 8))
correlation_matrix = df.select_dtypes(include=['float64', 'int64']).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
↪fmt=".2f")
plt.title('Mapa de calor de correlaciones entre variables numéricas')
plt.show()

# 4. Análisis temporal: Cambios de "Life Ladder" a lo largo del tiempo
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x='year', y='Life Ladder')
plt.title('Tendencia de Life Ladder a lo largo del tiempo')
plt.show()

# 5. Análisis temporal: Cambios de "Log GDP per capita" a lo largo del tiempo
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x='year', y='Log GDP per capita')

plt.title('Tendencia del Log GDP per capita a lo largo del tiempo')
plt.show()

# 6. Gráfico de barras para el promedio de Life Ladder por continente
plt.figure(figsize=(10, 6))
sns.barplot(x='Continent', y='Life Ladder', data=df, estimator='mean', ci=None)
plt.title('Promedio de Life Ladder por Continente')
plt.show()

# 7. Gráfico de densidad de Log GDP per capita por continente
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='Log GDP per capita', hue='Continent', fill=True)
plt.title('Distribución de Log GDP per capita por Continente')
plt.show()

# 8. Gráfico de violín para Log GDP per capita por continente
plt.figure(figsize=(10, 6))
sns.violinplot(x='Continent', y='Log GDP per capita', data=df)
plt.title('Distribución de Log GDP per capita por Continente')
plt.show()

# 9. Gráfico de dispersión categórico (strip plot) para Life Ladder por␣
↪continente
```

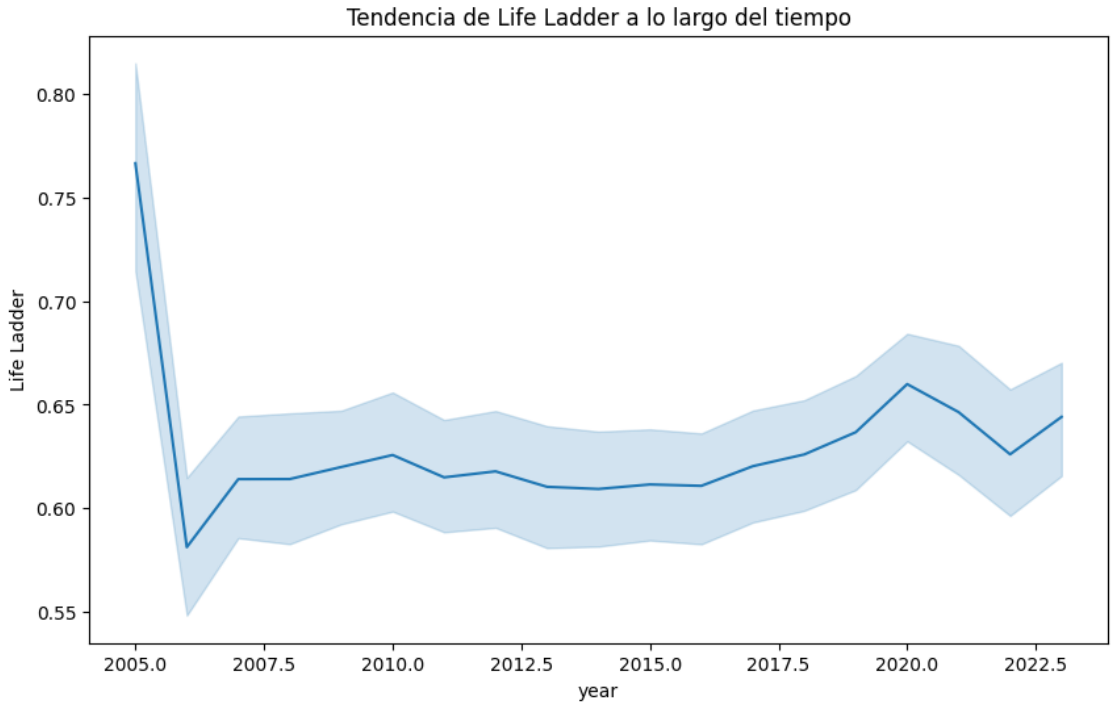
```python
plt.figure(figsize=(10, 6))
sns.stripplot(x='Continent', y='Life Ladder', data=df, jitter=True)
plt.title('Distribución de Life Ladder por Continente')
plt.show()
```

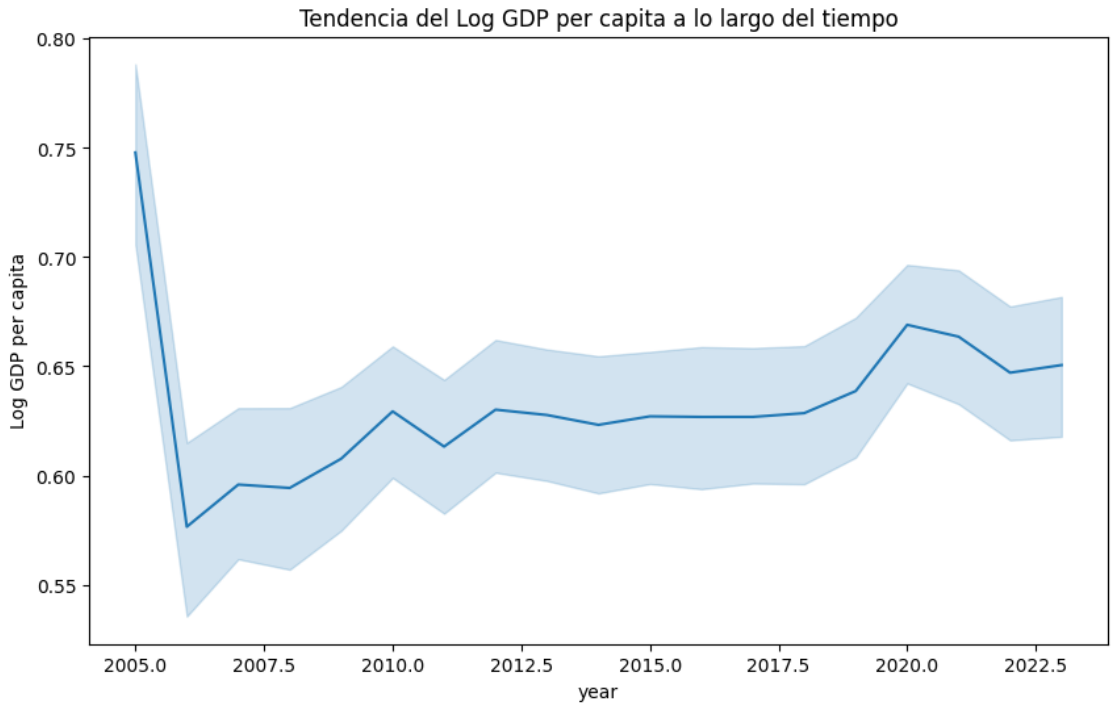
```
              year  Life Ladder  Log GDP per capita  Social support  \
count  2363.000000  2363.000000         2363.000000     2363.000000
mean   2014.763860     0.623711            0.629233        0.766016
std       5.059436     0.167041            0.187451        0.159535
min    2005.000000     0.000000            0.000000        0.000000
25%    2011.000000     0.499555            0.483900        0.679842
50%    2015.000000     0.618581            0.645796        0.799736
75%    2019.000000     0.748367            0.791267        0.890646
max    2023.000000     1.000000            1.000000        1.000000

       Healthy life expectancy at birth  Freedom to make life choices  \
count                       2363.000000                   2363.000000
mean                           0.836026                      0.690885
std                            0.100035                      0.183713
min                            0.000000                      0.000000
25%                            0.777254                      0.571995
50%                            0.860047                      0.718626
75%                            0.913229                      0.837517
max                            1.000000                      1.000000

        Generosity  Perceptions of corruption  Positive affect  \
count  2363.000000                 2363.000000      2363.000000
mean      0.326756                    0.740696         0.670168
std       0.154219                    0.197455         0.150433
min       0.000000                    0.000000         0.000000
25%       0.219231                    0.676160         0.557447
50%       0.305769                    0.797468         0.686525
75%       0.416346                    0.874473         0.790780
max       1.000000                    1.000000         1.000000

       Negative affect  GDP_social_interaction  Life_health_interaction  \
count      2363.000000             2363.000000               2363.000000
mean          0.305892                0.637560                  0.613013
std           0.139787                0.197820                  0.186948
min           0.000000                0.000000                  0.000000
25%           0.202572                0.489135                  0.464437
50%           0.289389                0.655147                  0.610672
75%           0.390675                0.803237                  0.756083
max           1.000000                1.000000                  1.000000

       Support_GDP_ratio  Generosity_corruption_ratio
count        2363.000000                  2363.000000
mean            0.432165                     0.345180
```

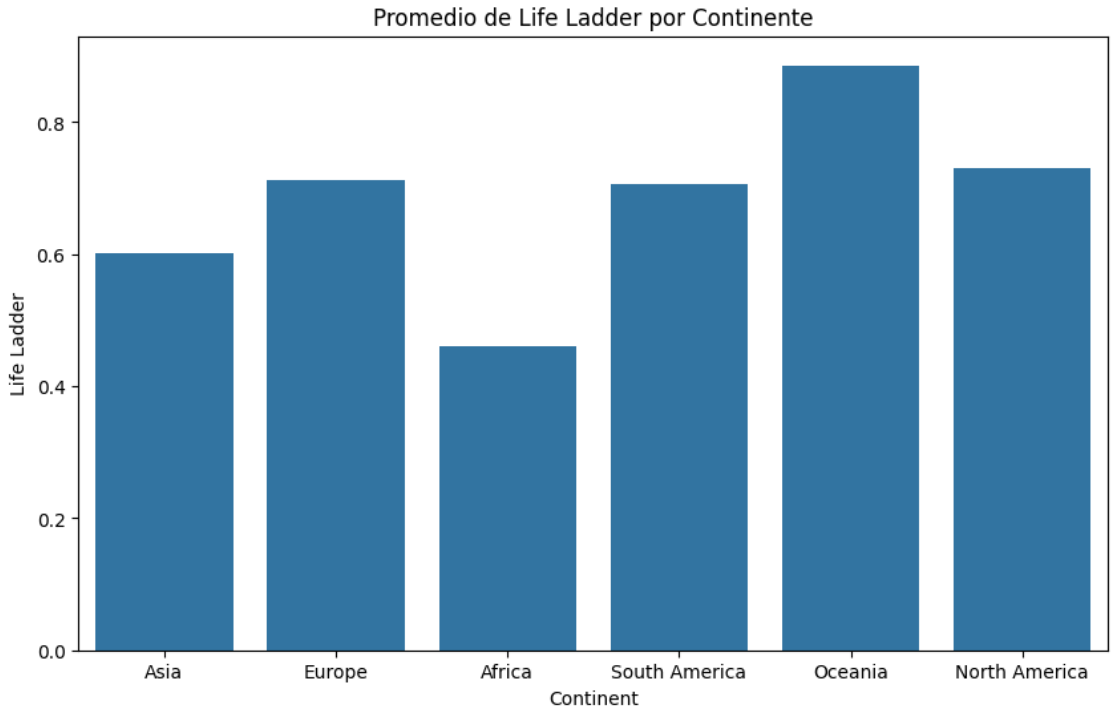
std	0.081778	0.045053
min	0.000000	0.000000
25%	0.391572	0.317950
50%	0.437042	0.336281
75%	0.472832	0.359484
max	1.000000	1.000000

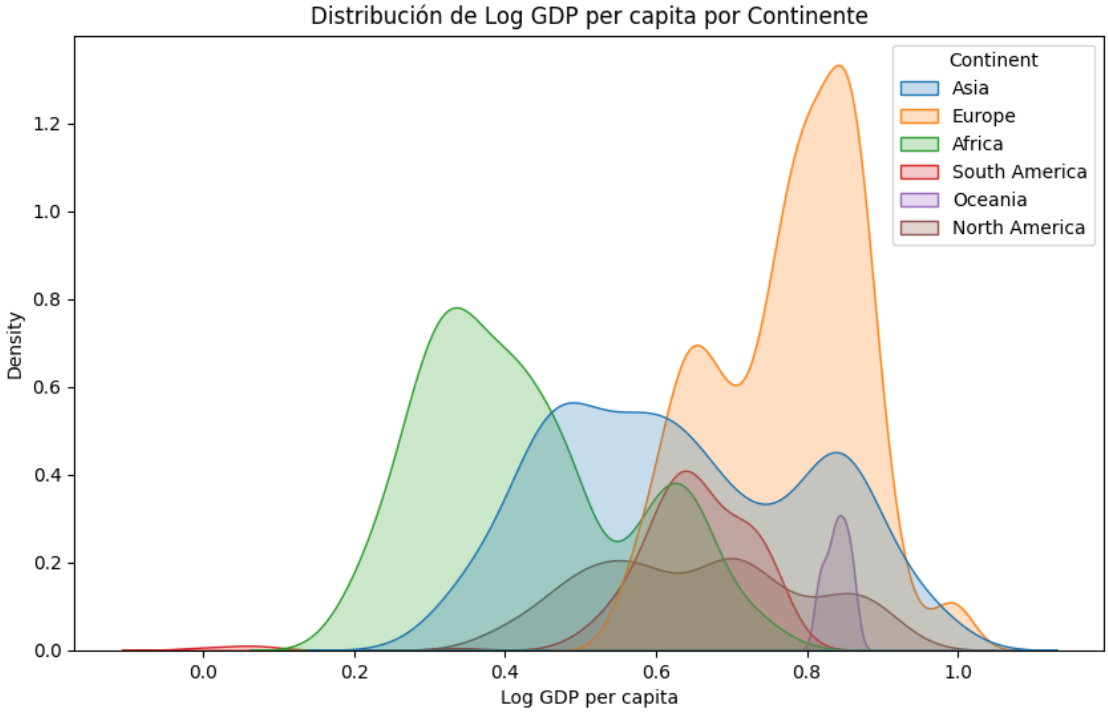


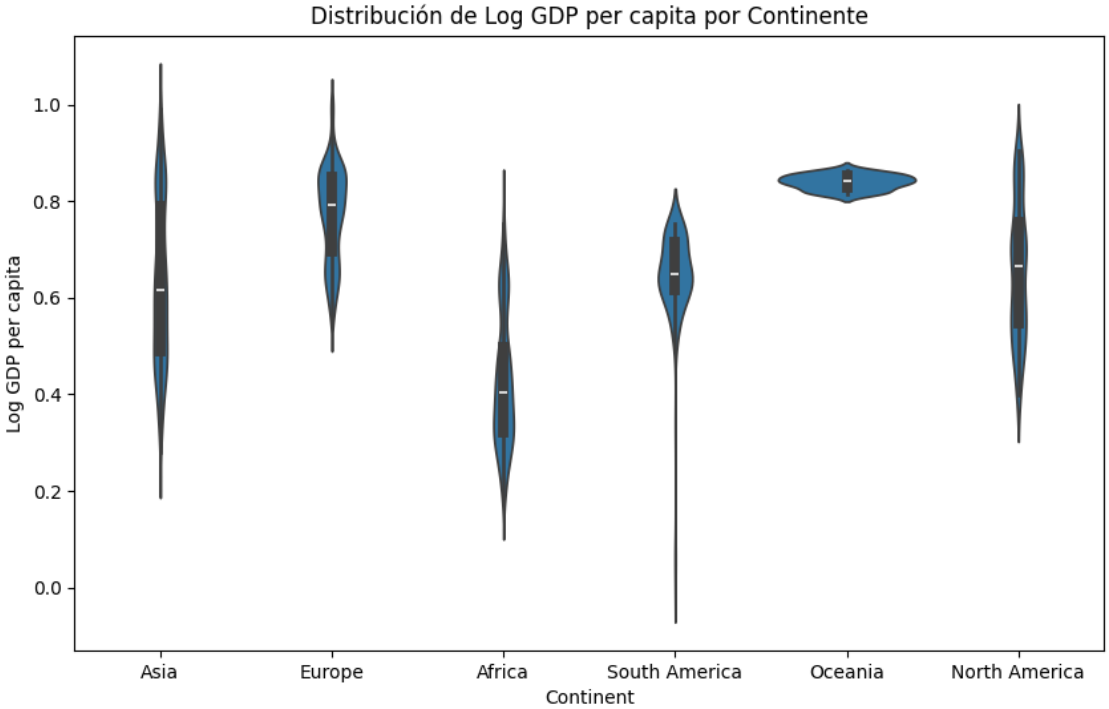

Mapa de calor de correlaciones entre variables numéricas
year 1.00 0.05 0.08 -0.04 0.17 0.23 0.03 -0.08 0.02 0.21 0.02 0.09 -0.14 0.00
Life Ladder 0.05 1.00 0.78 0.72 0.71 0.53 0.18 -0.44 0.51 -0.35 0.83 0.97 0.10 0.27
Log GDP per capita 0.08 0.78 1.00 0.68 0.81 0.36 0.01 -0.38 0.24 -0.26 0.91 0.85 -0.18 0.15
Social support -0.04 0.72 0.68 1.00 0.59 0.40 0.07 -0.23 0.43 -0.45 0.92 0.73 0.59 0.15
Healthy life expectancy at birth 0.17 0.71 0.81 0.59 1.00 0.36 0.02 -0.29 0.21 -0.16 0.76 0.85 -0.09 0.12
Freedom to make life choices 0.23 0.53 0.36 0.40 0.36 1.00 0.32 -0.47 0.58 -0.28 0.43 0.52 0.13 0.33
Generosity 0.03 0.18 0.01 0.07 0.02 0.32 1.00 -0.27 0.30 -0.07 0.06 0.15 0.08 0.84
Perceptions of corruption -0.08 -0.44 -0.38 -0.23 -0.29 -0.47 -0.27 1.00 -0.28 0.25 -0.35 -0.44 0.10 -0.42
Positive affect 0.02 0.51 0.24 0.43 0.21 0.58 0.30 -0.28 1.00 -0.33 0.38 0.45 0.30 0.30
Negative affect 0.21 -0.35 -0.26 -0.45 -0.16 -0.28 -0.07 0.25 -0.33 1.00 -0.40 -0.32 -0.32 -0.12
GDP_social_interaction 0.02 0.83 0.91 0.92 0.76 0.43 0.06 -0.35 0.38 -0.40 1.00 0.86 0.23 0.18
Life_health_interaction 0.09 0.97 0.85 0.73 0.85 0.52 0.15 -0.44 0.45 -0.32 0.86 1.00 0.04 0.25
Support_GDP_ratio -0.14 0.10 -0.18 0.59 -0.09 0.13 0.08 0.10 0.30 -0.32 0.23 0.04 1.00 0.04
Generosity_corruption_ratio 0.00 0.27 0.15 0.15 0.12 0.33 0.84 -0.42 0.30 -0.12 0.18 0.25 0.04 1.00
year
Life Ladder
Log GDP per capita
Social support
Healthy life expectancy at birth
Freedom to make life choices
Generosity
Perceptions of corruption
Positive affect
Negative affect
GDP_social_interaction
Life_health_interaction
Support_GDP_ratio
Generosity_corruption_ratio
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

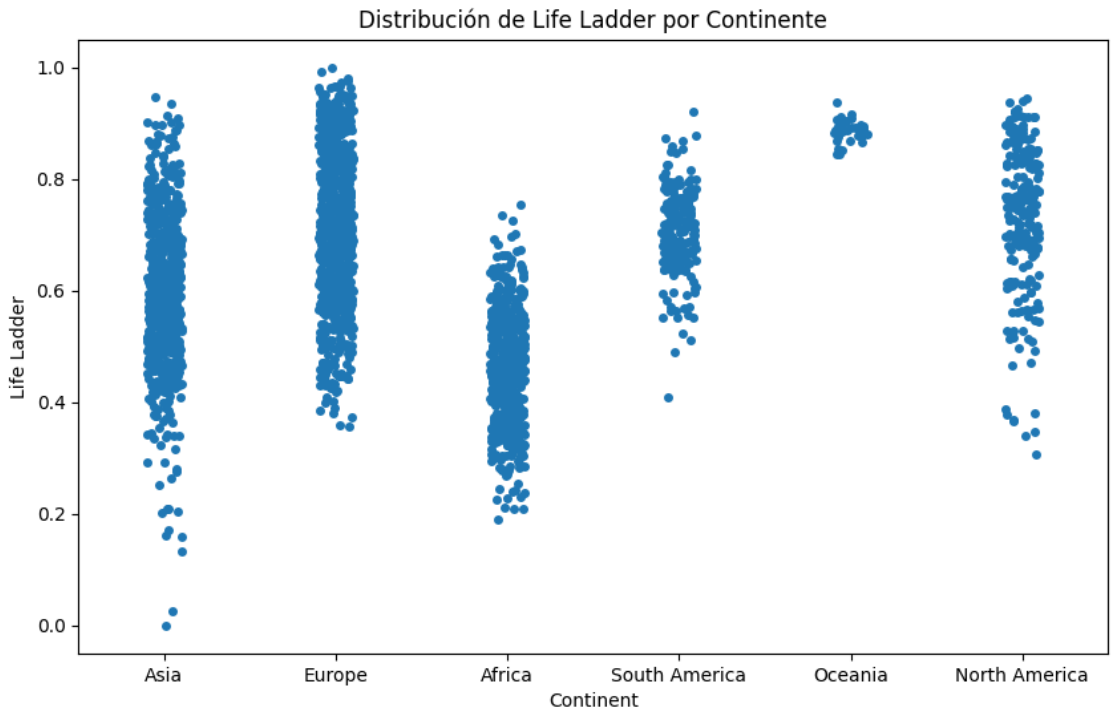

Tendencia de Life Ladder a lo largo del tiempo
Life Ladder
0.80
0.75
0.70
0.65
0.60
0.55
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
year


Tendencia del Log GDP per capita a lo largo del tiempo
Log GDP per capita
0.80
0.75
0.70
0.65
0.60
0.55
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
year


Promedio de Life Ladder por Continente
Life Ladder
0.8
0.6
0.4
0.2
0.0
Asia
Europe
Africa
South America
Oceania
North America
Continent


Distribución de Log GDP per capita por Continente
Continent
Asia
Europe
Africa
South America
Oceania
North America
Density
Log GDP per capita
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Distribución de Log GDP per capita por Continente
Log GDP per capita
Continent
Asia
Europe
Africa
South America
Oceania
North America
0.0
0.2
0.4
0.6
0.8
1.0



La distribución de las variables no cambia al normalizar los datos.

Tampoco se observa un cambio en las correlaciones de Life Ladder con las otras variables del dataset.

Los valores normalizados de Life Ladder muestran el valor promedio mundial normalizado, el cual alcanzó un valor mínimo entre los años 2005 y 2007. Después sus valores han fluctuado en un rango entre 0.60 y 0.65. La banda azul corresponde al intervalo de confianza para los valores de Life Ladder.

Los valores observados de Log GDP per capita muestran un patrón similar al de Life Ladder, lo cual concuerda con el valor de correlación mostrado en la matriz de correlación.

Respecto al valor promedio de Life Ladder, Oceanía presenta el mayor valor. El valor menor se observa en África. El valor de Life Ladder promedio es mayor al valor de Asia. Europa y Norteamérica presentan valores promedio similares.

Al comparar las distribuciones de Log GDP per cápita por continente se confirma que todas son diferentes, a pesar de solaparse entre sí. La distribución de menor ancho corresponde a Oceanía. En general las distribuciones no presentan datos que puedan ser considerados como sorpresivos, debido a que los valores más bajos de Log GDP se concentran en África, aumentando en Asia, Norteamérica, Sudamérica, Europa y Oceanía. El gráfico de violín permite visualizar los datos de las distribuciones considerando los cuartiles y la mediana de Log GDP per cápita por continente.

Finalmente, en el gráfico de distribución de Life Ladder por continente se verifica que los valores presentan poca dispersión en Oceanía. Asia presenta el mayor rango de valores de Life Ladder.

El punto con el menor valor de Sudamérica probablemente corresponde a Venezuela.

Paso 9: Estandarización de datos Se implementará el proceso de Estandarización dado que esto será útil cuando se necesite que los datos estén centrados en la media y con una desviación estándar uniforme. Generalmente, se utiliza en modelos estadísticos y algoritmos que asumen que los datos tienen una distribución aproximadamente normal, como regresiones lineales, análisis de componentes pricipales (PCA), y algunos modelos de máquinas de soporte vectorial (SVM).

[21]:
```
# Seleccionar las columnas numéricas excluyendo "year"
cols_num = df_cleaned.select_dtypes(include=['float64', 'int64']).columns.
 ↪drop('year')

