

## Taller de visualización en Big Data

### Objetivo

Aplicar herramientas de visualizaciones estáticas y dinámicas para procesos de BigData, utilizando Python mediante librerías , Matplotlib, Seaborn y Plotly.

Desarrollar cada uno de los siguientes puntos y proponer 5 ejercicios adicionales (explicar resultados) de visualización con el DataSet de medición de contaminación de aire de en Seúl, Corea del Sur:

### Contexto del DataSet

Este conjunto de datos trata sobre la información de medición de la contaminación del aire en Seúl, Corea del Sur. El Gobierno Metropolitano de Seúl proporciona muchos datos públicos, incluida información sobre la contaminación del aire, a través de la '*Plaza de Datos Abiertos*' se creó un conjunto de datos estructurados recopilando y ajustando varios conjuntos de datos relacionados con la contaminación del aire proporcionados por el Gobierno Metropolitano de Seúl.

### Contenido

Estos datos proporcionan valores promedio para seis contaminantes (SO2, NO2, CO, O3, PM10, PM2.5).

- Los datos se midieron cada hora entre 2017 y 2019.
- Los datos se midieron para 25 distritos en Seúl.
- Este conjunto de datos se divide en cuatro archivos.
  1. Información de medición: información de medición de la contaminación del aire
    - Se proporciona una medición promedio de 1 hora después de la calibración
    - Estado del instrumento:
      - 0: Normal, 1: Necesidad de calibración, 2: Anormal
      - 4: corte de energía, 8: en reparación, 9: datos anormales
  2. Información del elemento de medición: información sobre los elementos de medición de la contaminación del aire
  3. Información de la estación de medición: información sobre las estaciones de instrumentos de contaminación del aire
  4. Resumen de medición: un conjunto de datos condensado basado en los tres datos anteriores.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import matplotlib.ticker as ticker
```

```
1 df = pd.read_csv('AirPollutionSeoul/Measurement_summary.csv')
2 df.head()
```

	Measurement date	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5
0	2017-01-01 00:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.059	0.002	1.2	73.0	57.0
1	2017-01-01 01:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.058	0.002	1.2	71.0	59.0
2	2017-01-01 02:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70.0	59.0
3	2017-01-01 03:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70.0	58.0
4	2017-01-01 04:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.051	0.002	1.2	69.0	61.0

```
1 df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 647511 entries, 0 to 647510
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   Measurement date  647511 non-null   object 
 1   Station code      647511 non-null   int64  
 2   Address           647511 non-null   object 
 3   Latitude          647511 non-null   float64
 4   Longitude         647511 non-null   float64
 5   SO2               647511 non-null   float64
 6   NO2               647511 non-null   float64
 7   O3                647511 non-null   float64
 8   CO                647511 non-null   float64
 9   PM10              647511 non-null   float64
 10  PM2.5             647511 non-null   float64
dtypes: float64(8), int64(1), object(2)
memory usage: 54.3+ MB
```

```
1 df['Station code'].nunique()
```

25

```
1 list_scode = list(set(df['Station code']))
2 list_scode|
```

```
[101,
 102,
 103,
 104.
```

## Preprocesar datos

Del 101 al 125, los códigos de estación representan los distritos de Seúl. Personalmente, usar los nombres de los distritos es más conveniente para etiquetar la visualización, ya que es más conveniente para leer. Los nombres se extraerán de la columna "Dirección" para crear la columna "Distrito".

```
In [6]: 1 list_add = list(df['Address'])
2 District = [i.split(',')[-2] for i in list_add]
3 df['District'] = District
```

Cree una lista con los nombres de los 25 distritos para usarla más adelante.

```
In [7]: 1 list_district = list(set(District))
```

Se crea otras tres columnas, YM (Año-Mes), Año y Mes para aplicar con algunos gráficos. Para facilitar la visualización, los agruparemos en DataFrame mensual promedio.