# Inicializar el StandardScaler
scaler = StandardScaler()

# Aplicar la estandarización a las columnas numéricas
df_standarized = df_cleaned.copy()  # Hacer una copia del DataFrame
df_standarized[cols_num] = scaler.fit_transform(df_cleaned[cols_num])

# Guardar el dataset estandarizado en un nuevo archivo CSV
df_standarized.to_csv('dataset_estandarizado.csv', index=False)

# Mostrar las primeras filas del dataset estandarizado
print(df_standarized.head())
```

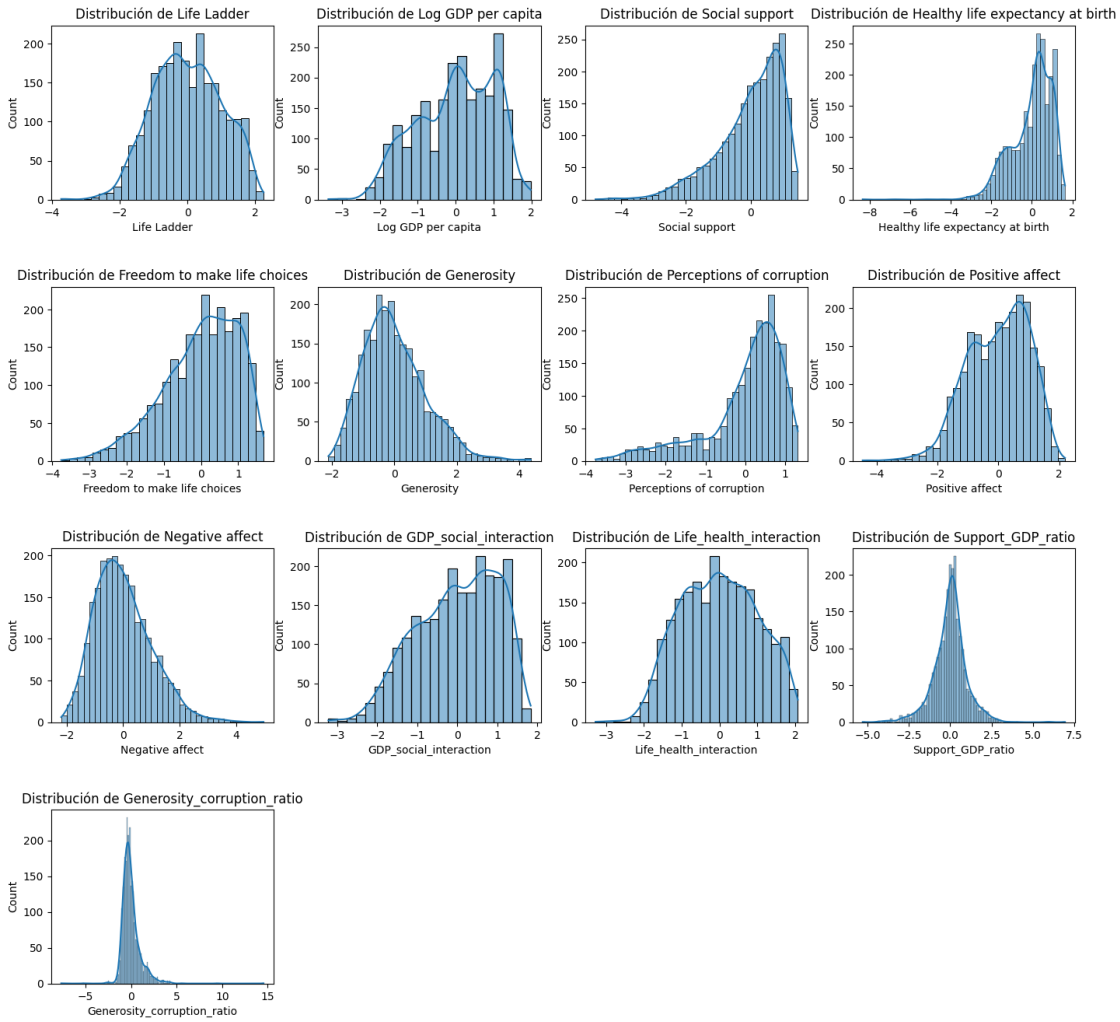
```
  Country name  year  Life Ladder  Log GDP per capita  Social support  \
0  Afghanistan  2008    -1.563665           -1.775565       -2.960521
1  Afghanistan  2009    -0.961150           -1.637591       -2.126237
2  Afghanistan  2010    -0.644785           -1.546477       -2.233620
3  Afghanistan  2011    -1.467689           -1.575113       -2.382304
4  Afghanistan  2012    -1.511233           -1.505692       -2.382304

   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                         -1.910373                     -0.237337    1.023819
1                         -1.866183                     -0.517830    1.167252
2                         -1.821994                     -1.086007    0.736954
3                         -1.777805                     -1.833988    0.998875
4                         -1.733615                     -1.582264    1.460353

   Perceptions of corruption  Positive affect  Negative affect Continent  \
0                   0.773828        -2.239582        -0.175599      Asia
1                   0.602840        -1.607700        -0.417175      Asia
2                  -0.161264        -1.268182         0.019962      Asia
3                  -0.033022        -1.617131        -0.072067      Asia
4                   0.207430        -0.353368        -0.060563      Asia

   GDP_social_interaction  Life_health_interaction  Support_GDP_ratio  \
0               -2.372655                -1.652810          -2.339106
1               -1.923565                -1.297520          -1.203071
2               -1.945757                -1.102566          -1.457473
```

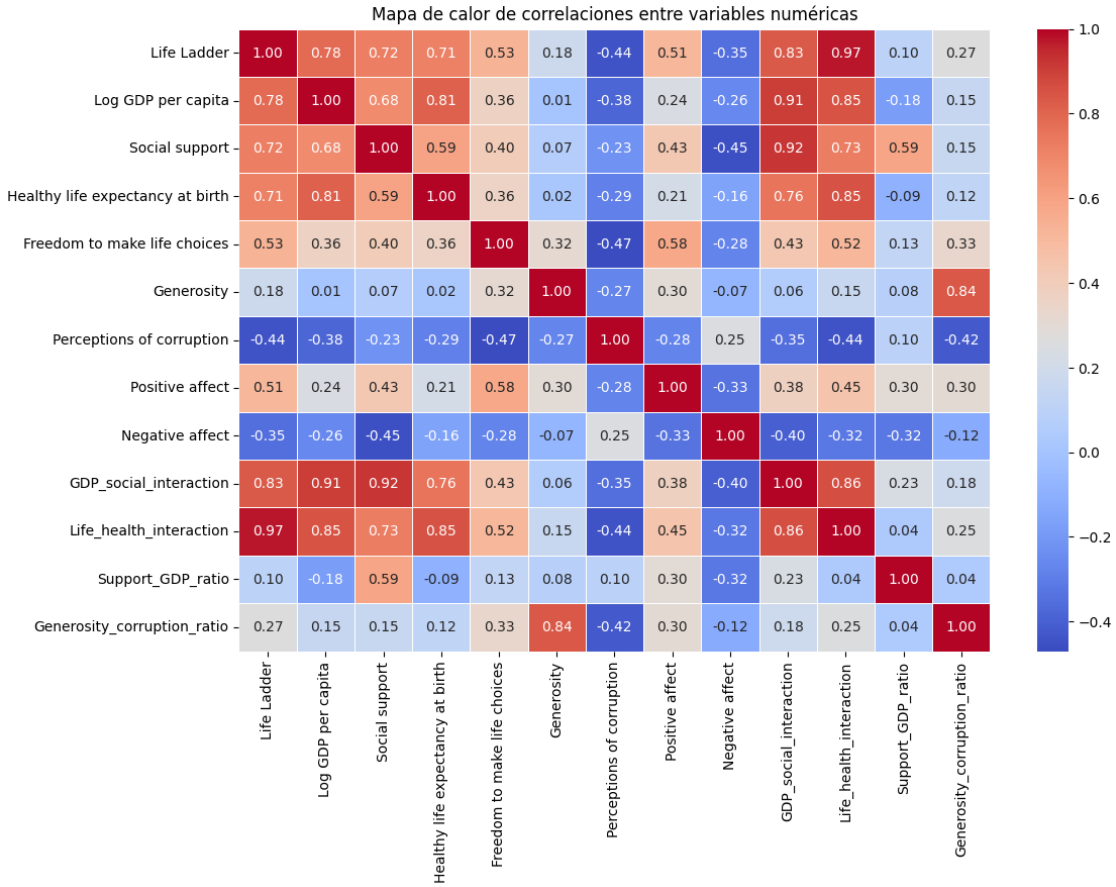
```
3                -2.029203                -1.563859          -1.650646
4                -2.006655                -1.577684          -1.717740

   Generosity_corruption_ratio GDP_category Life_Ladder_category
0                     0.501147         Bajo                 Bajo
1                     0.612366         Bajo                Medio
2                     0.438976         Bajo                Medio
3                     0.608703         Bajo                 Bajo
4                     0.878655         Bajo                 Bajo
```

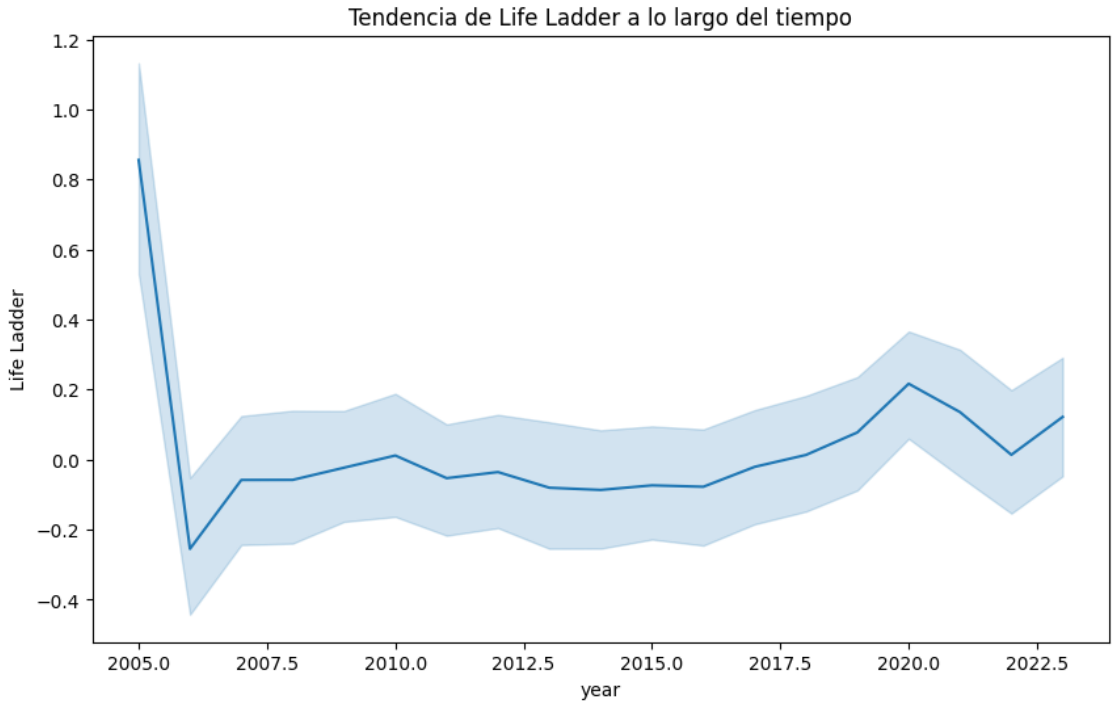
[22]:
```
#### SE MUESTRA UNA SERIE DE GRÁFICOS PARA CONOCER EL COMPORTAMIENTO DE LAS␣
↪VARIABLES ESTANDARIZADAS ###

# Cargar el dataset estandarizado
df_cleaned = df_standarized

# Seleccionar las columnas numéricas excluyendo "year"
cols_num = df_cleaned.select_dtypes(include=['float64', 'int64']).columns.
↪drop('year')

# 1. Mostrar distribución de los datos estandarizados
# Establecer el número de filas y columnas (7 filas, 2 columnas)
nrows = 4
ncols = 4

# Crear la figura con 4 filas y 4 columnas
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=(17, 4 * nrows))
fig.subplots_adjust(hspace=0.5)  # Ajustar el espacio entre gráficos

# Recorrer las variables numéricas y graficar cada histograma
for i, col in enumerate(cols_num):
    row = i // ncols  # Índice de la fila
    col_idx = i % ncols  # Índice de la columna
    sns.histplot(df_cleaned[col], kde=True, ax=ax[row, col_idx])  # Histogramas␣
↪con curva de densidad (KDE)
    ax[row, col_idx].set_title(f'Distribución de {col}')

# Eliminar cualquier gráfico vacío si hay menos de 14 variables
if len(cols_num) < nrows * ncols:
    for i in range(len(cols_num), nrows * ncols):
        fig.delaxes(ax.flatten()[i])

# Mostrar el gráfico
plt.show()
```

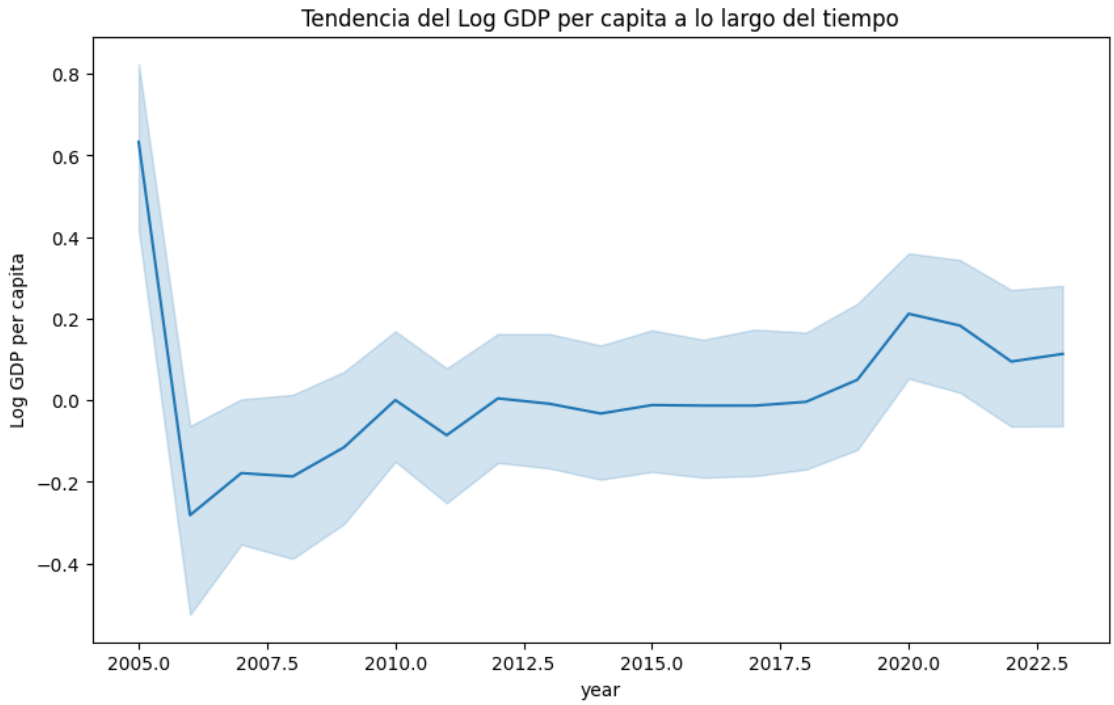
```python
# 3. Mapa de calor para visualizar las correlaciones entre las variables␣
↪numéricas
plt.figure(figsize=(12, 8))
correlation_matrix = df_cleaned[cols_num].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
↪fmt=".2f")
plt.title('Mapa de calor de correlaciones entre variables numéricas')
plt.show()

# 4. Análisis temporal: Cambios de "Life Ladder" a lo largo del tiempo
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_cleaned, x='year', y='Life Ladder')
plt.title('Tendencia de Life Ladder a lo largo del tiempo')
plt.show()

# 5. Análisis temporal: Cambios de "Log GDP per capita" a lo largo del tiempo
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_cleaned, x='year', y='Log GDP per capita')
plt.title('Tendencia del Log GDP per capita a lo largo del tiempo')
plt.show()

# 6. Gráfico de barras para Generosity por continente
plt.figure(figsize=(10, 6))
sns.barplot(x='Continent', y='Generosity', data=df_cleaned, estimator='mean',␣
↪ci=None)
plt.title('Promedio de Generosity por Continente')
plt.show()

# 7. Gráfico de densidad (KDE Plot) para Log GDP per capita por continente
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df_cleaned, x='Log GDP per capita', hue='Continent', fill=True)
plt.title('Distribución de Log GDP per capita por Continente')
plt.show()

# 8. Gráfico de barras apilado de Life Ladder y Positive affect por continente
plt.figure(figsize=(10, 6))
sns.barplot(x='Continent', y='Life Ladder', hue='Positive affect',␣
↪data=df_cleaned, ci=None)
plt.title('Comparación de Life Ladder y Positive affect por Continente')
plt.show()

# 9. Gráfico de densidad para Positive affect por continente
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df_cleaned, x='Positive affect', hue='Continent', fill=True)
plt.title('Distribución de Positive affect por Continente')
plt.show()
```

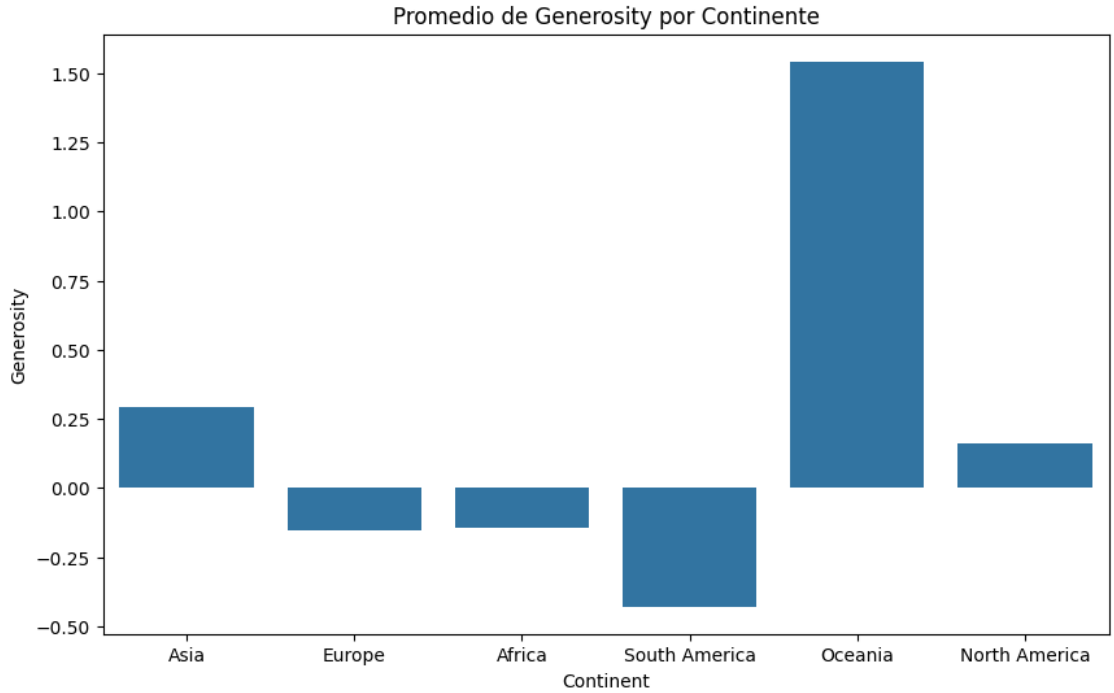
```python
# 10. Gráfico de línea: Tendencia de Social Support a lo largo del tiempo
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_cleaned, x='year', y='Social support')
plt.title('Tendencia de Social support a lo largo del tiempo')
plt.show()
```

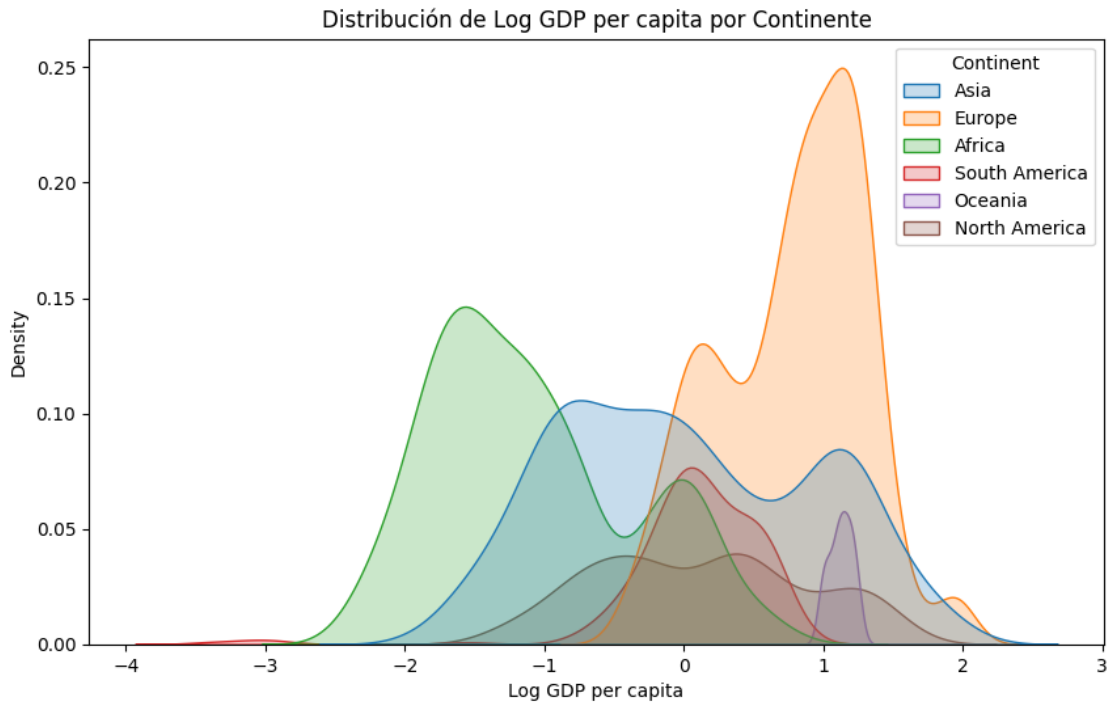


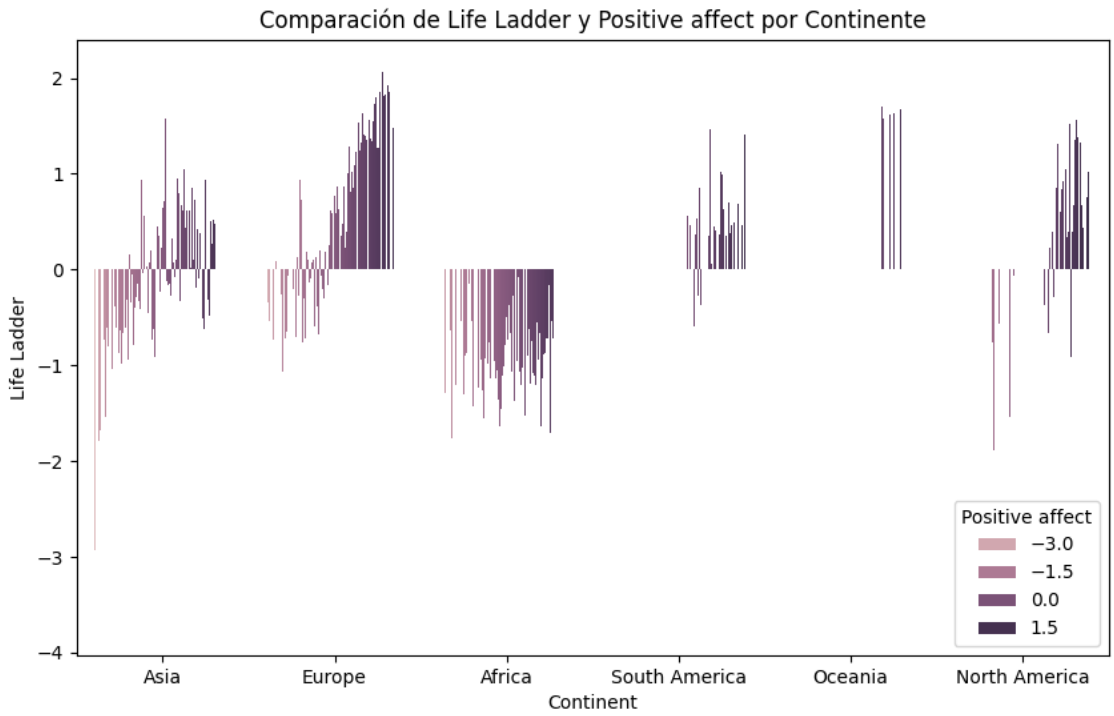

Mapa de calor de correlaciones entre variables numéricas
Life Ladder 1.00 0.78 0.72 0.71 0.53 0.18 -0.44 0.51 -0.35 0.83 0.97 0.10 0.27
Log GDP per capita 0.78 1.00 0.68 0.81 0.36 0.01 -0.38 0.24 -0.26 0.91 0.85 -0.18 0.15
Social support 0.72 0.68 1.00 0.59 0.40 0.07 -0.23 0.43 -0.45 0.92 0.73 0.59 0.15
Healthy life expectancy at birth 0.71 0.81 0.59 1.00 0.36 0.02 -0.29 0.21 -0.16 0.76 0.85 -0.09 0.12
Freedom to make life choices 0.53 0.36 0.40 0.36 1.00 0.32 -0.47 0.58 -0.28 0.43 0.52 0.13 0.33
Generosity 0.18 0.01 0.07 0.02 0.32 1.00 -0.27 0.30 -0.07 0.06 0.15 0.08 0.84
Perceptions of corruption -0.44 -0.38 -0.23 -0.29 -0.47 -0.27 1.00 -0.28 0.25 -0.35 -0.44 0.10 -0.42
Positive affect 0.51 0.24 0.43 0.21 0.58 0.30 -0.28 1.00 -0.33 0.38 0.45 0.30 0.30
Negative affect -0.35 -0.26 -0.45 -0.16 -0.28 -0.07 0.25 -0.33 1.00 -0.40 -0.32 -0.32 -0.12
GDP_social_interaction 0.83 0.91 0.92 0.76 0.43 0.06 -0.35 0.38 -0.40 1.00 0.86 0.23 0.18
Life_health_interaction 0.97 0.85 0.73 0.85 0.52 0.15 -0.44 0.45 -0.32 0.86 1.00 0.04 0.25
Support_GDP_ratio 0.10 -0.18 0.59 -0.09 0.13 0.08 0.10 0.30 -0.32 0.23 0.04 1.00 0.04
Generosity_corruption_ratio 0.27 0.15 0.15 0.12 0.33 0.84 -0.42 0.30 -0.12 0.18 0.25 0.04 1.00
Life Ladder
Log GDP per capita
Social support
Healthy life expectancy at birth
Freedom to make life choices
Generosity
Perceptions of corruption
Positive affect
Negative affect
GDP_social_interaction
Life_health_interaction
Support_GDP_ratio
Generosity_corruption_ratio
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

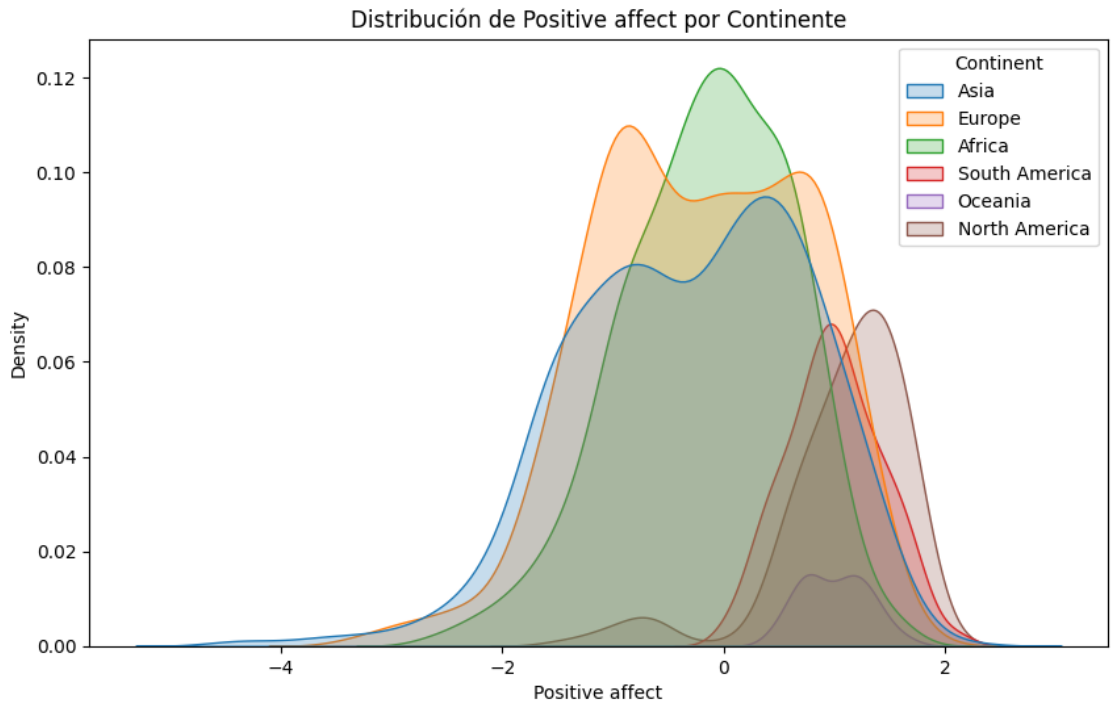

Tendencia de Life Ladder a lo largo del tiempo
Life Ladder
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
year

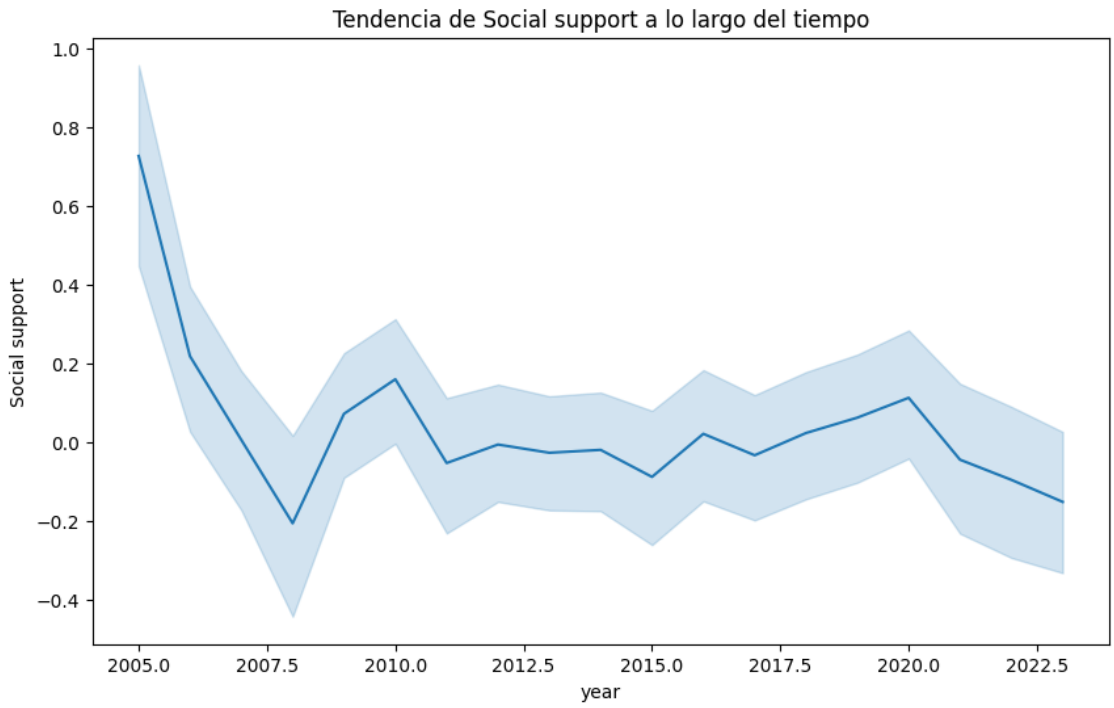

Tendencia del Log GDP per capita a lo largo del tiempo
Log GDP per capita
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
year


Promedio de Generosity por Continente
Generosity
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Asia
Europe
Africa
South America
Oceania
North America
Continent


Distribución de Log GDP per capita por Continente
Continent
Asia
Europe
Africa
South America
Oceania
North America
Density
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
Log GDP per capita


Comparación de Life Ladder y Positive affect por Continente
Life Ladder
2
1
0
−1
−2
−3
−4
Positive affect
−3.0
−1.5
0.0
1.5
Asia
Europe
Africa
South America
Oceania
North America
Continent





Dado que en este punto estamos revisando datos estandarizados, los gráficos muestran valores

negativos y positivos para cada variable estudiada.

Luego de estandarizar los datos no se observan cambios en los histogramas de las variables, valores de la matriz de correlación, tendencia de Life Ladder a lo largo del tiempo, tendencia de Log GDP per cápita a lo largo del tiempo y distribución de Log GDP per cápita por continente.

En el caso del gráfico de barras que representa al promedio de Generosity por continente, Oceanía tiene el mayor promedio. Sorprendentemente, Sudamérica presenta el menor valor promedio, siendo menor a los valores de África y Europa.

Al comparar Life Ladder con Positive Affect se corrobora que valores negativos de Positive Affect se encuentran correlacionados con valores negativos de Life Ladder. En el caso de la distribución de Positive Affect por continente se observa que Asia, África y Europa se concentran en la zona negativa, mientras que salvo por una cola de la distribución sudamericana, Sudamérica, Norteamérica y Oceanía se concentran en los valores entre 0 y 2.

Al analizar la tendencia de Social support a lo largo del tiempo se comprueba que ocurrió una disminución abrupta aproximadamente en el año 2008 (crisis subprime), tras lo cual se ha mantenido en valores entre 0 y 0.2, lo cual nos permite concluir que disminuyó permanentemente luego de la crisis subprime.

- Ya con una idea más clara de lo que queremos representar mediante las visualizaciones, a través el siguiente código decidimos mostrar la información única de las variables "Continent", "Country name" y "year". Todo esto para validar las posibles opciones que se pueden graficar, basándonos en conocimientos previos y globales que manejamos.

[23]:
```
# Mostrar los valores únicos en las columnas 'Continent', 'Country name' y␣
↪'year'
continent_values = df_heatmap['Continent'].unique() if 'Continent' in␣
↪df_cleaned.columns else "No 'continent' column found"
country_values = df_heatmap['Country name'].unique() if 'Country name' in␣
↪df_cleaned.columns else "No 'Country name' column found"
year_values = df_heatmap['year'].unique() if 'year' in df_cleaned.columns else␣
↪"No 'year' column found"

continent_values, country_values, year_values
```

[23]:
```
(array(['Asia', 'Europe', 'Africa', 'South America', 'Oceania',
        'North America'], dtype=object),
 array(['Afghanistan', 'Albania', 'Algeria', 'Angola', 'Argentina',
        'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Bahrain',
        'Bangladesh', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
        'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil',
        'Bulgaria', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon',
        'Canada', 'Central African Republic', 'Chad', 'Chile', 'China',
        'Colombia', 'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)',
        'Costa Rica', 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
        'Djibouti', 'Dominican Republic', 'Ecuador', 'Egypt',
        'El Salvador', 'Estonia', 'Eswatini', 'Ethiopia', 'Finland',
        'France', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana',
```

```
       'Greece', 'Guatemala', 'Guinea', 'Guyana', 'Haiti', 'Honduras',
       'Hong Kong S.A.R. of China', 'Hungary', 'Iceland', 'India',
       'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Ivory Coast', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya',
       'Kosovo', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
       'Lesotho', 'Liberia', 'Libya', 'Lithuania', 'Luxembourg',
       'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
       'Mauritania', 'Mauritius', 'Mexico', 'Moldova', 'Mongolia',
       'Montenegro', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia',
       'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua', 'Niger',
       'Nigeria', 'North Macedonia', 'Norway', 'Oman', 'Pakistan',
       'Panama', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal',
       'Qatar', 'Romania', 'Russia', 'Rwanda', 'Saudi Arabia', 'Senegal',
       'Serbia', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Somalia', 'Somaliland region', 'South Africa', 'South Korea',
       'South Sudan', 'Spain', 'Sri Lanka', 'State of Palestine', 'Sudan',
       'Suriname', 'Sweden', 'Switzerland', 'Syria',
       'Taiwan Province of China', 'Tajikistan', 'Tanzania', 'Thailand',
       'Togo', 'Trinidad and Tobago', 'Tunisia', 'Turkmenistan',
       'Türkiye', 'Uganda', 'Ukraine', 'United Arab Emirates',
       'United Kingdom', 'United States', 'Uruguay', 'Uzbekistan',
       'Venezuela', 'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype=object),
 array([2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
        2019, 2021, 2022, 2023, 2007, 2020, 2006, 2005]))
```

- Dentro de las opciones que decidimos mostrar, consideramos que un heatmap es una alternativa válida para representar la forma en que se encuentran correlacionadas las variables, lo que ayudará a saber cuáles de ellas son las que más influyen sobre la variable en estudio (Life ladder).
- Para esto, hacemos llamados a la copia del dataframe limpio generado en una primera etapa del proceso, llamado "df_heatmap".

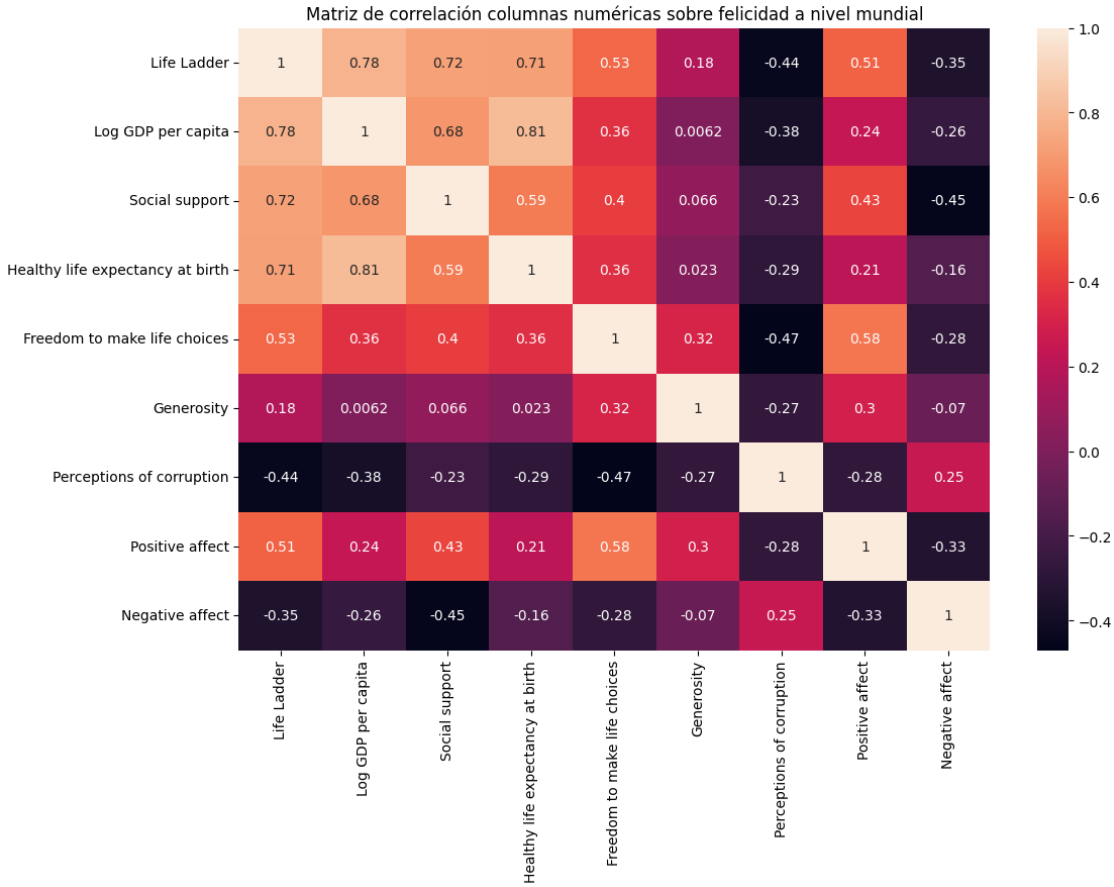
[24]:
```
# Seleccionamos solo las variables numéricas originales para trabajar esta␣
↪nueva visualización, y quitamos el año (year)
columnas_deseadas = ['Life Ladder', 'Log GDP per capita', 'Social support',␣
↪'Healthy life expectancy at birth',
                     'Freedom to make life choices', 'Generosity', 'Perceptions␣
↪of corruption', 'Positive affect', 'Negative affect']
df_numeric = df_heatmap[columnas_deseadas]
df_numeric.head()
```

[24]:
```
   Life Ladder  Log GDP per capita  Social support  \
0        3.724               7.350           0.451
1        4.402               7.509           0.552
2        4.758               7.614           0.539
3        3.832               7.581           0.521
```

```
4        3.783                7.661           0.521

   Healthy life expectancy at birth  Freedom to make life choices  Generosity  \
0                              50.5                         0.718       0.164
1                              50.8                         0.679       0.187
2                              51.1                         0.600       0.118
3                              51.4                         0.496       0.160
4                              51.7                         0.531       0.234

   Perceptions of corruption  Positive affect  Negative affect
0                      0.882            0.414            0.258
1                      0.850            0.481            0.237
2                      0.707            0.517            0.275
3                      0.731            0.480            0.267
4                      0.776            0.614            0.268
```

- Una vez validada la existencia de los datos en este dataframe, generamos un heatmap sencillo para comprender la correlación entre variables. La intención es identificar de qué forma podríamos visualizar esta información, y que por si sola cuente una historia.

[25]:
```python
# Previsualizaciones de Heatmap

# Calcular matiz de correlación
correlation_matrix = df_numeric.corr(numeric_only=True)

# Ajustar tamaño del gráfico
fig, ax = plt.subplots(figsize=(12, 8))

# Dibujar el heatmap
sns.heatmap(correlation_matrix, ax=ax, annot=True)
plt.title("Matriz de correlación columnas numéricas sobre felicidad a nivel␣
 ↪mundial")
plt.show()
```



Observaciones:

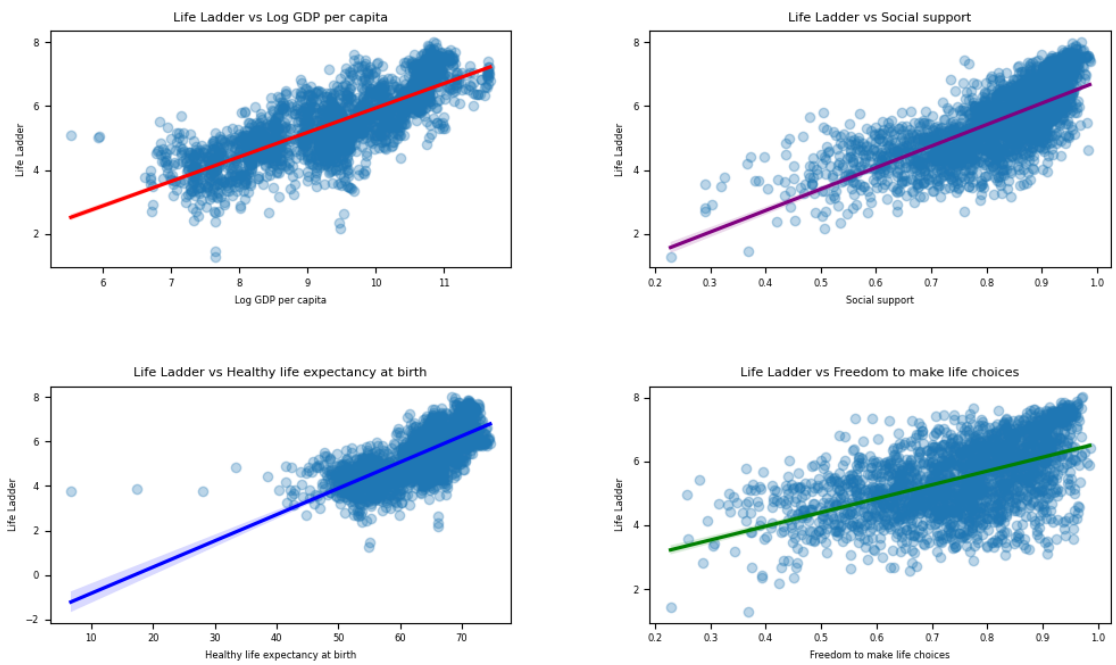
- A nivel mundial, la Satisfacción de la Vida (`Life Ladder`) presenta mayor correlación con las variables: PIB per cápita (`Log GRP per capita`) con un 0.78, el Apoyo Social (`Social support`) con un 0.72, la Esperanza de Vida saludable al nacer (`Healthy life expectancy at birth`) con un 0.71, y también, la Libertad de Elección (`Freedom to make life choices`) con un 0.53. Podríamos inferir que estas 4 variables contibuyen de forma positiva en la Satisfacción de vida de las personas.
- Por otro lado, la variable Percepción sobre la Corrupción (`Perceptions of corruption`) tiene una correlación de -0.44 respecto a la Satisfacción de la vida; mientras que el Afecto Negativo (`Negative affect`) tiene una correlación de -0.35. Podemos inferir que estas variables provocan una disminución respecto a la medida de Satisfacción de la Vida en las personas.

- De acuerdo con los datos más relevantes capturados y descritos anteriormente, visualizaremos cada una de estas variables y la compararemos con Life ladder para saber cómo es su impacto sobre esta.

[26]:
```
# ---- REFLEJAMOS LA RELACIÓN ENTRE VARIABLES QUE TIENEN UN IMPACTO POSITIVO␣
↪SOBRE LIFE LADDER, SEGÚN INFORMACIÓN ENTREGADA POR EL HEATMAP ----
```

```python
# Establecer el tamaño del gráfico
plt.figure(figsize=(10, 6))

# Parámetros personalizables
titulo_size = 8
ejes_size = 6
ticks_size = 6

# Subplot 1: Life Ladder vs Log GDP per capita
plt.subplot(2, 2, 1)  # Primera fila, primera columna
sns.regplot(x='Log GDP per capita', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'marker': 'o'}, line_kws={'color':␣
 ↪'red'})
plt.title('Life Ladder vs Log GDP per capita', fontsize=titulo_size)
plt.xlabel('Log GDP per capita', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(fontsize=ticks_size)