```
In [8]: 1
2 list_YM = [i.split(" ")[0][-3:] for i in list(df['Measurement date'])]
3 list_Year = [i.split(" ")[0][0:4] for i in list(df['Measurement date'])]
4 list_Month = [i.split(" ")[0][5:7] for i in list(df['Measurement date'])]
5
6 df['YM'] = list_YM
7 df['Year'] = list_Year
8 df['Month'] = list_Month
9
10 #create a monthly dataframe
11 df_monthly = df.groupby(['Station code', 'District', 'YM', 'Year', 'Month']).mean()
12 df_monthly = df_monthly[['SO2', 'NO2', 'O3', 'CO', 'PM10', 'PM2.5']].reset_index()
13
14 df_monthly.head()
```

Out[8]:

	Station code	District	YM	Year	Month	SO2	NO2	O3	CO	PM10	PM2.5
0	101	Jongno-gu	2017-01	2017	01	0.004401	0.037481	0.014972	0.695968	51.024194	35.118280
1	101	Jongno-gu	2017-02	2017	02	-0.022152	0.010290	-0.008138	0.600000	41.970238	28.857143
2	101	Jongno-gu	2017-03	2017	03	0.005015	0.041267	0.026862	0.698118	55.146505	40.311828
3	101	Jongno-gu	2017-04	2017	04	0.003308	0.034699	0.033512	0.598056	50.769444	26.536111
4	101	Jongno-gu	2017-05	2017	05	0.003461	0.031724	0.039319	0.490591	55.129032	22.680108

## Plot de datos

Gráfico de series temporales múltiples.

```
In [9]: 1 sns.set_style('darkgrid')
2 sns.set(rc={'figure.figsize':(14,8)})
3
4 ax = sns.lineplot(data=df_monthly, x ='YM', y = 'PM2.5',
5                     hue='District', palette='viridis',
6                     legend='full', lw=3)
7
8 ax.xaxis.set_major_locator(ticker.MultipleLocator(4))
9 plt.legend(bbox_to_anchor=(1, 1))
10 plt.ylabel('PM2.5 (\u00b5g/m\u00b3)')
11 plt.xlabel('Year-Month')
12 plt.show()
```

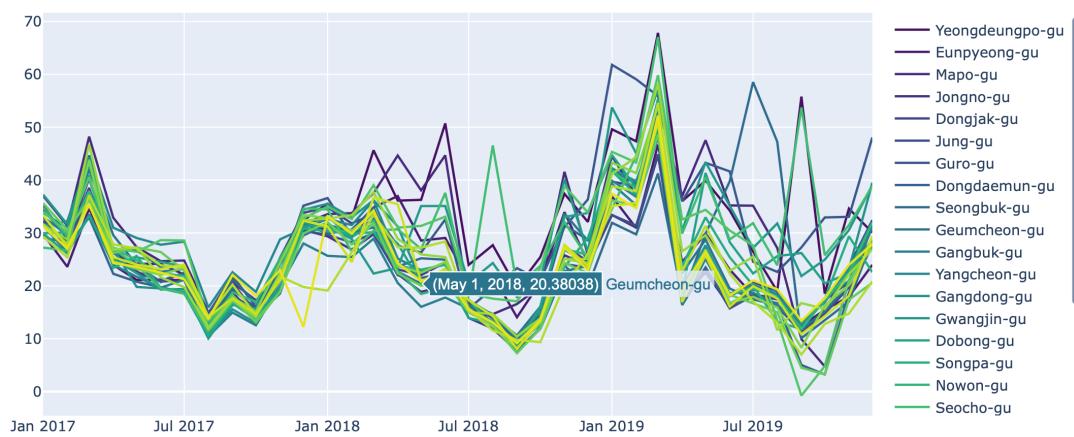


Las líneas superpuestas son difíciles de leer. En 2017, se puede ver que la cantidad de PM2.5 en muchas estaciones fue en la misma dirección. Sin embargo, en 2018 y 2019, las líneas de contaminación fueron arbitrarias y es difícil distinguirlas.

## Grafico Interactivo

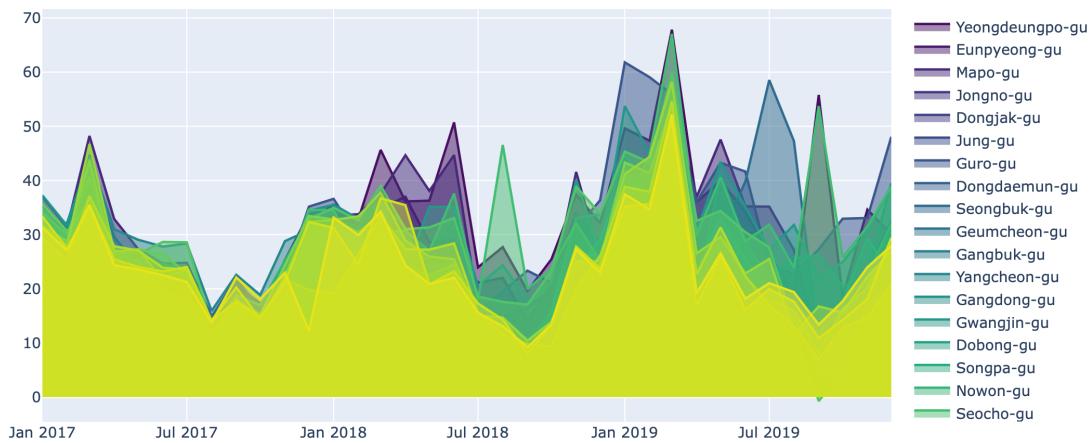
Plotly es una biblioteca gráfica para hacer gráficos interactivos. El gráfico interactivo ayuda a acercar el área con líneas superpuestas.