# Subplot 2: Life Ladder vs Social support
plt.subplot(2, 2, 2)  # Primera fila, segunda columna
sns.regplot(x='Social support', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'marker': 'o'}, line_kws={'color':␣
 ↪'purple'})
plt.title('Life Ladder vs Social support', fontsize=titulo_size)
plt.xlabel('Social support', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(fontsize=ticks_size)

# Subplot 3: Life Ladder vs Healthy life expectancy at birth
plt.subplot(2, 2, 3)  # Segunda fila, primera columna
sns.regplot(x='Healthy life expectancy at birth', y='Life Ladder',␣
 ↪data=df_numeric,
            scatter_kws={'alpha': 0.3, 'marker': 'o'}, line_kws={'color':␣
 ↪'blue'})
plt.title('Life Ladder vs Healthy life expectancy at birth',␣
 ↪fontsize=titulo_size)
plt.xlabel('Healthy life expectancy at birth', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(fontsize=ticks_size)

# Subplot 4: Life Ladder vs Freedom to make life choices
plt.subplot(2, 2, 4)  # Segunda fila, segunda columna
sns.regplot(x='Freedom to make life choices', y='Life Ladder', data=df_numeric,
```

```
            scatter_kws={'alpha': 0.3, 'marker': 'o'}, line_kws={'color':␣
 ↪'green'})
plt.title('Life Ladder vs Freedom to make life choices', fontsize=titulo_size)
plt.xlabel('Freedom to make life choices', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(fontsize=ticks_size)

# Ajustar el layout para que no se solapen
plt.tight_layout()

# Ajustar el espacio entre los gráficos
plt.subplots_adjust(wspace=0.3, hspace=0.5)  # Incrementa el espacio entre los␣
 ↪gráficos

# Mostrar la gráfica
plt.show()
```

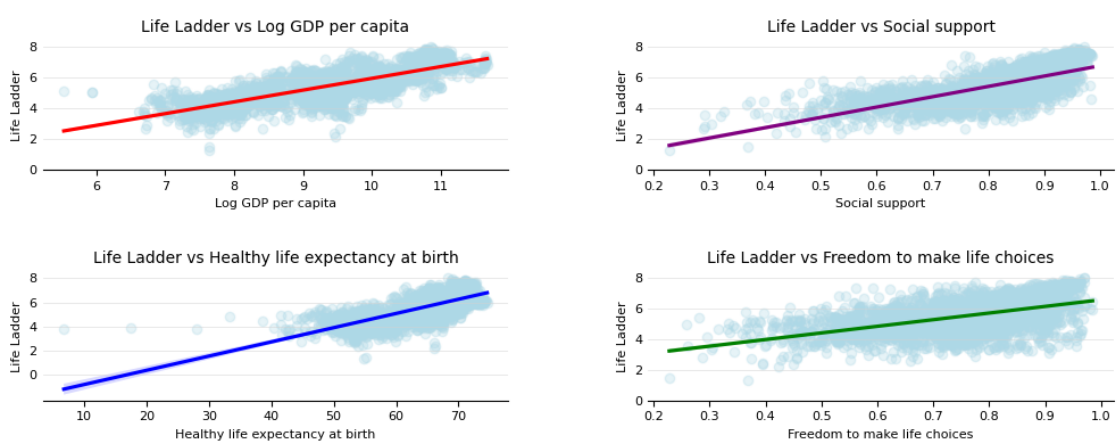


- Mediante estos Subplots podemos validar el impacto positivo que tienen estas variables sobre Life ladder. Sin embargo, para poder mostrar estos gráficos en nuestra visualización final debemos hacer un par de ajustes con la finalidad que los gráficos queden más llamativos y autoexplicativos. Es por esto que a continuación realizaremos algunas modificaciones adicionales que nos ayudarán a lograr nuestro objetivo.

[27]:
```
# ---- REFLEJAMOS LA RELACIÓN ENTRE VARIABLES QUE TIENEN UN IMPACTO POSITIVO␣
↪SOBRE LIFE LADDER, SEGÚN INFORMACIÓN ENTREGADA POR EL HEATMAP ----

# Establecer el tamaño del gráfico
plt.figure(figsize=(10, 4))

# Parámetros personalizables
titulo_size = 10
ejes_size = 8
ticks_size = 8
bubble_color = 'lightblue'  # Color editable de las burbujas

# Subplot 1: Life Ladder vs Log GDP per capita
plt.subplot(2, 2, 1)  # Primera fila, primera columna
sns.regplot(x='Log GDP per capita', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'red'})
plt.title('Life Ladder vs Log GDP per capita', fontsize=titulo_size)
plt.xlabel('Log GDP per capita', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)


# Subplot 2: Life Ladder vs Social support
plt.subplot(2, 2, 2)  # Primera fila, segunda columna
sns.regplot(x='Social support', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'purple'})
plt.title('Life Ladder vs Social support', fontsize=titulo_size)
plt.xlabel('Social support', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
```

```
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)


# Subplot 3: Life Ladder vs Healthy life expectancy at birth
plt.subplot(2, 2, 3)  # Segunda fila, primera columna
sns.regplot(x='Healthy life expectancy at birth', y='Life Ladder',␣
↪data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'blue'})
plt.title('Life Ladder vs Healthy life expectancy at birth',␣
↪fontsize=titulo_size)
plt.xlabel('Healthy life expectancy at birth', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)


# Subplot 4: Life Ladder vs Freedom to make life choices
```

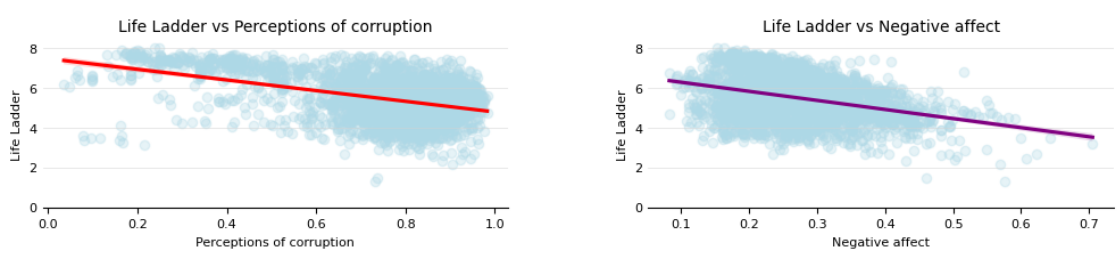
```
plt.subplot(2, 2, 4)  # Segunda fila, segunda columna
sns.regplot(x='Freedom to make life choices', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
 ↪line_kws={'color': 'green'})
plt.title('Life Ladder vs Freedom to make life choices', fontsize=titulo_size)
plt.xlabel('Freedom to make life choices', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
 ↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)


# Ajustar el layout para que no se solapen
plt.tight_layout()

# Ajustar el espacio entre los gráficos
plt.subplots_adjust(wspace=0.3, hspace=0.8)

# Mostrar la gráfica
plt.show()
```



- Como podemos apreciar en estos gráficos resultantes, ya hemos quitados un par de características innatas de los subplots, por lo tanto, visualmente ya son más atractivos y la información se logra entender con mayor rapidéz. También, se hace una prueba cambiando los colores de la gráfica para entender cómo mejorar dicho parámetro.
- A continuación, siguiendo el mismo esquema anterior visualizamos las variables que poseen un impacto negativo sobre Life ladder.

[28]:
```
# ---- REFLEJAR RELACIÓN ENTRE VARIABLES QUE TIENEN UN IMPACTO NEGATIVO SOBRE␣
↪LIFE LADDER ----

# Establecer el tamaño del gráfico
plt.figure(figsize=(10, 4))

# Parámetros personalizables
titulo_size = 10
ejes_size = 8
ticks_size = 8
bubble_color = 'lightblue'  # Color editable de las burbujas

# Subplot 1: Life Ladder vs Perceptions of corruption
plt.subplot(2, 2, 1)  # Primera fila, primera columna
sns.regplot(x='Perceptions of corruption', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'red'})
plt.title('Life Ladder vs Perceptions of corruption', fontsize=titulo_size)
plt.xlabel('Perceptions of corruption', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)
```

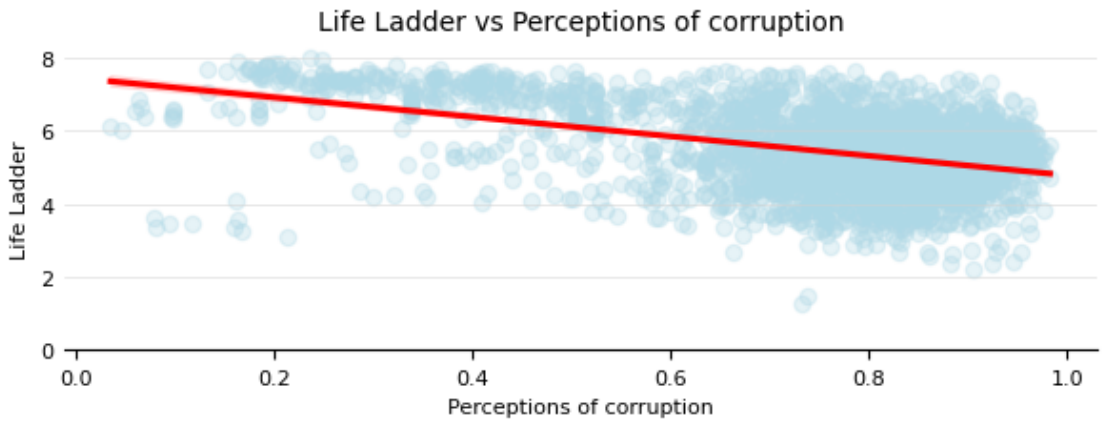
```
# Subplot 2: Life Ladder vs Negative affect
plt.subplot(2, 2, 2)  # Primera fila, segunda columna (disposición 2x2)
sns.regplot(x='Negative affect', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
 ↪line_kws={'color': 'purple'})
plt.title('Life Ladder vs Negative affect', fontsize=titulo_size)
plt.xlabel('Negative affect', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
 ↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)

# Ajustar el layout para que no se solapen
plt.tight_layout()

# Ajustar el espacio entre los gráficos
plt.subplots_adjust(wspace=0.3, hspace=0.4)  # Ajuste del espacio entre gráficos

# Mostrar la gráfica
plt.show()
```

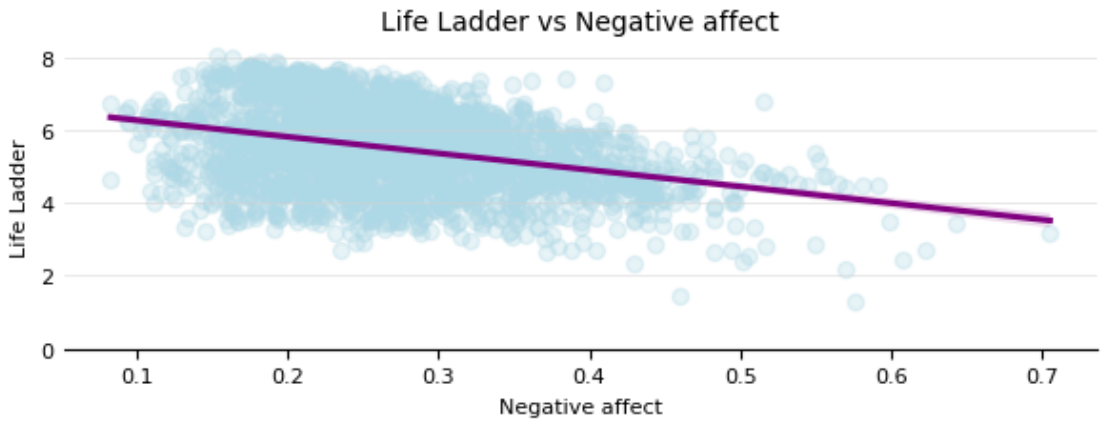


- En esta sección, nos dimos cuenta que para la visualización final sería buena opción mostrar estos gráficos ordenados verticalmente. Por lo tanto, a continuación hacemos una prueba para validar esta nueva idea.

[29]:
```
# ---- REFLEJAR RELACIÓN ENTRE VARIABLES QUE TIENEN UN IMPACTO NEGATIVO SOBRE␣
↪LIFE LADDER ----

# Establecer el tamaño del gráfico
plt.figure(figsize=(6, 5))

# Parámetros personalizables
titulo_size = 10
ejes_size = 8
ticks_size = 8
bubble_color = 'lightblue'  # Color editable de las burbujas

# Subplot 1: Life Ladder vs Perceptions of corruption
plt.subplot(2, 1, 1)  # Primera fila, primer gráfico (disposición 2x1)
sns.regplot(x='Perceptions of corruption', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'red'})
plt.title('Life Ladder vs Perceptions of corruption', fontsize=titulo_size)
plt.xlabel('Perceptions of corruption', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)

# Subplot 2: Life Ladder vs Negative affect
plt.subplot(2, 1, 2)  # Segunda fila, segundo gráfico (disposición 2x1)
sns.regplot(x='Negative affect', y='Life Ladder', data=df_numeric,
            scatter_kws={'alpha': 0.3, 'color': bubble_color, 'marker': 'o'},␣
↪line_kws={'color': 'purple'})
plt.title('Life Ladder vs Negative affect', fontsize=titulo_size)
plt.xlabel('Negative affect', fontsize=ejes_size)
plt.ylabel('Life Ladder', fontsize=ejes_size)
plt.xticks(fontsize=ticks_size)
```

```
plt.yticks(ticks=range(0, 9, 2), fontsize=ticks_size)  # Ajustar el eje Y para␣
 ↪que tenga ticks de 2 en 2

# Eliminar los ticks del eje Y
plt.tick_params(axis='y', length=0)

# Agregar líneas de fondo grises y ajustar su estilo
plt.grid(True, which='both', axis ='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)

# Eliminar las líneas negras de los ejes (spines)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(True)

# Ajustar el layout para que no se solapen
plt.tight_layout()

# Ajustar el espacio entre los gráficos
plt.subplots_adjust(wspace=0.5, hspace=0.6)  # Ajuste del espacio entre gráficos

# Mostrar la gráfica
plt.show()
```





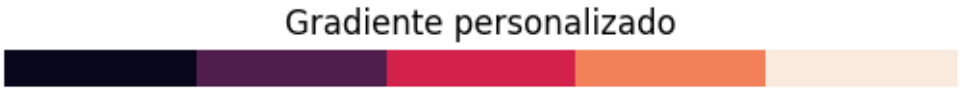
- Con los gráficos ordenados de esta manera, se podrían dejar en una posición específica que complementen la explicación del Heatmap.
- Cuando generemos la visualización final los incorporaremos y generaremos modificaciones para que todo quede alineado (colores, formas, posiciones, tamaños, etc.)
- Ya con las ideas más claras, decidimos reutilizar código ya trabajado en la construcción de un heatmap rotado. Es por esto que a continuación añadimos la función que nos permitirá realizar este trabajo.

[30]:
```
# Definir función para rotar de ejes

def rotate_axes(fig, rect, angle):
    tr = Affine2D().scale(1, 1).rotate_deg(angle)

    grid_helper = floating_axes.GridHelperCurveLinear(
        tr, extremes=(0, correlation_matrix.shape[0],0, correlation_matrix.
↪shape[0]))

    ax1 = floating_axes.FloatingSubplot(fig, rect, grid_helper=grid_helper)
```

```
    fig.add_subplot(ax1)
    ax1.axis[:].set_visible(False)
    aux_ax = ax1.get_aux_axes(tr)
    return aux_ax
```

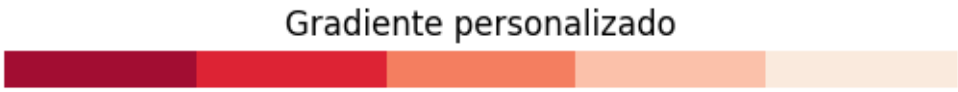
- En esta oportunidad, probamos con diferentes paletas de colores con distintos tonos, a fin de encontrar uno representativo que no contuviera colores demasiados intensos que molesten visualmente al momento de revisar toda la información.
- Partimos con una paleta simple:

[31]:
```
# Definir paleta de colores para heatmap

colores_personalizados = ['#06071b', '#07081f', '#1d122d', '#30183b',␣
↪'#431c46','#491e4a','#682056',
                          ␣
↪'#891d5b','#a41a5b','#b91658','#c81952','#dc2b47','#e43840','#e8403c','#f36e4b',
                          ␣
↪'#f2704c','#f3865d','#f6ab82','#f7b48f','#f6bc99','#f9e0cd','#faeadd']  #␣
↪Gradiente de color oscuro a claro

# Paleta de colores personalizado a partir de la lista de colores
mi_paleta = LinearSegmentedColormap.from_list("mi_paleta",␣
↪colores_personalizados, N=5)

# Crear un gradiente lineal
gradient = np.linspace(0, 1, 256)
gradient = np.vstack((gradient, gradient))  # Se apila para que el gradiente␣
↪sea continuo

# Mostrar la imagen del gradiente
plt.imshow(gradient, aspect=5, cmap=mi_paleta)  # Cambia el aspecto para␣
↪hacerlo más angosto
plt.title('Gradiente personalizado')
plt.axis('off')  # Ocultar los ejes
plt.show()
```



[32]:
```
# Definir nueva paleta de colores para heatmap
```

```
colores_personalizados_2 =␣
↪['#a20d32','#ee2e3e','#d71634','#ef4d35','#f36f51','#f68d6f','#f8a98e','#fcc9b4','#fde2d6',
↪ # Gradiente de color oscuro a claro

# Paleta de colores personalizado a partir de la lista de colores
mi_paleta_2 = LinearSegmentedColormap.from_list("mi_paleta_2",␣
↪colores_personalizados_2, N=5)

# Crear un gradiente lineal
gradient = np.linspace(0, 1, 256)
gradient = np.vstack((gradient, gradient))  # Se apila para que el gradiente␣
↪sea continuo

# Mostrar la imagen del gradiente
plt.imshow(gradient, aspect=5, cmap=mi_paleta_2)  # Cambia el aspecto para␣
↪hacerlo más angosto
plt.title('Gradiente personalizado')
plt.axis('off')  # Ocultar los ejes
plt.show()
```



- En el transcurso, recordamos que en el heatmap existen valores mayores que cero y menos que cero. Por lo tanto, decidimos crear una paleta de colores divergente que explicase esta situación de mejor forma mediante la representación en el gráfico.
- A continuación, se muestra el gráfico donde realizamos diferentes pruebas a fin de escoger la más adecuada que se adapte a nuestra visualización futura.

[33]:
```
# Definir nueva paleta de colores divergente
#colores_divergentes = ['#a20d32', '#d71634', '#ee2e3e', '#ef4d35',␣
↪'#f36f51','#f68d6f', '#f8a98e', '#fcc9b4', '#fde2d6', '#faeadd']
#colores_divergentes = ['#a20d32', '#d71634', '#ee2e3e', '#ef4d35', '#f36f51',␣
↪'#fcc9b4', '#fde2d6', '#faeadd','#D3D3D3','#A1D6F9','#6FAFE9','#4A90E2']
#colores_divergentes = ['#a20d32','#C75A5A','#F9DCD0', '#D9F1F4', '#A3D5E3',␣
↪'#75B4D3', '#5392C1']
#colores_divergentes =  ['#C98B99', '#E4B7C0', '#F5E1E8', '#C6E7EC', '#A5D5DE',␣
↪'#89B8C6']
#colores_divergentes = ['#a20d32','#d71634','#ee2e3e','#fde2d6',␣
↪'#faeadd','#fcc9b4','#f8a98e','#f36f51','#ef4d35']
#colores_divergentes = ['#4A90E2', '#6FAFE9', '#A1D6F9', '#D3D3D3', '#F6D4C2',␣
↪'#F4A482', '#F47362', '#E6492D']
```

```
#colores_divergentes = ['#D6A564', '#EBC68A', '#F2D8A6', '#F9EAD2', '#FFFFFF',␣
↪'#C8E8E5', '#B4DAD2', '#7BB0A6', '#56968C']
#colores_divergentes = ['#6FA3B6', '#92B9C8', '#BFD7DE', '#F3F3F3', '#F2C7BD',␣
↪'#E8A696', '#D68C78', '#C57554']
colores_divergentes = ['#6FA3B6', '#92B9C8', '#BFD7DE', '#F3F3F3', '#E8A696',␣
↪'#D68C78', '#C57554']
#colores_divergentes = ['#C7D7B4', '#A9C49B', '#8FB68B', '#7BAC7B', '#669E6D',␣
↪'#4F8C75', '#4C6D83', '#40587A']
#colores_divergentes = ['#154B3D', '#357C63', '#7EB7A0', '#F0EDED', '#C49AC5',␣
↪'#7C5089', '#532A61']
#colores_divergentes = ['#ef8a62','#f7f7f7','#67a9cf']

# Paleta de colores personalizada a partir de la lista de colores (N es la␣
↪cantidad de divisiones de la paleta)
mi_paleta_divergente = LinearSegmentedColormap.
↪from_list("mi_paleta_divergente", colores_divergentes, N=256)

# Crear un gradiente lineal que va de -1 a 1 para simular datos con valores␣
↪tanto negativos como positivos
gradient = np.linspace(-1, 1, 256)  # Los valores van de -1 a 1 (donde 0 es el␣
↪punto medio)
gradient = np.vstack((gradient, gradient))  # Se apila para que el gradiente␣
↪sea continuo

# Mostrar la imagen del gradiente con la nueva paleta divergente
plt.imshow(gradient, aspect=5, cmap=mi_paleta_divergente)  # Usa la nueva paleta
plt.title('Gradiente de color divergente')
plt.axis('off')  # Ocultar los ejes
plt.show()
```

- A continuación, se encuentra el código donde desarrollamos la visualización final de este análisis, donde nos enfocamos claramente en cómo los diferentes indicadores de bienestar influyen sobre la Expectativa de vida (Life ladder).
- Al leer detalladamente el código, se podrá apreciar cómo hicimos llamado a la función donde rotamos el heatmap, aplicamos etiquedas en los ejes del gráfico, incorporamos títulos y subtítulos. Además, incorporamos el gradiente de colores y sumamos a la visualización los gráficos de regresión lineal simple para complementar la historia.
- Como ya conocíamos los parámetros que nos permiten modificar estas visualizaciones, realizamos cambios de tamaño de los gráficos, cambios de color, ajustamos los tamaños y adecuamos las posiciones para que quedaran bien seccionados.

- También, nació la idea de explicar cómo ha sido la tendencia de Life ladder a lo largo de los años, es por esto que añadimos un gráfico de lineas al final de la visualización.
- Por último, mencionar que incorporamos algunas conclusiones que complementan la comprensión del heatmap para aquellas personas que no están acostumbradas a analizar este tipo de visualizaciones.

VISUALIZACIÓN FINAL

[34]:
```
# Previsualizaciones de Heatmap rotado

# Calcular matiz de correlación
correlation_matrix = df_numeric.corr(numeric_only=True)

# Ajustar tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 6))
ax0 = rotate_axes(fig, 111, -135)

# Dibujar el heatmap
sns.heatmap(correlation_matrix, annot=False, cmap=mi_paleta_divergente, ax=ax0,␣
 ↪linewidths=0.2, cbar=False)


# ---- Añadir etiquetas a los ejes del heatmap ----

plt.text(-7.4, -8.4, 'Life Ladder', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.8, -10.0, 'Log GDP per capita', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-6.4, -10.15, 'Social support', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.9, -13.0, 'Healthy life expectancy at birth', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-6.85, -13.45, 'Freedom to make life choices', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-3.85, -11.8, 'Generosity', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-4.9, -14.3, 'Perceptions of corruption', color = 'black', rotation =␣
 ↪45, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.8, -13.71, 'Positive affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.26, -14.6, 'Negative affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)

plt.text(0.36, -12.6, 'Negative affect', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
```

```
plt.text(1.1, -11.88, 'Positive affect', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(1.7, -11.2, 'Perceptions of corruption', color = 'black', rotation =␣
↪315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(2.4, -10.45, 'Generosity', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.15, -9.8, 'Freedom to make life choices', color = 'black', rotation␣
↪= 315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.87, -9.0, 'Healthy life expectancy at birth', color = 'black',␣
↪rotation = 315, rotation_mode = 'anchor', transform_rotates_text = True,␣
↪fontsize=9)
plt.text(4.6, -8.3, 'Social support', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(5.28, -7.62, 'Log GDP per capita', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(5.95, -6.9, 'Life Ladder', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)


# ---- Añadir Titulo, Subtítulo, Líneas -----

# Título principal
plt.text(0.1, 1.2, "Análisis del Indicador de Satisfacción de Vida (Life␣
↪Ladder) en el mundo",
     fontsize=14, fontweight='bold', ha='left', transform=fig.transFigure)

# Insertar la línea sobre el título
fig.add_artist(
    plt.Line2D([0.1, 1.5], [1.24, 1.24], lw=1, color="black",␣
↪solid_capstyle="butt", transform=fig.transFigure)
)

# Subtítulo
plt.text(x=0.1, y=1.1, s="¿Cómo impactan los indicadores de bienestar sobre la␣
↪Satisfacción de Vida?",
         fontsize=11, fontweight='normal', ha='left', transform=fig.transFigure)

# Insertar Línea corta sobre el subtítulo
fig.add_artist(
    plt.Line2D([0.1, 0.26], [1.14, 1.14], lw=0.5, color="black",␣
↪solid_capstyle="butt", transform=fig.transFigure)
)

# ----- Incorporar Gradiente de Colores sobre Heatmap ----

# Crear un eje separado para la barra de color en la parte superior derecha
```

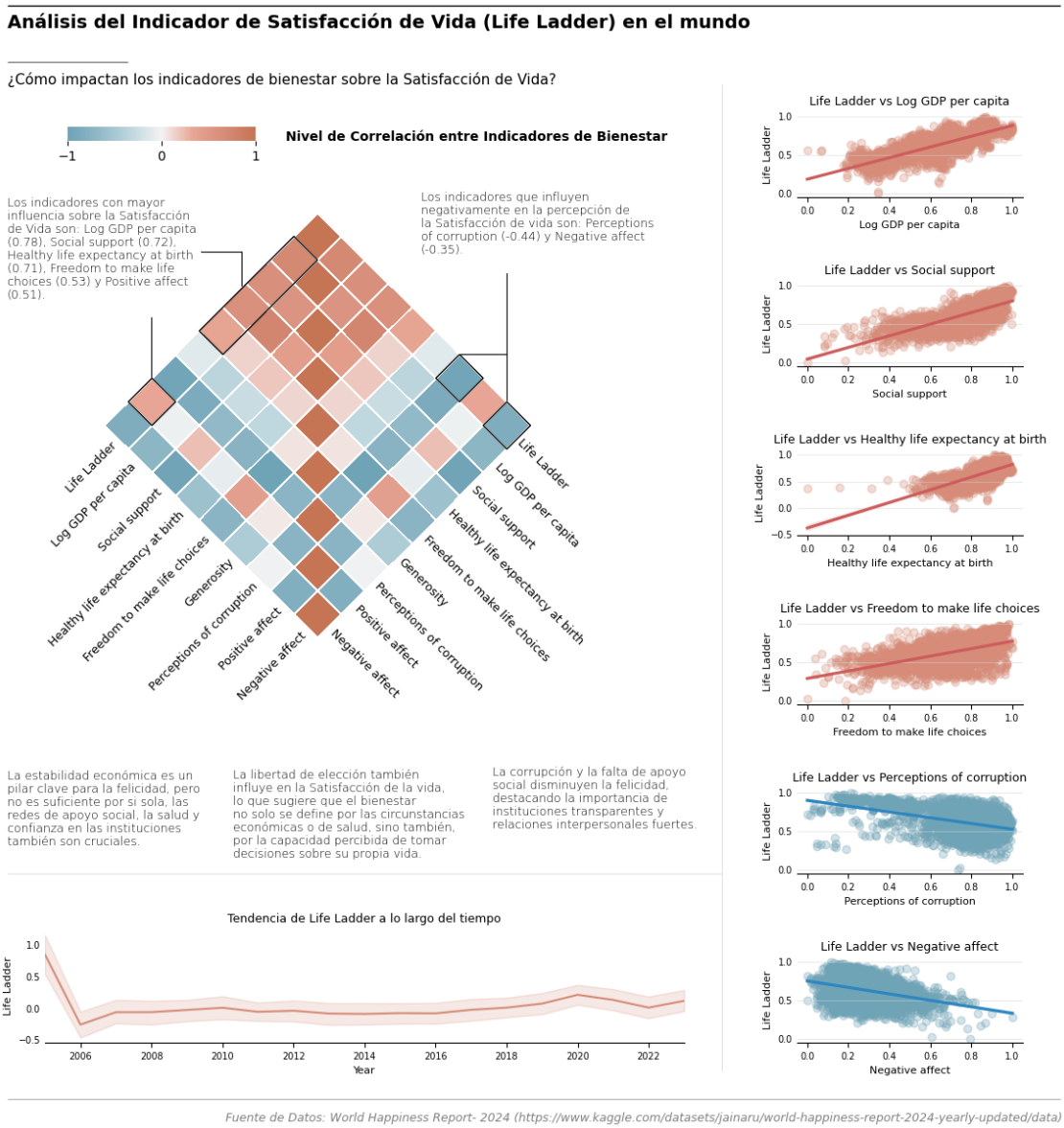
```
cbar_ax = fig.add_axes([0.18, 1.0, 0.25, 0.025])  # [left, bottom, width,␣
↪height]

# Generar la barra de color en el eje separado
sm = plt.cm.ScalarMappable(cmap=mi_paleta_divergente, norm=plt.
↪Normalize(vmin=-1, vmax=1))
cbar = fig.colorbar(sm, cax=cbar_ax, orientation='horizontal', ticks=[-1.0,0.0,␣
↪1.0])

# Eliminar el contorno de la barra de color
cbar.outline.set_visible(False)

# Mover los ticks más hacia el centro (interior) de la barra
cbar.ax.tick_params(axis='x', which='both', direction='out')  # Ticks hacia␣
↪adentro de la barra

# Añadir título de la barra de color
plt.text(0.47, 1.0, 'Nivel de Correlación entre Indicadores de Bienestar ',␣
↪fontsize=10, fontweight='bold', ha='left', transform=fig.transFigure)  #␣
↪Negrita


# ---- Añadir Gráficos de Regresión encima del Heatmap ----

# Gráfico 1: Life Ladder vs Log GDP per capita
reg_ax1 = fig.add_axes([1.15, 0.9, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Log GDP per capita', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Log GDP per capita', fontsize=9)
reg_ax1.set_xlabel('Log GDP per capita', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 2: Life Ladder vs Social support
reg_ax1 = fig.add_axes([1.15, 0.6, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
```

```
sns.regplot(x='Social support', y='Life Ladder', data=df, scatter_kws={'alpha':␣
↪0.3, 'color': '#D68C78', 'marker': 'o'}, line_kws={'color': '#CD5C5C'},␣
↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Social support', fontsize=9)
reg_ax1.set_xlabel('Social support', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 3: Life Ladder vs Healthy life expectancy at birth
reg_ax1 = fig.add_axes([1.15, 0.3, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Healthy life expectancy at birth', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Healthy life expectancy at birth', fontsize=9)
reg_ax1.set_xlabel('Healthy life expectancy at birth', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 4: Life Ladder vs Freedom to make life choices
reg_ax1 = fig.add_axes([1.15, 0.0, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Freedom to make life choices', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Freedom to make life choices', fontsize=9)
reg_ax1.set_xlabel('Freedom to make life choices', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
```

```
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 5: Life Ladder vs Perceptions of corruption
reg_ax1 = fig.add_axes([1.15, -0.3, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Perceptions of corruption', y='Life Ladder', data=df,␣
 ↪scatter_kws={'alpha': 0.3, 'color': '#6FA3B6', 'marker': 'o'},␣
 ↪line_kws={'color': '#2e86c1'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Perceptions of corruption', fontsize=9)
reg_ax1.set_xlabel('Perceptions of corruption', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 6: Life Ladder vs Negative affect
reg_ax1 = fig.add_axes([1.15, -0.6, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Negative affect', y='Life Ladder', data=df, scatter_kws={'alpha':
 ↪ 0.3, 'color': '#6FA3B6', 'marker': 'o'}, line_kws={'color': '#2e86c1'},␣
 ↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Negative affect', fontsize=9)
reg_ax1.set_xlabel('Negative affect', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)


# ---- Añadir gráfico de lineas para representar tendencia de Life Ladder a lo␣
 ↪largo de los años ----

min_year = df_cleaned['year'].min()
```

```
max_year = df_cleaned['year'].max()
reg_ax1 = fig.add_axes([0.15, -0.6, 0.85, 0.2])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.lineplot(x='year', y='Life Ladder', data=df_cleaned, ax=reg_ax1,␣
↪color='#D68C78')
reg_ax1.set_xlim(min_year, max_year)
reg_ax1.set_title('Tendencia de Life Ladder a lo largo del tiempo', fontsize=9)
reg_ax1.set_xlabel('Year', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# ---- Añadir rectángulos y líneas para destacar información ----

# Lado izquierdo
ax0.add_patch(Rectangle((1, 0), 4, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((7, 0), 1, 1, fill=None, alpha=1))
ax.axvline(0.37, 0.9, 0.7, color = 'black', linewidth=0.8)
ax.axvline(0.215, 0.75, 0.6, color = 'black', linewidth=0.8)
ax.axhline(0.9, 0.3, 0.37, color = 'black', linewidth=0.8 )

# Lado derecho
ax0.add_patch(Rectangle((0, 6), 1, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((0, 8), 1, 1, fill=None, alpha=1))
ax.axvline(0.824, 0.85, 0.55, color = 'black', linewidth=0.8)
ax.axhline(0.665, 0.74, 0.824, color = 'black', linewidth=0.8 )

# ---- Texto descriptivo Heatmap ----

plt.text(0.1, 0.72, 'Los indicadores con mayor \ninfluencia sobre la␣
↪Satisfacción \nde Vida son: Log GDP per capita \n(0.78), Social support (0.
↪72), \nHealthy life expectancy at birth \n(0.71), Freedom to make life␣
↪\nchoices (0.53) y Positive affect \n(0.51).', fontsize=9, fontweight=False,␣
↪ha='left', transform=fig.transFigure)  # Negrita
plt.text(0.65, 0.8, 'Los indicadores que influyen \nnegativamente en la␣
↪percepción de \nla Satisfacción de vida son: Perceptions \nof corruption (-0.
↪44) y Negative affect \n(-0.35).', fontsize=9, fontweight=False, ha='left',␣
↪transform=fig.transFigure)  # Negrita

# ---- Texto conslusiones Heatmap ----
```

```
plt.text(0.1, -0.25, 'La estabilidad económica es un \npilar clave para la␣
↪felicidad, pero \nno es suficiente por si sola, las \nredes de apoyo social,␣
↪la salud y \nconfianza en las instituciones \ntambién son cruciales.',␣
↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
↪Negrita
plt.text(0.4, -0.272, 'La libertad de elección también \ninfluye en la␣
↪Satisfacción de la vida, \nlo que sugiere que el bienestar \nno solo se␣
↪define por las circunstancias\neconómicas o de salud, sino también, \npor la␣
↪capacidad percibida de tomar \ndecisiones sobre su propia vida.',␣
↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
↪Negrita
plt.text(0.745, -0.22, 'La corrupción y la falta de apoyo \nsocial disminuyen␣
↪la felicidad, \ndestacando la importancia de \ninstituciones transparentes y␣
↪\nrelaciones interpersonales fuertes.', fontsize=9, fontweight=False,␣
↪ha='left', transform=fig.transFigure)  # Negrita

# ---- Ajustes linea inferior ----

plt.text(x=0.39, y=-0.74, s="Fuente de Datos: World Happiness Report- 2024␣
↪(https://www.kaggle.com/datasets/jainaru/
↪world-happiness-report-2024-yearly-updated/data)",
        fontsize=9, color='gray', style='italic', ha='left', transform=fig.
↪transFigure)

# Insertar una línea sobre datos del pie de página
fig.add_artist(plt.Line2D([0.1, 1.5], [- 0.7, -0.7], lw=0.5, color="grey",␣
↪solid_capstyle="butt", transform=fig.transFigure))


# ---- Lineas divisorias  --

fig.add_artist(plt.Line2D([1.05, 1.05], [- 0.65, 1.1], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))
fig.add_artist(plt.Line2D([0.1, 1.05], [- 0.3, -0.3], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))


# ----- Configuraciones Extras ------

ax.set_axis_off()

# Mostrar gráfico
plt.show()
```



Una vez finalizado nuestro Heatmap junto a nuestras otras 7 gráficas, crearemos un **Bumpchart** para generar un ranking de los países pertenecientes a América del Sur y ver como ha ido evolucionando su Life Ladder a través del tiempo

[35]:
```
# Agrupamos en un nuevo df, los países de América del sur

paises_america_sur = ['Argentina', 'Bolivia', 'Brasil', 'Chile', 'Colombia',␣
 ↪'Ecuador', 'Guyana', 'Paraguay', 'Perú', 'Surinam', 'Uruguay', 'Venezuela']
df_america_sur = df_heatmap[df_heatmap['Country name'].isin(paises_america_sur)]
```

[36]:
```
df_america_sur.head()
```

[36]:

```
   Country name  year  Life Ladder  Log GDP per capita  Social support  \
46    Argentina  2006        6.313               9.937           0.938
47    Argentina  2007        6.073              10.013           0.862
48    Argentina  2008        5.961              10.043           0.892
49    Argentina  2009        6.424               9.972           0.919
50    Argentina  2010        6.441              10.066           0.927

    Healthy life expectancy at birth  Freedom to make life choices  \
46                             65.82                         0.733
47                             65.94                         0.653
48                             66.06                         0.678
49                             66.18                         0.637
50                             66.30                         0.730

    Generosity  Perceptions of corruption  Positive affect  Negative affect  \
46      -0.162                      0.852            0.748            0.328
47      -0.146                      0.881            0.750            0.279
48      -0.137                      0.865            0.720            0.318
49      -0.135                      0.885            0.762            0.237
50      -0.132                      0.855            0.765            0.211

        Continent
46  South America
47  South America
48  South America
49  South America
50  South America
```

Luego de agrupar la nueva base con sólo países de America del Sur, la vamos a pivotear para poder crear un bumpchart.

[37]:

```
# Pivotar los datos para tener la forma adecuada para el bump chart
df_pivot = df_america_sur.pivot(index='year', columns='Country name',␣
↪values='Life Ladder')
```

[38]:

```
df_pivot.head(25)
```

[38]:

```
Country name  Argentina  Bolivia  Chile  Colombia  Ecuador  Guyana  Paraguay  \
year
2005                NaN      NaN    NaN       NaN      NaN     NaN       NaN
2006              6.313    5.374  6.063     6.025    5.024     NaN     4.730
2007              6.073    5.628  5.698     6.138    4.996   5.993     5.272
2008              5.961    5.298  5.789     6.168    5.297     NaN     5.570
2009              6.424    6.086  6.494     6.272    6.022     NaN     5.576
2010              6.441    5.781  6.636     6.408    5.838     NaN     5.841
2011              6.776    5.779  6.526     6.464    5.795     NaN     5.677
2012              6.468    6.019  6.599     6.375    5.961     NaN     5.820
2013              6.582    5.767  6.740     6.607    6.019     NaN     5.936
```

```
2014              6.671    5.865  6.844     6.449    5.946     NaN     5.119
2015              6.697    5.834  6.533     6.388    5.964     NaN     5.560
2016              6.427    5.770  6.579     6.234    6.115     NaN     5.801
2017              6.039    5.651  6.320     6.157    5.840     NaN     5.713
2018              5.793    5.916  6.436     5.984    6.128     NaN       NaN
2019              6.086    5.674  5.942     6.350    5.809     NaN     5.653
2020              5.901    5.559  6.151     5.709    5.354     NaN     5.501
2021              5.908    5.569  6.436     5.290    5.435     NaN     5.576
2022              6.261    5.929  6.415     5.892    5.887     NaN     6.138
2023              6.393    5.860  6.230     5.904    5.852     NaN     6.214

Country name  Uruguay  Venezuela
year
2005              NaN      7.170
2006            5.786      6.525
2007            5.694        NaN
2008            5.664      6.258
2009            6.296      7.189
2010            6.062      7.478
2011            6.554      6.580
2012            6.450      7.067
2013            6.444      6.553
2014            6.561      6.136
2015            6.628      5.569
2016            6.171      4.041
2017            6.336      5.071
2018            6.372      5.006
2019            6.600      5.081
2020            6.310      4.574
2021            6.502      5.108
2022            6.671      5.949
2023            6.662      5.765
```

Podemos ver que existen valores NaN dentro del nuevo dataframe, por lo que para este caso eliminaremos el año 2005, eliminaremos Guyana, e iterpolaremos los valores NaN restantes.

[39]:
```
# Eliminar la fila con el año 2005
df_pivot_clean = df_pivot.drop(index=2005)
```

[40]:
```
df_pivot_clean.head()
```

[40]:
```
Country name  Argentina  Bolivia  Chile  Colombia  Ecuador  Guyana  Paraguay  \
year
2006              6.313    5.374  6.063     6.025    5.024     NaN     4.730
2007              6.073    5.628  5.698     6.138    4.996   5.993     5.272
2008              5.961    5.298  5.789     6.168    5.297     NaN     5.570
2009              6.424    6.086  6.494     6.272    6.022     NaN     5.576
2010              6.441    5.781  6.636     6.408    5.838     NaN     5.841
```

```
Country name  Uruguay  Venezuela
year
2006            5.786      6.525
2007            5.694        NaN
2008            5.664      6.258
2009            6.296      7.189
2010            6.062      7.478
```

[41]:
```
# Eliminar la columna "Guyana"
df_pivot_clean = df_pivot_clean.drop(columns='Guyana')
```

[42]:
```
df_pivot_clean.head()
```

[42]:
```
Country name  Argentina  Bolivia  Chile  Colombia  Ecuador  Paraguay  Uruguay  \
year
2006              6.313    5.374  6.063     6.025    5.024     4.730    5.786
2007              6.073    5.628  5.698     6.138    4.996     5.272    5.694
2008              5.961    5.298  5.789     6.168    5.297     5.570    5.664
2009              6.424    6.086  6.494     6.272    6.022     5.576    6.296
2010              6.441    5.781  6.636     6.408    5.838     5.841    6.062

Country name  Venezuela
year
2006              6.525
2007                NaN
2008              6.258
2009              7.189
2010              7.478
```

[43]:
```
# Interpolación lineal para rellenar valores NaN
df_pivot_clean = df_pivot_clean.interpolate(method='linear', axis=0)
```

[44]:
```
df_pivot_clean.head(25)
```

[44]:
```
Country name  Argentina  Bolivia  Chile  Colombia  Ecuador  Paraguay  Uruguay  \
year
2006              6.313    5.374  6.063     6.025    5.024     4.730    5.786
2007              6.073    5.628  5.698     6.138    4.996     5.272    5.694
2008              5.961    5.298  5.789     6.168    5.297     5.570    5.664
2009              6.424    6.086  6.494     6.272    6.022     5.576    6.296
2010              6.441    5.781  6.636     6.408    5.838     5.841    6.062
2011              6.776    5.779  6.526     6.464    5.795     5.677    6.554
2012              6.468    6.019  6.599     6.375    5.961     5.820    6.450
2013              6.582    5.767  6.740     6.607    6.019     5.936    6.444
2014              6.671    5.865  6.844     6.449    5.946     5.119    6.561
2015              6.697    5.834  6.533     6.388    5.964     5.560    6.628
```

```
2016              6.427    5.770  6.579     6.234    6.115     5.801    6.171
2017              6.039    5.651  6.320     6.157    5.840     5.713    6.336
2018              5.793    5.916  6.436     5.984    6.128     5.683    6.372
2019              6.086    5.674  5.942     6.350    5.809     5.653    6.600
2020              5.901    5.559  6.151     5.709    5.354     5.501    6.310
2021              5.908    5.569  6.436     5.290    5.435     5.576    6.502
2022              6.261    5.929  6.415     5.892    5.887     6.138    6.671
2023              6.393    5.860  6.230     5.904    5.852     6.214    6.662

Country name  Venezuela
year
2006             6.5250
2007             6.3915
2008             6.2580
2009             7.1890
2010             7.4780
2011             6.5800
2012             7.0670
2013             6.5530
2014             6.1360
2015             5.5690
2016             4.0410
2017             5.0710
2018             5.0060
2019             5.0810
2020             4.5740
2021             5.1080
2022             5.9490
2023             5.7650
```

Luego de ordenar y limpiar nuestro dataframe, crearemos nuestro BumpChart.

[45]:
```python
# Crear un nuevo DataFrame para almacenar los rankings
df_ranking = pd.DataFrame(index=df_pivot_clean.index)

# Iterar a través de los años
for year in df_pivot_clean.index:
  # Obtener los valores de Felicidad para el año actual
  year_data = df_pivot_clean.loc[year]
  # Ordenar los valores de Felicidad de mayor a menor
  sorted_year_data = year_data.sort_values(ascending=False)
  # Asignar un ranking del 1 al 8 a los países según su posición
  rank = 1
  for country in sorted_year_data.index:
    df_ranking.loc[year, country] = rank
    rank += 1
```

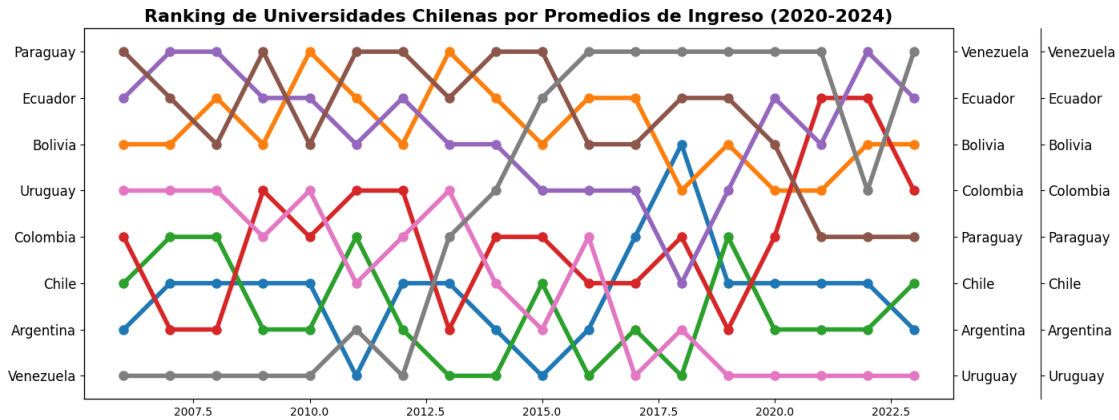
```python
# Mostrar el nuevo DataFrame con los rankings
print(df_ranking)
```

```
      Venezuela  Argentina  Chile  Colombia  Uruguay  Bolivia  Ecuador  \
year
2006        1.0        2.0    3.0       4.0      5.0      6.0      7.0
2007        1.0        3.0    4.0       2.0      5.0      6.0      8.0
2008        1.0        3.0    4.0       2.0      5.0      7.0      8.0
2009        1.0        3.0    2.0       5.0      4.0      6.0      7.0
2010        1.0        3.0    2.0       4.0      5.0      8.0      7.0
2011        2.0        1.0    4.0       5.0      3.0      7.0      6.0
2012        1.0        3.0    2.0       5.0      4.0      6.0      7.0
2013        4.0        3.0    1.0       2.0      5.0      8.0      6.0
2014        5.0        2.0    1.0       4.0      3.0      7.0      6.0
2015        7.0        1.0    3.0       4.0      2.0      6.0      5.0
2016        8.0        2.0    1.0       3.0      4.0      7.0      5.0
2017        8.0        4.0    2.0       3.0      1.0      7.0      5.0
2018        8.0        6.0    1.0       4.0      2.0      5.0      3.0
2019        8.0        3.0    4.0       2.0      1.0      6.0      5.0
2020        8.0        3.0    2.0       4.0      1.0      5.0      7.0
2021        8.0        3.0    2.0       7.0      1.0      5.0      6.0
2022        5.0        3.0    2.0       7.0      1.0      6.0      8.0
2023        8.0        2.0    3.0       5.0      1.0      6.0      7.0

      Paraguay
year
2006       8.0
2007       7.0
2008       6.0
2009       8.0
2010       6.0
2011       8.0
2012       8.0
2013       7.0
2014       8.0
2015       8.0
2016       6.0
2017       6.0
2018       7.0
2019       7.0
2020       6.0
2021       4.0
2022       4.0
2023       4.0
```

[46]:
```python
def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
```

```
            line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()

    # Crear el tercer eje si show_rank_axis es True
    if show_rank_axis:
        far_right_yaxis = left_yaxis.twinx()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

    for idx, col in enumerate(df_pivot_clean.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

        # Añadir scatter si es necesario
        if scatter:
            left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')

    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    for axis in [left_yaxis, right_yaxis]:
        axis.set_yticks(y_ticks)
        axis.set_ylim((lines + 0.5, 0.5))
        axis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    left_labels = df_ranking.iloc[0].sort_values().index
    right_labels = df_ranking.iloc[-1].sort_values().index

    left_yaxis.set_yticklabels(left_labels, fontsize=12)
    right_yaxis.set_yticklabels(right_labels, fontsize=12)

    # Configurar el tercer eje si es necesario
    if show_rank_axis:
```

```
        far_right_yaxis.spines["right"].set_position(("axes",␣
↪rank_axis_distance))
        far_right_yaxis.set_yticks(y_ticks)
        far_right_yaxis.set_yticklabels(right_labels, fontsize=12)
        far_right_yaxis.set_ylim((lines + 0.5, 0.5))
        far_right_yaxis.invert_yaxis()

    # Añadir título
    plt.title('Ranking de Universidades Chilenas por Promedios de Ingreso␣
↪(2020-2024)',
              fontsize=16, weight='bold')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```

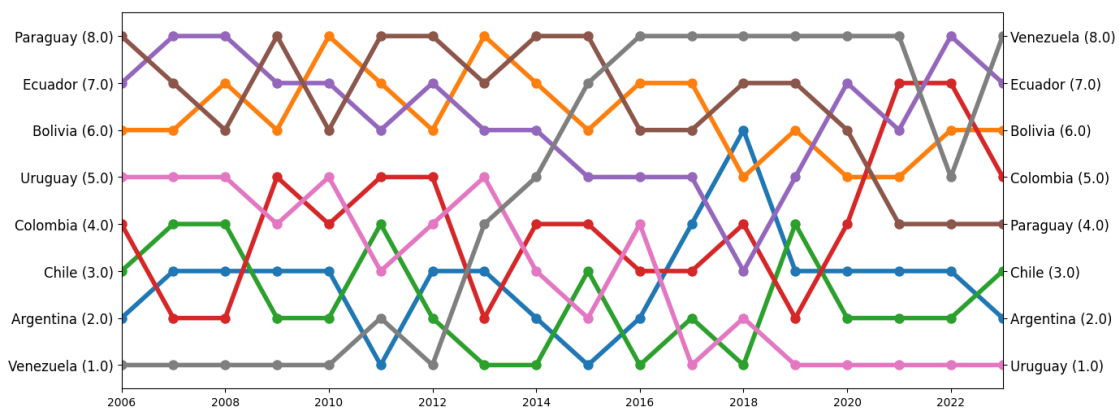


Al crear el primero Bumpchart, podemos ver que en el lado derecho aparece un eje Y extra el cual eliminaremos. Además, pondremos los números de ranking correspondiente a cada uno de los países

[47]:
```
def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()
```

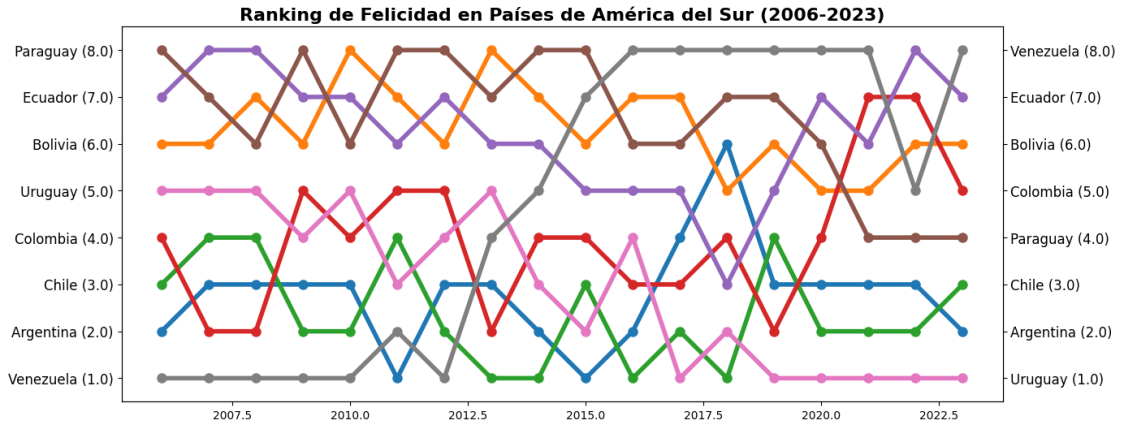
```
    # Crear el tercer eje si show_rank_axis es True
    if show_rank_axis:
        far_right_yaxis = left_yaxis.twinx()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

    for idx, col in enumerate(df_pivot_clean.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

        # Añadir scatter si es necesario
        if scatter:
            left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')

    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    for axis in [left_yaxis, right_yaxis]:
        axis.set_yticks(y_ticks)
        axis.set_ylim((lines + 0.5, 0.5))
        axis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    left_labels = [f"{df_ranking.iloc[0].sort_values().index[i]} ({df_ranking.
↪iloc[0].sort_values().values[i]})" for i in range(len(df_ranking.iloc[0]))]
    right_labels = [f"{df_ranking.iloc[-1].sort_values().index[i]} ({df_ranking.
↪iloc[-1].sort_values().values[i]})" for i in range(len(df_ranking.iloc[-1]))]

    left_yaxis.set_yticklabels(left_labels, fontsize=12)
    right_yaxis.set_yticklabels(right_labels, fontsize=12)

    # Configurar el tercer eje si es necesario
    if show_rank_axis:
        far_right_yaxis.spines["right"].set_position(("axes",␣
↪rank_axis_distance))
        far_right_yaxis.set_yticks(y_ticks)
        far_right_yaxis.set_yticklabels(right_labels, fontsize=12)
        far_right_yaxis.set_ylim((lines + 0.5, 0.5))
        far_right_yaxis.invert_yaxis()
```

```
    # Eliminar el tercer eje
    if show_rank_axis:
        far_right_yaxis.set_visible(False)

    # Ajustar el eje x
    ax.set_xlim([2006, 2023])

    # Ocultar todos los ax.spines
    for spine in ax.spines.values():
        spine.set_visible(False)

    # Añadir título
    plt.title('Ranking de Felicidad en Países de América del Sur (2006-2023)',
              fontsize=16, weight='bold')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```



[48]:
```
def bumpchart(df_ranking, show_rank_axis=False, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()
```

```
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

    for idx, col in enumerate(df_pivot_clean.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

        # Añadir scatter si es necesario
        if scatter:
            left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')

    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    for axis in [left_yaxis, right_yaxis]:
        axis.set_yticks(y_ticks)
        axis.set_ylim((lines + 0.5, 0.5))
        axis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    left_labels = [f"{label} ({df_ranking.iloc[0].sort_values().get(label)})"␣
↪for label in df_ranking.iloc[0].sort_values().index]
    right_labels = [f"{label} ({df_ranking.iloc[-1].sort_values().get(label)})"␣
↪for label in df_ranking.iloc[-1].sort_values().index]

    left_yaxis.set_yticklabels(left_labels, fontsize=12)
    right_yaxis.set_yticklabels(right_labels, fontsize=12)

    # Añadir título
    plt.title('Ranking de Felicidad en Países de América del Sur (2006-2023)',
              fontsize=16, weight='bold')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```

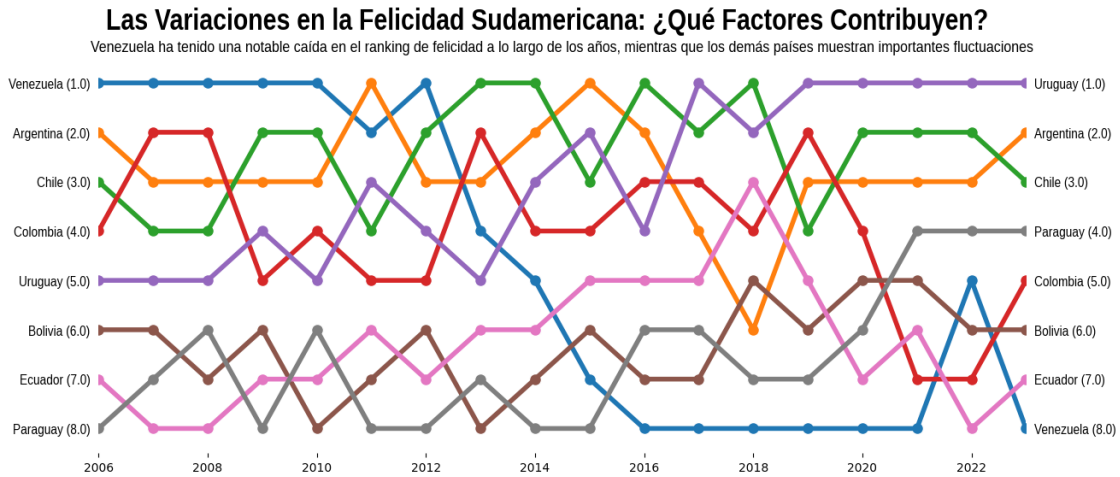


Una vez eliminado el eje y extra y añadido los valores del ranking correspondiente a cada uno de los países mejoraremos aún más la gráfica. Pondremos un mejor título y subtítlo, además ajustaremos el tipo de letra y tamaño para darle un mejor aspecto.

[49]:
```
def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

    for idx, col in enumerate(df_ranking.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

        # Añadir scatter si es necesario
        if scatter:
            left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')
```

```
    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    # Establecer las posiciones y etiquetas de los ticks en el eje izquierdo
    left_yaxis.set_yticks(y_ticks)
    left_labels = [f"{df_ranking.iloc[0].sort_values().index[i]} ({df_ranking.
↪iloc[0].sort_values().values[i]})" for i in range(len(df_ranking.iloc[0]))]
    left_yaxis.set_yticklabels(left_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    left_yaxis.set_ylim((lines + 0.5, 0.5))
    #left_yaxis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    # Establecer las posiciones y etiquetas de los ticks en el eje derecho
    right_yaxis.set_yticks(y_ticks)
    right_labels = [f"{df_ranking.iloc[-1].sort_values().index[i]} ({df_ranking.
↪iloc[-1].sort_values().values[i]})" for i in range(len(df_ranking.iloc[-1]))]
    right_yaxis.set_yticklabels(right_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    right_yaxis.set_ylim((lines + 0.5, 0.5))
    #right_yaxis.invert_yaxis()

    # Eliminar los ticks del eje y
    left_yaxis.tick_params(left=False)
    right_yaxis.tick_params(right=False)

    # Ocultar las líneas de los ejes (spines)
    for axis in [left_yaxis, right_yaxis]:
        for spine in axis.spines.values():
            spine.set_visible(False)

    # Ajustar el eje x
    ax.set_xlim([2006, 2023])

    # Añadir título
    plt.title('Venezuela ha tenido una notable caída en el ranking de felicidad␣
↪a lo largo de los años, mientras que los demás países muestran importantes␣
↪fluctuaciones',
              fontsize=14, fontfamily='serif', fontname= 'Liberation Sans␣
↪Narrow')
    plt.suptitle("Las Variaciones en la Felicidad Sudamericana: ¿Qué Factores␣
↪Contribuyen?", fontsize=25, weight='bold', fontfamily='serif', fontname=␣
↪'Liberation Sans Narrow')

    return ax
```

```
# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```

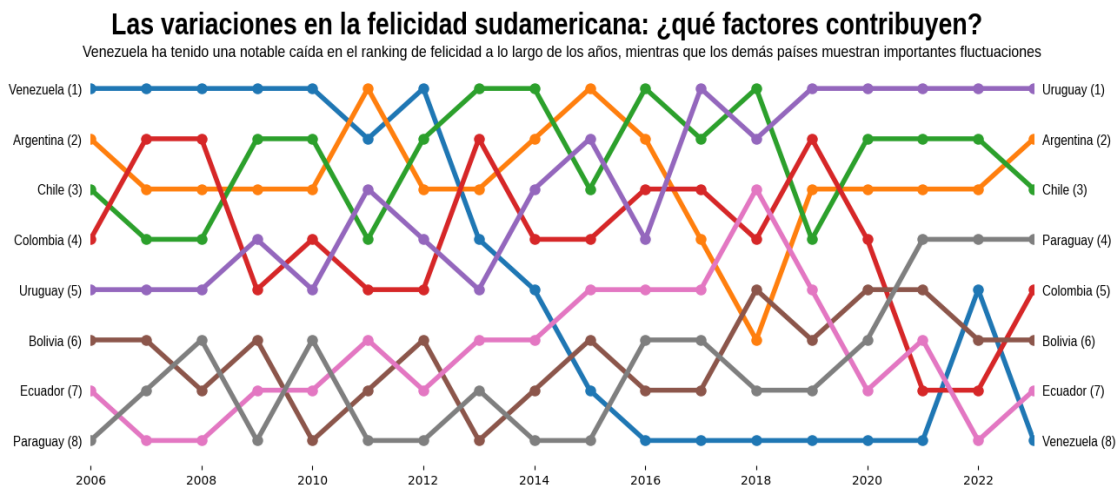


Por último para mejorar un poco más la visualización, transformaremos los valores en las etiquetas de los Ejes y, a enteros.

[50]:
```
def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

    for idx, col in enumerate(df_ranking.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

        # Añadir scatter si es necesario
```

```
        if scatter:
            left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')

    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    # Establecer las posiciones y etiquetas de los ticks en el eje izquierdo
    left_yaxis.set_yticks(y_ticks)
    left_labels = [f"{df_ranking.iloc[0].sort_values().index[i]}␣
↪({int(df_ranking.iloc[0].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[0]))]
    left_yaxis.set_yticklabels(left_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    left_yaxis.set_ylim((lines + 0.5, 0.5))
    #left_yaxis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    # Establecer las posiciones y etiquetas de los ticks en el eje derecho
    right_yaxis.set_yticks(y_ticks)
    right_labels = [f"{df_ranking.iloc[-1].sort_values().index[i]}␣
↪({int(df_ranking.iloc[-1].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[-1]))]
    right_yaxis.set_yticklabels(right_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    right_yaxis.set_ylim((lines + 0.5, 0.5))
    #right_yaxis.invert_yaxis()

    # Eliminar los ticks del eje y
    left_yaxis.tick_params(left=False)
    right_yaxis.tick_params(right=False)

    # Ocultar las líneas de los ejes (spines)
    for axis in [left_yaxis, right_yaxis]:
        for spine in axis.spines.values():
            spine.set_visible(False)

    # Ajustar el eje x
    ax.set_xlim([2006, 2023])

    # Añadir título
    plt.title('Venezuela ha tenido una notable caída en el ranking de felicidad␣
↪a lo largo de los años, mientras que los demás países muestran importantes␣
↪fluctuaciones',
              fontsize=14, fontfamily='serif', fontname= 'Liberation Sans␣
↪Narrow')
```

```
    plt.suptitle("Las variaciones en la felicidad sudamericana: ¿qué factores␣
↪contribuyen?", fontsize=25, weight='bold', fontfamily='serif', fontname=␣
↪'Liberation Sans Narrow')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```



1.1 Ensayo explicativo

1.1.1 Gráficos a usar en el ensayo

Mapa

[51]:
```
# Definir nueva paleta de colores divergente invertida
colores_divergentes = ['#4394BC', '#92B9C8', '#BFD7DE', '#F3F3F3', '#E8A696',␣
↪'#D68C78', '#C57554']
colores_divergentes_invertidos = colores_divergentes[::-1]  # Invertir la lista␣
↪de colores
```

[52]:
```
# # Crear la paleta de colores personalizada a partir de la lista de colores␣
↪invertida
# mi_paleta_divergente = LinearSegmentedColormap.
↪from_list("mi_paleta_divergente", colores_divergentes_invertidos, N=256)

# # Cargar el archivo Shapefile
# shapefile_path = r"/content/drive/MyDrive/Datasets_para_COLAB/
↪Tarea_4_Visualización/world-administrative-boundaries/
↪world-administrative-boundaries.shp"
```

```
# world = gpd.read_file(shapefile_path)

# # Diccionario para mapear los nombres del CSV a los nombres del Shapefile
# mapeo_nombres = {
#     'Bosnia and Herzegovina': 'Bosnia & Herzegovina',
#     'Congo (Brazzaville)': 'Congo',
#     'Congo (Kinshasa)': 'Democratic Republic of the Congo',
#     'Czechia': 'Czech Republic',
#     'Eswatini': 'Swaziland',
#     'Hong Kong S.A.R. of China': 'Hong Kong',
#     'Iran': 'Iran (Islamic Republic of)',
#     'Ivory Coast': "Côte d'Ivoire",
#     'Kosovo': 'Serbia',
#     'Laos': "Lao People's Democratic Republic",
#     'Libya': 'Libyan Arab Jamahiriya',
#     'Moldova': 'Moldova, Republic of',
#     'North Macedonia': 'The former Yugoslav Republic of Macedonia',
#     'Russia': 'Russian Federation',
#     'Somaliland region': 'Somalia',
#     'South Korea': 'Republic of Korea',
#     'State of Palestine': ['West Bank', 'Gaza Strip'],
#     'Syria': 'Syrian Arab Republic',
#     'Taiwan Province of China': 'Taiwan',
#     'Tanzania': 'United Republic of Tanzania',
#     'Türkiye': 'Turkey',
#     'United Kingdom': 'U.K. of Great Britain and Northern Ireland',
#     'United States': 'United States of America'
# }

# # Función para aplicar el mapeo
# def aplicar_mapeo(nombre):
#     if nombre == 'State of Palestine':
#         return ['West Bank', 'Gaza Strip']
#     return [mapeo_nombres.get(nombre, nombre)]

# # Aplicar el mapeo a la columna 'Country name'
# df_excel['Country name'] = df_excel['Country name'].apply(lambda x:␣
 ↪aplicar_mapeo(x))

# # Aplanar la lista de nombres de países
# df_excel_exploded = df_excel.explode('Country name')

# # Unir los datos de "Life Ladder" con los datos geográficos por el nombre del␣
 ↪país
# world = world.merge(df_excel_exploded, how='left', left_on='name',␣
 ↪right_on='Country name')
```

```
# # Crear el mapa temático utilizando la paleta de colores invertida
# fig, ax = plt.subplots(1, 1, figsize=(12, 10))
# world.plot(column='Life Ladder', cmap=mi_paleta_divergente, ax=ax,
#            missing_kwds={"color": "lightgrey", "label": "No data"})

# # ----- Incorporar Gradiente de Colores sobre Heatmap ----

# # Crear un eje separado para la barra de color en la parte superior derecha
# cbar_ax = fig.add_axes([0.75, 0.63, 0.1, 0.01])  # [left, bottom, width,␣
↪height]

# # Generar la barra de color en el eje separado
# sm = plt.cm.ScalarMappable(cmap=mi_paleta_divergente, norm=plt.
↪Normalize(vmin=1, vmax=9))
# cbar = fig.colorbar(sm, cax=cbar_ax, orientation='horizontal', ticks=[1.0, 9.
↪0])

# # Eliminar el contorno de la barra de color
# cbar.outline.set_visible(False)

# # Mover los ticks más hacia el centro (interior) de la barra
# cbar.ax.tick_params(axis='x', which='both', direction='out')  # Ticks hacia␣
↪adentro de la barra

# # Eliminar los contornos grises alrededor de los países
# ax.set_axis_off()  # Elimina los bordes del gráfico para que no haya líneas

# # --- Agregar la línea horizontal antes del título ---
# y_min, y_max = ax.get_ylim()  # Obtener los límites actuales del eje Y
# x_min, x_max = ax.get_xlim()  # Obtener los límites actuales del eje X

# # Dibujar la línea horizontal en la parte superior
# ax.plot([x_min, x_max], [y_max + 60, y_max + 60], color="black", lw=1.5)  #␣
↪Línea horizontal

# # Título del gráfico
# ax.text(
#     x_min,
#     y_max+55,
#     'Felicidad en Colores: ¿Qué tan feliz es el mundo?',
#     ha='left',
#     va='center',
#     fontsize=12,
#     weight='bold'
# )
```

```
# ax.plot([x_min, x_min+10], [y_max + 45, y_max + 45], color="black", lw=1.5) ␣
↪# Línea horizontal

# # Subtítulo del gráfico
# ax.text(
#     x_min,
#     y_max+40,
#     'Los países en azul disfrutan de vidas más plenas, mientras que los tonos␣
↪rojos reflejan mayores niveles de insatisfacción',
#     ha='left',
#     va='center',
#     fontsize=10,
# )

# # Subtítulo del gráfico
# ax.text(
#     x_max-63,
#     y_max+28,
#     'Escala de Bienestar',
#     ha='left',
#     va='center',
#     fontsize=8,
# )

# no_data_patch = mpatches.Patch(color='lightgrey', label='No data')
# plt.legend(handles=[no_data_patch], frameon=False, bbox_to_anchor=(-1.3,␣
↪-30), loc='upper left', prop={'size': 8})

# plt.show()
```

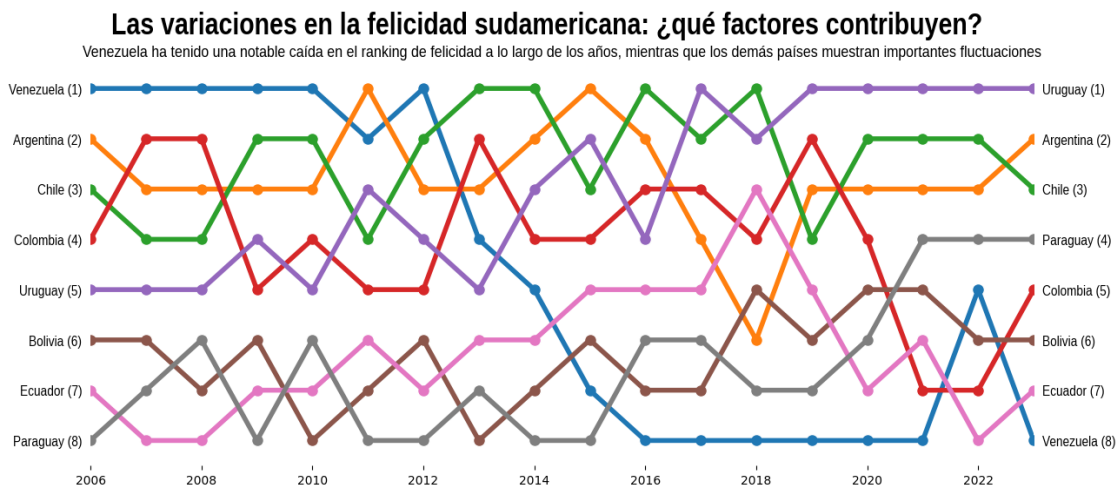
El código del mapa fue comentado porque surge un error si si código es corrido más de una vez.

Bumpchart de países sudamericanos

[53]:
```
def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(14, 6))  # Aumentamos el tamaño del␣
↪gráfico
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
```

```
            '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']  # Colores␣
↪personalizados

   for idx, col in enumerate(df_ranking.columns):
       y = df_ranking[col]
       x = df_ranking.index.values

       # Graficar en el eje izquierdo
       left_yaxis.plot(x, y, color=colors[idx], linewidth=4, **line_args)

       # Añadir scatter si es necesario
       if scatter:
           left_yaxis.scatter(x, y, color=colors[idx], s=100,␣
↪edgecolor='white')

   # Configuraciones para las etiquetas de los ejes
   lines = len(df_ranking.columns)
   y_ticks = [*range(1, lines + 1)]

   # Establecer las posiciones y etiquetas de los ticks en el eje izquierdo
   left_yaxis.set_yticks(y_ticks)
   left_labels = [f"{df_ranking.iloc[0].sort_values().index[i]}␣
↪({int(df_ranking.iloc[0].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[0]))]
   left_yaxis.set_yticklabels(left_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
   left_yaxis.set_ylim((lines + 0.5, 0.5))
   #left_yaxis.invert_yaxis()  # Invertir el eje para reflejar los rankings

   # Establecer las posiciones y etiquetas de los ticks en el eje derecho
   right_yaxis.set_yticks(y_ticks)
   right_labels = [f"{df_ranking.iloc[-1].sort_values().index[i]}␣
↪({int(df_ranking.iloc[-1].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[-1]))]
   right_yaxis.set_yticklabels(right_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
   right_yaxis.set_ylim((lines + 0.5, 0.5))
   #right_yaxis.invert_yaxis()

   # Eliminar los ticks del eje y
   left_yaxis.tick_params(left=False)
   right_yaxis.tick_params(right=False)

   # Ocultar las líneas de los ejes (spines)
   for axis in [left_yaxis, right_yaxis]:
       for spine in axis.spines.values():
```

```
            spine.set_visible(False)

    # Ajustar el eje x
    ax.set_xlim([2006, 2023])

    # Añadir título
    plt.title('Venezuela ha tenido una notable caída en el ranking de felicidad␣
↪a lo largo de los años, mientras que los demás países muestran importantes␣
↪fluctuaciones',
              fontsize=14, fontfamily='serif', fontname= 'Liberation Sans␣
↪Narrow')
    plt.suptitle("Las variaciones en la felicidad sudamericana: ¿qué factores␣
↪contribuyen?", fontsize=25, weight='bold', fontfamily='serif', fontname=␣
↪'Liberation Sans Narrow')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)
plt.show()
```



Heatmap

[54]:
```
# Previsualizaciones de Heatmap rotado

# Calcular matiz de correlación
correlation_matrix = df_numeric.corr(numeric_only=True)

# Ajustar tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 6))
```

```
ax0 = rotate_axes(fig, 111, -135)

# Dibujar el heatmap
sns.heatmap(correlation_matrix, annot=False, cmap=mi_paleta_divergente, ax=ax0,␣
 ↪linewidths=0.2, cbar=False)


# ---- Añadir etiquetas a los ejes del heatmap ----

plt.text(-7.4, -8.4, 'Life Ladder', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.8, -10.0, 'Log GDP per capita', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-6.4, -10.15, 'Social support', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.9, -13.0, 'Healthy life expectancy at birth', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-6.85, -13.45, 'Freedom to make life choices', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-3.85, -11.8, 'Generosity', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-4.9, -14.3, 'Perceptions of corruption', color = 'black', rotation =␣
 ↪45, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.8, -13.71, 'Positive affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.26, -14.6, 'Negative affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)

plt.text(0.36, -12.6, 'Negative affect', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(1.1, -11.88, 'Positive affect', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(1.7, -11.2, 'Perceptions of corruption', color = 'black', rotation =␣
 ↪315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(2.4, -10.45, 'Generosity', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.15, -9.8, 'Freedom to make life choices', color = 'black', rotation␣
 ↪= 315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.87, -9.0, 'Healthy life expectancy at birth', color = 'black',␣
 ↪rotation = 315, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(4.6, -8.3, 'Social support', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
```

```
plt.text(5.28, -7.62, 'Log GDP per capita', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(5.95, -6.9, 'Life Ladder', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)


# ---- Añadir Titulo, Subtítulo, Líneas -----

# Título principal
plt.text(0.1, 1.2, "Análisis del Indicador de Satisfacción de Vida (Life␣
 ↪Ladder) en el mundo",
     fontsize=14, fontweight='bold', ha='left', transform=fig.transFigure)

# Insertar la línea sobre el título
fig.add_artist(
    plt.Line2D([0.1, 1.5], [1.24, 1.24], lw=1, color="black",␣
 ↪solid_capstyle="butt", transform=fig.transFigure)
)

# Subtítulo
plt.text(x=0.1, y=1.1, s="¿Cómo impactan los indicadores de bienestar sobre la␣
 ↪Satisfacción de Vida?",
         fontsize=11, fontweight='normal', ha='left', transform=fig.transFigure)

# Insertar Línea corta sobre el subtítulo
fig.add_artist(
    plt.Line2D([0.1, 0.26], [1.14, 1.14], lw=0.5, color="black",␣
 ↪solid_capstyle="butt", transform=fig.transFigure)
)

# ----- Incorporar Gradiente de Colores sobre Heatmap ----

# Crear un eje separado para la barra de color en la parte superior derecha
cbar_ax = fig.add_axes([0.18, 1.0, 0.25, 0.025])  # [left, bottom, width,␣
 ↪height]

# Generar la barra de color en el eje separado
sm = plt.cm.ScalarMappable(cmap=mi_paleta_divergente, norm=plt.
 ↪Normalize(vmin=-1, vmax=1))
cbar = fig.colorbar(sm, cax=cbar_ax, orientation='horizontal', ticks=[-1.0,0.0,␣
 ↪1.0])

# Eliminar el contorno de la barra de color
cbar.outline.set_visible(False)

# Mover los ticks más hacia el centro (interior) de la barra
```

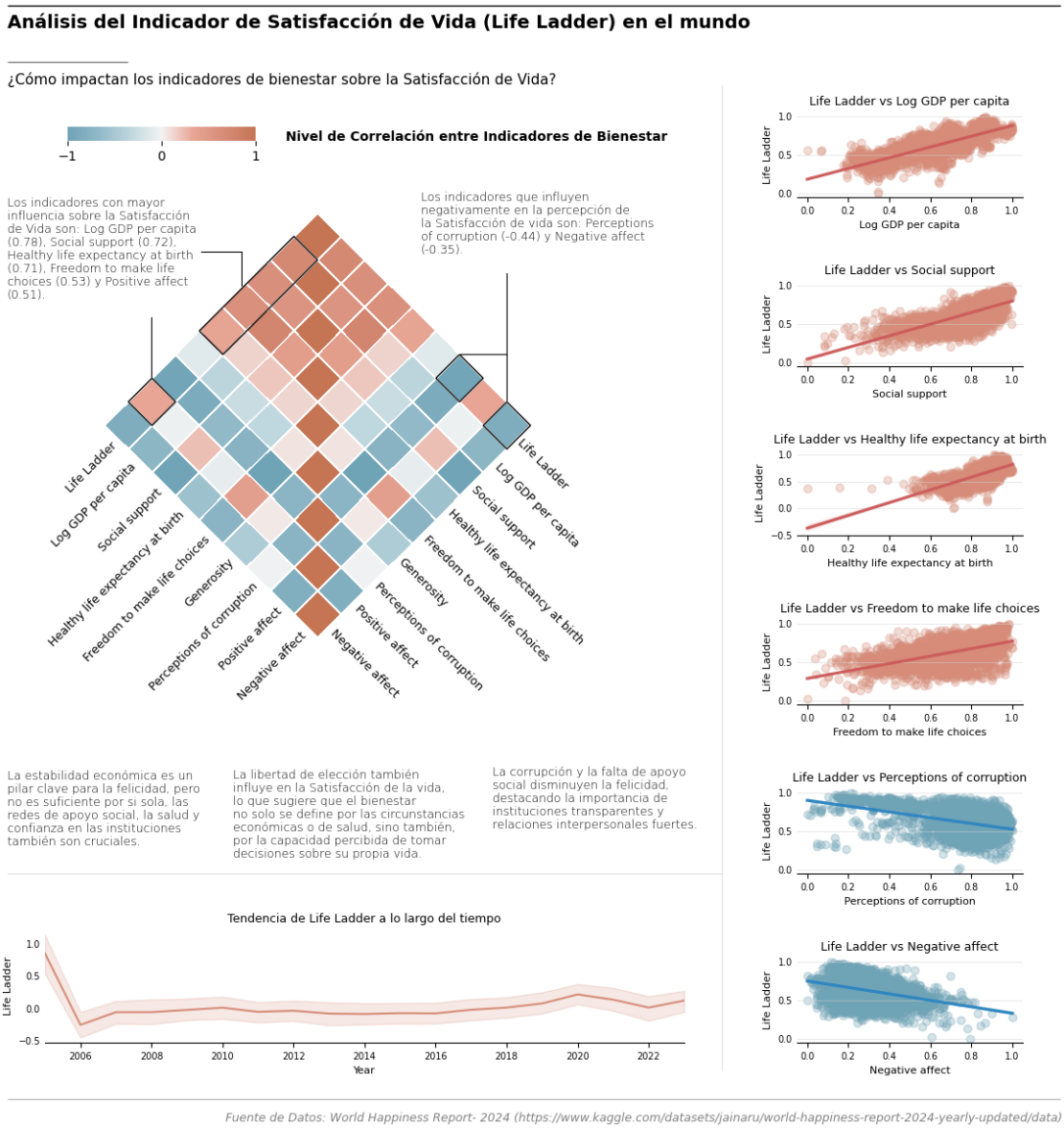
```
cbar.ax.tick_params(axis='x', which='both', direction='out')  # Ticks hacia␣
↪adentro de la barra

# Añadir título de la barra de color
plt.text(0.47, 1.0, 'Nivel de Correlación entre Indicadores de Bienestar ',␣
↪fontsize=10, fontweight='bold', ha='left', transform=fig.transFigure)  #␣
↪Negrita


# ---- Añadir Gráficos de Regresión encima del Heatmap ----

# Gráfico 1: Life Ladder vs Log GDP per capita
reg_ax1 = fig.add_axes([1.15, 0.9, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Log GDP per capita', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Log GDP per capita', fontsize=9)
reg_ax1.set_xlabel('Log GDP per capita', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 2: Life Ladder vs Social support
reg_ax1 = fig.add_axes([1.15, 0.6, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Social support', y='Life Ladder', data=df, scatter_kws={'alpha':␣
↪0.3, 'color': '#D68C78', 'marker': 'o'}, line_kws={'color': '#CD5C5C'},␣
↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Social support', fontsize=9)
reg_ax1.set_xlabel('Social support', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)
```

```
# Gráfico 3: Life Ladder vs Healthy life expectancy at birth
reg_ax1 = fig.add_axes([1.15, 0.3, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Healthy life expectancy at birth', y='Life Ladder', data=df,␣
 ↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
 ↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Healthy life expectancy at birth', fontsize=9)
reg_ax1.set_xlabel('Healthy life expectancy at birth', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 4: Life Ladder vs Freedom to make life choices
reg_ax1 = fig.add_axes([1.15, 0.0, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Freedom to make life choices', y='Life Ladder', data=df,␣
 ↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
 ↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Freedom to make life choices', fontsize=9)
reg_ax1.set_xlabel('Freedom to make life choices', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 5: Life Ladder vs Perceptions of corruption
reg_ax1 = fig.add_axes([1.15, -0.3, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Perceptions of corruption', y='Life Ladder', data=df,␣
 ↪scatter_kws={'alpha': 0.3, 'color': '#6FA3B6', 'marker': 'o'},␣
 ↪line_kws={'color': '#2e86c1'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Perceptions of corruption', fontsize=9)
reg_ax1.set_xlabel('Perceptions of corruption', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
```

```
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 6: Life Ladder vs Negative affect
reg_ax1 = fig.add_axes([1.15, -0.6, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Negative affect', y='Life Ladder', data=df, scatter_kws={'alpha':
 ↪ 0.3, 'color': '#6FA3B6', 'marker': 'o'}, line_kws={'color': '#2e86c1'},␣
 ↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Negative affect', fontsize=9)
reg_ax1.set_xlabel('Negative affect', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)


# ---- Añadir gráfico de lineas para representar tendencia de Life Ladder a lo␣
 ↪largo de los años ----

min_year = df_cleaned['year'].min()
max_year = df_cleaned['year'].max()
reg_ax1 = fig.add_axes([0.15, -0.6, 0.85, 0.2])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.lineplot(x='year', y='Life Ladder', data=df_cleaned, ax=reg_ax1,␣
 ↪color='#D68C78')
reg_ax1.set_xlim(min_year, max_year)
reg_ax1.set_title('Tendencia de Life Ladder a lo largo del tiempo', fontsize=9)
reg_ax1.set_xlabel('Year', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
```

```
reg_ax1.spines['bottom'].set_visible(True)

# ---- Añadir rectángulos y líneas para destacar información ----

# Lado izquierdo
ax0.add_patch(Rectangle((1, 0), 4, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((7, 0), 1, 1, fill=None, alpha=1))
ax.axvline(0.37, 0.9, 0.7, color = 'black', linewidth=0.8)
ax.axvline(0.215, 0.75, 0.6, color = 'black', linewidth=0.8)
ax.axhline(0.9, 0.3, 0.37, color = 'black', linewidth=0.8 )

# Lado derecho
ax0.add_patch(Rectangle((0, 6), 1, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((0, 8), 1, 1, fill=None, alpha=1))
ax.axvline(0.824, 0.85, 0.55, color = 'black', linewidth=0.8)
ax.axhline(0.665, 0.74, 0.824, color = 'black', linewidth=0.8 )

# ---- Texto descriptivo Heatmap ----

plt.text(0.1, 0.72, 'Los indicadores con mayor \ninfluencia sobre la␣
 ↪Satisfacción \nde Vida son: Log GDP per capita \n(0.78), Social support (0.
 ↪72), \nHealthy life expectancy at birth \n(0.71), Freedom to make life␣
 ↪\nchoices (0.53) y Positive affect \n(0.51).', fontsize=9, fontweight=False,␣
 ↪ha='left', transform=fig.transFigure)  # Negrita
plt.text(0.65, 0.8, 'Los indicadores que influyen \nnegativamente en la␣
 ↪percepción de \nla Satisfacción de vida son: Perceptions \nof corruption (-0.
 ↪44) y Negative affect \n(-0.35).', fontsize=9, fontweight=False, ha='left',␣
 ↪transform=fig.transFigure)  # Negrita

# ---- Texto conslusiones Heatmap ----

plt.text(0.1, -0.25, 'La estabilidad económica es un \npilar clave para la␣
 ↪felicidad, pero \nno es suficiente por si sola, las \nredes de apoyo social,␣
 ↪la salud y \nconfianza en las instituciones \ntambién son cruciales.',␣
 ↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
 ↪Negrita
plt.text(0.4, -0.272, 'La libertad de elección también \ninfluye en la␣
 ↪Satisfacción de la vida, \nlo que sugiere que el bienestar \nno solo se␣
 ↪define por las circunstancias\neconómicas o de salud, sino también, \npor la␣
 ↪capacidad percibida de tomar \ndecisiones sobre su propia vida.',␣
 ↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
 ↪Negrita
plt.text(0.745, -0.22, 'La corrupción y la falta de apoyo \nsocial disminuyen␣
 ↪la felicidad, \ndestacando la importancia de \ninstituciones transparentes y␣
 ↪\nrelaciones interpersonales fuertes.', fontsize=9, fontweight=False,␣
 ↪ha='left', transform=fig.transFigure)  # Negrita
```

```python
# ---- Ajustes linea inferior ----

plt.text(x=0.39, y=-0.74, s="Fuente de Datos: World Happiness Report- 2024␣
↪(https://www.kaggle.com/datasets/jainaru/
↪world-happiness-report-2024-yearly-updated/data)",
        fontsize=9, color='gray', style='italic', ha='left', transform=fig.
↪transFigure)

# Insertar una línea sobre datos del pie de página
fig.add_artist(plt.Line2D([0.1, 1.5], [- 0.7, -0.7], lw=0.5, color="grey",␣
↪solid_capstyle="butt", transform=fig.transFigure))


# ---- Lineas divisorias  --

fig.add_artist(plt.Line2D([1.05, 1.05], [- 0.65, 1.1], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))
fig.add_artist(plt.Line2D([0.1, 1.05], [- 0.3, -0.3], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))


# ----- Configuraciones Extras ------

ax.set_axis_off()

# Mostrar gráfico
plt.show()
```



1.2 Visualización Final

[55]:
```
# ... (Your existing code for map, heatmap, and bumpchart) ...


# Assuming you have your map, heatmap, and bumpchart code defined as:
# - map_fig, map_ax
# - heatmap_fig, heatmap_ax
# - bumpchart_fig, bumpchart_ax
```

```
# Create a new figure and axes for the combined plot
fig = plt.figure(figsize=(16, 12))  # Adjust figure size as needed


# Add the map to the top
# gs = fig.add_gridspec(2, 2)
# ax_map = fig.add_subplot(gs[0, :])
ax.axis('off')
# ... (Your map plotting code here using ax_map) ...

# Crear la paleta de colores personalizada a partir de la lista de colores␣
 ↪invertida
mi_paleta_divergente = LinearSegmentedColormap.
 ↪from_list("mi_paleta_divergente", colores_divergentes_invertidos, N=256)

# Cargar el archivo Shapefile
shapefile_path = r"world-administrative-boundaries/
 ↪world-administrative-boundaries.shp"
world = gpd.read_file(shapefile_path)

# Diccionario para mapear los nombres del CSV a los nombres del Shapefile
mapeo_nombres = {
    'Bosnia and Herzegovina': 'Bosnia & Herzegovina',
    'Congo (Brazzaville)': 'Congo',
    'Congo (Kinshasa)': 'Democratic Republic of the Congo',
    'Czechia': 'Czech Republic',
    'Eswatini': 'Swaziland',
    'Hong Kong S.A.R. of China': 'Hong Kong',
    'Iran': 'Iran (Islamic Republic of)',
    'Ivory Coast': "Côte d'Ivoire",
    'Kosovo': 'Serbia',
    'Laos': "Lao People's Democratic Republic",
    'Libya': 'Libyan Arab Jamahiriya',
    'Moldova': 'Moldova, Republic of',
    'North Macedonia': 'The former Yugoslav Republic of Macedonia',
    'Russia': 'Russian Federation',
    'Somaliland region': 'Somalia',
    'South Korea': 'Republic of Korea',
    'State of Palestine': ['West Bank', 'Gaza Strip'],
    'Syria': 'Syrian Arab Republic',
    'Taiwan Province of China': 'Taiwan',
    'Tanzania': 'United Republic of Tanzania',
    'Türkiye': 'Turkey',
    'United Kingdom': 'U.K. of Great Britain and Northern Ireland',
    'United States': 'United States of America'
}
```

```python
# Función para aplicar el mapeo
def aplicar_mapeo(nombre):
    if nombre == 'State of Palestine':
        return ['West Bank', 'Gaza Strip']
    return [mapeo_nombres.get(nombre, nombre)]

# Aplicar el mapeo a la columna 'Country name'
df_excel['Country name'] = df_excel['Country name'].apply(lambda x:␣
 ↪aplicar_mapeo(x))

# Aplanar la lista de nombres de países
df_excel_exploded = df_excel.explode('Country name')

# Unir los datos de "Life Ladder" con los datos geográficos por el nombre del␣
 ↪país
world = world.merge(df_excel_exploded, how='left', left_on='name',␣
 ↪right_on='Country name')

# Crear el mapa temático utilizando la paleta de colores invertida
fig, ax = plt.subplots(1, 1, figsize=(12, 10))
world.plot(column='Life Ladder', cmap=mi_paleta_divergente, ax=ax,
           missing_kwds={"color": "lightgrey", "label": "No data"})

# ----- Incorporar Gradiente de Colores sobre Heatmap ----

# Crear un eje separado para la barra de color en la parte superior derecha
cbar_ax = fig.add_axes([0.75, 0.63, 0.1, 0.01])  # [left, bottom, width, height]

# Generar la barra de color en el eje separado
sm = plt.cm.ScalarMappable(cmap=mi_paleta_divergente, norm=plt.
 ↪Normalize(vmin=1, vmax=9))
cbar = fig.colorbar(sm, cax=cbar_ax, orientation='horizontal', ticks=[1.0, 9.0])

# Eliminar el contorno de la barra de color
cbar.outline.set_visible(False)

# Mover los ticks más hacia el centro (interior) de la barra
cbar.ax.tick_params(axis='x', which='both', direction='out')  # Ticks hacia␣
 ↪adentro de la barra

# Eliminar los contornos grises alrededor de los países
ax.set_axis_off()  # Elimina los bordes del gráfico para que no haya líneas

# --- Agregar la línea horizontal antes del título ---
y_min, y_max = ax.get_ylim()  # Obtener los límites actuales del eje Y
x_min, x_max = ax.get_xlim()  # Obtener los límites actuales del eje X
```

```
# Dibujar la línea horizontal en la parte superior
ax.plot([x_min, x_max], [y_max + 60, y_max + 60], color="black", lw=1.5)  #␣
 ↪Línea horizontal

# Título del gráfico
ax.text(
    x_min,
    y_max+55,
    'Felicidad en colores: ¿Qué tan feliz es el mundo?',
    ha='left',
    va='center',
    fontsize=12,
    weight='bold'
)

ax.plot([x_min, x_min+10], [y_max + 45, y_max + 45], color="black", lw=1.5)  #␣
 ↪Línea horizontal

# Subtítulo del gráfico
ax.text(
    x_min,
    y_max+40,
    'Los países en azul disfrutan de vidas más plenas, mientras que los tonos␣
 ↪rojos reflejan mayores niveles de insatisfacción',
    ha='left',
    va='center',
    fontsize=10,
)

# Subtítulo del gráfico
ax.text(
    x_max-63,
    y_max+28,
    'Escala de Bienestar',
    ha='left',
    va='center',
    fontsize=8,
)

no_data_patch = mpatches.Patch(color='lightgrey', label='No data')
plt.legend(handles=[no_data_patch], frameon=False, bbox_to_anchor=(-1.3, -30),␣
 ↪loc='upper left', prop={'size': 8})

ax.set_axis_off()

# Add the heatmap and bumpchart side by side below the map
# ax_heatmap = fig.add_subplot(gs[1, 0])
```

```python
# ... (Your heatmap plotting code here using ax_heatmap) ...

# Previsualizaciones de Heatmap rotado

# Calcular matiz de correlación
correlation_matrix = df_numeric.corr(numeric_only=True)

# Ajustar tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 6))
ax0 = rotate_axes(fig, 111, -135)

# Dibujar el heatmap
sns.heatmap(correlation_matrix, annot=False, cmap=mi_paleta_divergente, ax=ax0,␣
 ↪linewidths=0.2, cbar=False)


# ---- Añadir etiquetas a los ejes del heatmap ----

plt.text(-7.4, -8.4, 'Life Ladder', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.8, -10.0, 'Log GDP per capita', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-6.4, -10.15, 'Social support', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-7.9, -13.0, 'Healthy life expectancy at birth', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-6.85, -13.45, 'Freedom to make life choices', color = 'black',␣
 ↪rotation = 45, rotation_mode = 'anchor', transform_rotates_text = True,␣
 ↪fontsize=9)
plt.text(-3.85, -11.8, 'Generosity', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-4.9, -14.3, 'Perceptions of corruption', color = 'black', rotation =␣
 ↪45, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.8, -13.71, 'Positive affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(-2.26, -14.6, 'Negative affect', color = 'black', rotation = 45,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)

plt.text(0.36, -12.6, 'Negative affect', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(1.1, -11.88, 'Positive affect', color = 'black', rotation = 315,␣
 ↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(1.7, -11.2, 'Perceptions of corruption', color = 'black', rotation =␣
 ↪315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
```

```
plt.text(2.4, -10.45, 'Generosity', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.15, -9.8, 'Freedom to make life choices', color = 'black', rotation␣
↪= 315, rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(3.87, -9.0, 'Healthy life expectancy at birth', color = 'black',␣
↪rotation = 315, rotation_mode = 'anchor', transform_rotates_text = True,␣
↪fontsize=9)
plt.text(4.6, -8.3, 'Social support', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(5.28, -7.62, 'Log GDP per capita', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)
plt.text(5.95, -6.9, 'Life Ladder', color = 'black', rotation = 315,␣
↪rotation_mode = 'anchor', transform_rotates_text = True, fontsize=9)


# ---- Añadir Titulo, Subtítulo, Líneas -----

# Título principal
plt.text(0.1, 1.2, "Análisis del Indicador de Satisfacción de Vida (Life␣
↪Ladder) en el mundo",
    fontsize=14, fontweight='bold', ha='left', transform=fig.transFigure)

# Insertar la línea sobre el título
# fig.add_artist(
#     plt.Line2D([0.1, 1.5], [1.24, 1.24], lw=1, color="black",␣
↪solid_capstyle="butt", transform=fig.transFigure)
# )

# Subtítulo
plt.text(x=0.1, y=1.1, s="¿Cómo impactan los indicadores de bienestar sobre la␣
↪Satisfacción de Vida?",
        fontsize=11, fontweight='normal', ha='left', transform=fig.transFigure)

# Insertar Línea corta sobre el subtítulo
fig.add_artist(
    plt.Line2D([0.1, 0.26], [1.14, 1.14], lw=0.5, color="black",␣
↪solid_capstyle="butt", transform=fig.transFigure)
)

# ----- Incorporar Gradiente de Colores sobre Heatmap ----

# Crear un eje separado para la barra de color en la parte superior derecha
cbar_ax = fig.add_axes([0.18, 1.0, 0.25, 0.025])  # [left, bottom, width,␣
↪height]

# Generar la barra de color en el eje separado
```

```
sm = plt.cm.ScalarMappable(cmap=mi_paleta_divergente, norm=plt.
 ↪Normalize(vmin=-1, vmax=1))
cbar = fig.colorbar(sm, cax=cbar_ax, orientation='horizontal', ticks=[-1.0,0.0,␣
 ↪1.0])

# Eliminar el contorno de la barra de color
cbar.outline.set_visible(False)

# Mover los ticks más hacia el centro (interior) de la barra
cbar.ax.tick_params(axis='x', which='both', direction='out')  # Ticks hacia␣
 ↪adentro de la barra

# Añadir título de la barra de color
plt.text(0.47, 1.0, 'Nivel de Correlación entre Indicadores de Bienestar ',␣
 ↪fontsize=10, fontweight='bold', ha='left', transform=fig.transFigure)  #␣
 ↪Negrita


# ---- Añadir Gráficos de Regresión encima del Heatmap ----

# Gráfico 1: Life Ladder vs Log GDP per capita
reg_ax1 = fig.add_axes([1.15, 0.9, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Log GDP per capita', y='Life Ladder', data=df,␣
 ↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
 ↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Log GDP per capita', fontsize=9)
reg_ax1.set_xlabel('Log GDP per capita', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
 ↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 2: Life Ladder vs Social support
reg_ax1 = fig.add_axes([1.15, 0.6, 0.3, 0.15])  # Posicionar gráfico en␣
 ↪coordenadas específicas
sns.regplot(x='Social support', y='Life Ladder', data=df, scatter_kws={'alpha':␣
 ↪0.3, 'color': '#D68C78', 'marker': 'o'}, line_kws={'color': '#CD5C5C'},␣
 ↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Social support', fontsize=9)
reg_ax1.set_xlabel('Social support', fontsize=8)
```

```
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 3: Life Ladder vs Healthy life expectancy at birth
reg_ax1 = fig.add_axes([1.15, 0.3, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Healthy life expectancy at birth', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Healthy life expectancy at birth', fontsize=9)
reg_ax1.set_xlabel('Healthy life expectancy at birth', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 4: Life Ladder vs Freedom to make life choices
reg_ax1 = fig.add_axes([1.15, 0.0, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Freedom to make life choices', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#D68C78', 'marker': 'o'},␣
↪line_kws={'color': '#CD5C5C'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Freedom to make life choices', fontsize=9)
reg_ax1.set_xlabel('Freedom to make life choices', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)
```

```
# Gráfico 5: Life Ladder vs Perceptions of corruption
reg_ax1 = fig.add_axes([1.15, -0.3, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Perceptions of corruption', y='Life Ladder', data=df,␣
↪scatter_kws={'alpha': 0.3, 'color': '#6FA3B6', 'marker': 'o'},␣
↪line_kws={'color': '#2e86c1'}, ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Perceptions of corruption', fontsize=9)
reg_ax1.set_xlabel('Perceptions of corruption', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# Gráfico 6: Life Ladder vs Negative affect
reg_ax1 = fig.add_axes([1.15, -0.6, 0.3, 0.15])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.regplot(x='Negative affect', y='Life Ladder', data=df, scatter_kws={'alpha':
↪ 0.3, 'color': '#6FA3B6', 'marker': 'o'}, line_kws={'color': '#2e86c1'},␣
↪ax=reg_ax1)
reg_ax1.set_title('Life Ladder vs Negative affect', fontsize=9)
reg_ax1.set_xlabel('Negative affect', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.grid(True, which='both', axis='y', linestyle='-', linewidth=0.5,␣
↪color='lightgray', alpha=0.7)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)


# ---- Añadir gráfico de lineas para representar tendencia de Life Ladder a lo␣
↪largo de los años ----

min_year = df_cleaned['year'].min()
max_year = df_cleaned['year'].max()
reg_ax1 = fig.add_axes([0.15, -0.6, 0.85, 0.2])  # Posicionar gráfico en␣
↪coordenadas específicas
sns.lineplot(x='year', y='Life Ladder', data=df_cleaned, ax=reg_ax1,␣
↪color='#D68C78')
```

```
reg_ax1.set_xlim(min_year, max_year)
reg_ax1.set_title('Tendencia de Life Ladder a lo largo del tiempo', fontsize=9)
reg_ax1.set_xlabel('Year', fontsize=8)
reg_ax1.set_ylabel('Life Ladder', fontsize=8)
reg_ax1.tick_params(labelsize=7)
reg_ax1.tick_params(axis='y', length=0)
reg_ax1.spines['top'].set_visible(False)
reg_ax1.spines['right'].set_visible(False)
reg_ax1.spines['left'].set_visible(False)
reg_ax1.spines['bottom'].set_visible(True)

# ---- Añadir rectángulos y líneas para destacar información ----

# Lado izquierdo
ax0.add_patch(Rectangle((1, 0), 4, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((7, 0), 1, 1, fill=None, alpha=1))
ax.axvline(0.37, 0.9, 0.7, color = 'black', linewidth=0.8)
ax.axvline(0.215, 0.75, 0.6, color = 'black', linewidth=0.8)
ax.axhline(0.9, 0.3, 0.37, color = 'black', linewidth=0.8 )

# Lado derecho
ax0.add_patch(Rectangle((0, 6), 1, 1, fill=None, alpha=1))
ax0.add_patch(Rectangle((0, 8), 1, 1, fill=None, alpha=1))
ax.axvline(0.824, 0.85, 0.55, color = 'black', linewidth=0.8)
ax.axhline(0.665, 0.74, 0.824, color = 'black', linewidth=0.8 )

# ---- Texto descriptivo Heatmap ----

plt.text(0.1, 0.72, 'Los indicadores con mayor \ninfluencia sobre la␣
↪Satisfacción \nde Vida son: Log GDP per capita \n(0.78), Social support (0.
↪72), \nHealthy life expectancy at birth \n(0.71), Freedom to make life␣
↪\nchoices (0.53) y Positive affect \n(0.51).', fontsize=9, fontweight=False,␣
↪ha='left', transform=fig.transFigure)  # Negrita
plt.text(0.65, 0.8, 'Los indicadores que influyen \nnegativamente en la␣
↪percepción de \nla Satisfacción de vida son: Perceptions \nof corruption (-0.
↪44) y Negative affect \n(-0.35).', fontsize=9, fontweight=False, ha='left',␣
↪transform=fig.transFigure)  # Negrita

# ---- Texto conclusiones Heatmap ----

plt.text(0.1, -0.25, 'La estabilidad económica es un \npilar clave para la␣
↪felicidad, pero \nno es suficiente por si sola, las \nredes de apoyo social,␣
↪la salud y \nconfianza en las instituciones \ntambién son cruciales.',␣
↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
↪Negrita
```

```
plt.text(0.4, -0.272, 'La libertad de elección también \ninfluye en la␣
↪Satisfacción de la vida, \nlo que sugiere que el bienestar \nno solo se␣
↪define por las circunstancias\neconómicas o de salud, sino también, \npor la␣
↪capacidad percibida de tomar \ndecisiones sobre su propia vida.',␣
↪fontsize=9, fontweight=False, ha='left', transform=fig.transFigure)  #␣
↪Negrita
plt.text(0.745, -0.22, 'La corrupción y la falta de apoyo \nsocial disminuyen␣
↪la felicidad, \ndestacando la importancia de \ninstituciones transparentes y␣
↪\nrelaciones interpersonales fuertes.', fontsize=9, fontweight=False,␣
↪ha='left', transform=fig.transFigure)  # Negrita

# ---- Ajustes línea inferior ----

# plt.text(x=0.39, y=-0.74, s="Fuente de Datos: World Happiness Report- 2024␣
↪(https://www.kaggle.com/datasets/jainaru/
↪world-happiness-report-2024-yearly-updated/data)",
        #  fontsize=9, color='gray', style='italic', ha='left', transform=fig.
↪transFigure)

# Insertar una línea sobre datos del pie de página
# fig.add_artist(plt.Line2D([0.1, 1.5], [- 0.7, -0.7], lw=0.5, color="grey",␣
↪solid_capstyle="butt", transform=fig.transFigure))


# ---- Líneas divisorias  --

fig.add_artist(plt.Line2D([1.05, 1.05], [- 0.65, 1.1], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))
fig.add_artist(plt.Line2D([0.1, 1.05], [- 0.3, -0.3], lw=0.5,␣
↪color="lightgray", solid_capstyle="butt", transform=fig.transFigure))


# ----- Configuraciones Extras ------

ax.set_axis_off()

# ax_bumpchart = fig.add_subplot(gs[1, 1])
# ... (Your bumpchart plotting code here using ax_bumpchart) ...

def bumpchart(df_ranking, show_rank_axis=True, rank_axis_distance=1.1,
              ax=None, scatter=True, holes=False,
              line_args={}, scatter_args={}, hole_args={}):

    if ax is None:
        fig, ax = plt.subplots(figsize=(11, 4))  # Aumentamos el tamaño del␣
↪gráfico
```

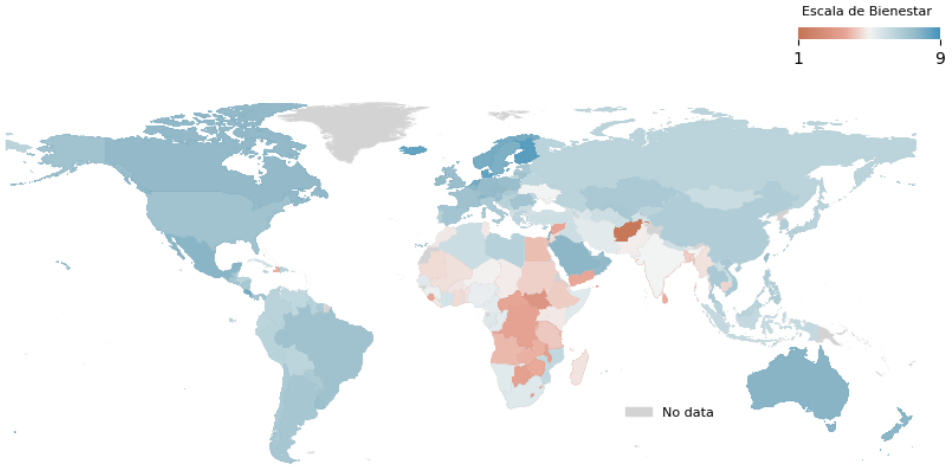
```
    left_yaxis = ax

    # Crear un segundo eje a la derecha
    right_yaxis = left_yaxis.twinx()

    # Definir una paleta de colores más suaves
    colors = ['#aec7e8', '#ffbb78', '#98df8a', '#ff9896', '#c5b0d5', '#c49c94',
              '#f7b6d2', '#dbdb8d', '#9edae5', '#7f7f7f']

    # Resaltar Venezuela con un color más intenso
    venezuela_color = '#d73027'

    for idx, col in enumerate(df_ranking.columns):
        y = df_ranking[col]
        x = df_ranking.index.values

        color = colors[idx]
        if col == 'Venezuela':
            color = venezuela_color

        # Graficar en el eje izquierdo
        left_yaxis.plot(x, y, color=color, linewidth=4, **line_args)

        # Añadir scatter si es necesario
        if scatter:
            left_yaxis.scatter(x, y, color=color, s=100, edgecolor='white')

    # Configuraciones para las etiquetas de los ejes
    lines = len(df_ranking.columns)
    y_ticks = [*range(1, lines + 1)]

    # Establecer las posiciones y etiquetas de los ticks en el eje izquierdo
    left_yaxis.set_yticks(y_ticks)
    left_labels = [f"{df_ranking.iloc[0].sort_values().index[i]}␣
↪({int(df_ranking.iloc[0].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[0]))]
    left_yaxis.set_yticklabels(left_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    left_yaxis.set_ylim((lines + 0.5, 0.5))
    #left_yaxis.invert_yaxis()  # Invertir el eje para reflejar los rankings

    # Establecer las posiciones y etiquetas de los ticks en el eje derecho
    right_yaxis.set_yticks(y_ticks)
    right_labels = [f"{df_ranking.iloc[-1].sort_values().index[i]}␣
↪({int(df_ranking.iloc[-1].sort_values().values[i])})" for i in␣
↪range(len(df_ranking.iloc[-1]))]
```

```
    right_yaxis.set_yticklabels(right_labels, fontsize=12, fontfamily='serif',␣
↪fontname= 'Liberation Sans Narrow')
    right_yaxis.set_ylim((lines + 0.5, 0.5))
    #right_yaxis.invert_yaxis()

    # Eliminar los ticks del eje y
    left_yaxis.tick_params(left=False)
    right_yaxis.tick_params(right=False)

    # Ocultar las líneas de los ejes (spines)
    for axis in [left_yaxis, right_yaxis]:
        for spine in axis.spines.values():
            spine.set_visible(False)

    # Ajustar el eje x
    ax.set_xlim([2006, 2023])

    # Añadir título
    plt.title('Venezuela ha tenido una notable caída en el ranking de felicidad␣
↪a lo largo de los años, mientras que los demás países muestran importantes␣
↪fluctuaciones',
              fontsize=12, fontfamily='serif', fontname= 'Liberation Sans␣
↪Narrow')
    plt.suptitle("Las variaciones en la felicidad sudamericana: ¿Qué factores␣
↪contribuyen?", fontsize=14, weight='bold', fontfamily='serif', fontname=␣
↪'Liberation Sans Narrow')

    return ax

# Llamada a la función con los datos limpios
bumpchart(df_ranking)

# Adjust layout and spacing as needed
plt.tight_layout()
plt.show()
```

```
<Figure size 1600x1200 with 0 Axes>
```

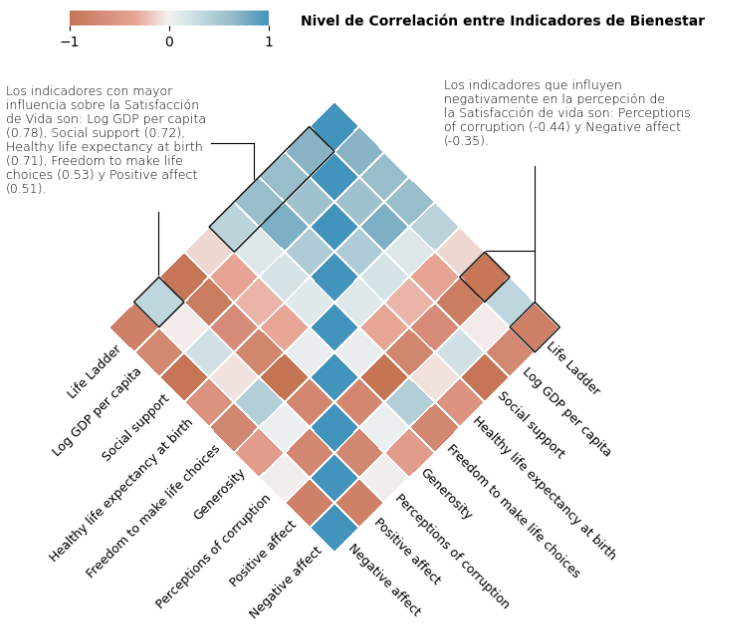
Felicidad en colores: ¿Qué tan feliz es el mundo?

Los países en azul disfrutan de vidas más plenas, mientras que los tonos rojos reflejan mayores niveles de insatisfacción



Análisis del Indicador de Satisfacción de Vida (Life Ladder) en el mundo

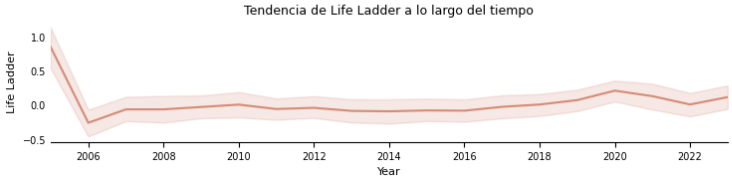
¿Cómo impactan los indicadores de bienestar sobre la Satisfacción de Vida?

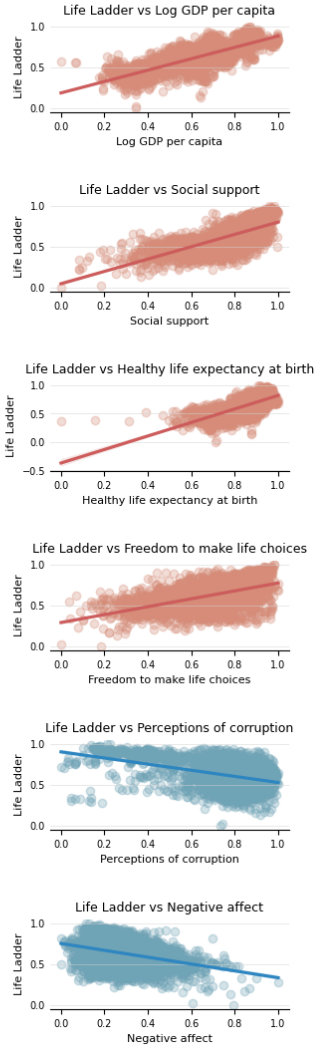


La estabilidad económica es un pilar clave para la felicidad, pero no es suficiente por si sola, las redes de apoyo social, la salud y confianza en las instituciones también son cruciales.

La libertad de elección también influye en la Satisfacción de la vida, lo que sugiere que el bienestar no solo se define por las circunstancias económicas o de salud, sino también, por la capacidad percibida de tomar decisiones sobre su propia vida.

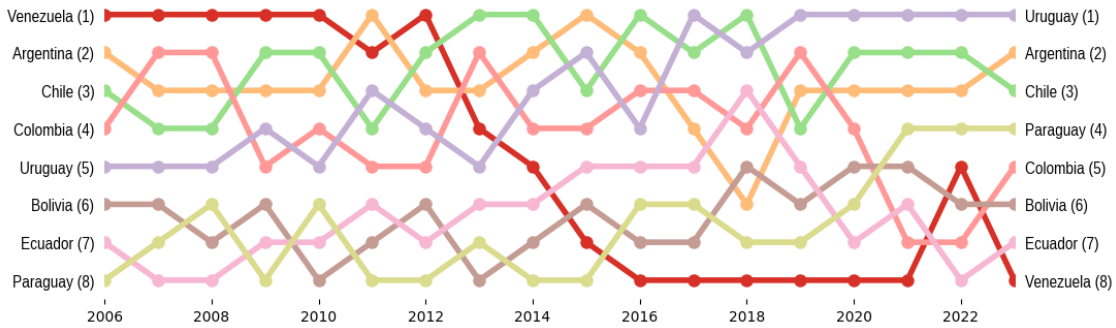
La corrupción y la falta de apoyo social disminuyen la felicidad, destacando la importancia de instituciones transparentes y relaciones interpersonales fuertes.





Las variaciones en la felicidad sudamericana: ¿Qué factores contribuyen?

Venezuela ha tenido una notable caída en el ranking de felicidad a lo largo de los años, mientras que los demás países muestran importantes fluctuaciones



En nuestro ensayo visual en su versión final nos enfocaremos primero en analizar los patrones de felicidad a nivel mundial. A nivel general, Oceanía presenta valores parejos de felicidad. Los niveles de fecilidad más bajos se observan en el continente africano, mientras que en Asia se verifica que la felicidad en Afganistán es muy baja. Un patrón similar se observa en Yemen, Siria y Sri Lanka. Los niveles de bienestar en Norteamérica en promedio son mayores que en Sudamérica. Destaca negativamente el nivel de bienestar de Haití, el cual es incluso inferior al de Venezuela. En Europa se comprueba que los países nórdicos presentan mayores niveles de bienestar en general.

Considerando la diversidad de niveles de bienestar en el mundo, cabe preguntarse sobre los factores que influyen positiva y negativamente. El PIB per cápita, el apoyo social, la esperanza de vida, la libertad de elección a nivel personal y la cantidad de experiencias positivas mejoran la percepción de bienestar de la población de un país. En cambio, la percepción de vivir en un país corrupto. contribuyendo esto a los sentimientos de tristeza y desesperanza disminuyen el bienestar de las personas. Los sentimientos negativos debieran ser mayores en los países con mayor pobreza, falta de apoyo social y sentir que no existe libertad de acción.

Para realizar un análisis más detallado evaluaremos la evolución de la percepción de bienestar en los países sudamericanos. Venezuela es un caso paradigmático de degradación de los niveles de bienestar, debido a que en el año 2006 era el país con mayor percepción de bienestar en Sudamérica, la cual disminuyó estrepitosamente desde el año 2012 debido a su crisis económica. Al año 2023 Uruguay es el país con mayor sensación de bienestar, considerando que en el año 2006 se encontraba en el quinto lugar a nivel continental. En el caso de Chile la percepción de bienestar también ha fluctuado entre los años 2006 y 2023, experimentando disminuciones durante la época de la crisis subprime (2007-2008), durante el año 2011 en el cual ocurrieron movilizaciones estudiantiles, año 2019 (estallido social) y año 2023 debido a las perspectivas económicas actuales.

Finalmente podemos concluir que el nivel de bienestar es un parámetro dinámico, claramente dependiente de los eventos sociales y económicos que transcurren en cada país.

[ ]:

```
```