```
1 import plotly.graph_objects as go
2
3 #extract color palette, the palette can be changed
4 pal = list(sns.color_palette(palette='viridis', n_colors=len(list_scode)).as_hex())
5
6 fig = go.Figure()
7 for d,p in zip(list_district, pal):
8     fig.add_trace(go.Scatter(x = df_monthly[df_monthly['District']==d]['YM'],
9                               y = df_monthly[df_monthly['District']==d]['PM2.5'],
10                              name = d,
11                              line_color = p,
12                              fill=None)) #tozeroy
13
14 fig.show()
```



Con Plotly, también se puede hacer un gráfico de área interactivo.

```
1 fig = go.Figure()
2 for d,p in zip(list_district, pal):
3     fig.add_trace(go.Scatter(x = df_monthly[df_monthly['District']==d]['YM'],
4                               y = df_monthly[df_monthly['District']==d]['PM2.5'],
5                               name = d,
6                               line_color = p,
7                               fill='tozeroy')) #tozeroxy
8
9 fig.show()
```



## Comparar uno a uno con Small Multiple Time Series.

Con la biblioteca Seaborn, podemos hacer pequeñas series temporales múltiples. La idea detrás de estos gráficos es trazar cada línea una por una mientras las compara con la silueta de las otras líneas. El código en el enlace del sitio web oficial está aquí.

## Cambiar el punto de vista con Facet Grid

FacetGrid de Seaborn se puede utilizar para crear cuadrículas de varias parcelas. En este caso, los atributos 'Mes' y 'Año' se configuran como filas y columnas, respectivamente. Desde otra perspectiva, los valores se pueden comparar simultáneamente mensualmente en vertical y anualmente en horizontal

```
In [15]: 1 g = sns.FacetGrid(df_monthly, col="Year", row="Month", height=4.2, aspect=1.9)
2 g = g.map(sns.barplot, 'District', 'PM2.5', palette='viridis', ci=None, order = list_district)
3
4 g.set_xticklabels(rotation = 90)
5 plt.show()
```



## Uso del color con mapa de calor

Un mapa de calor representa los datos en un gráfico bidimensional que muestra los valores en colores. Para manejar los datos de la serie temporal, podemos configurar los grupos en la vertical y la línea de tiempo en las dimensiones horizontales. La diferencia de color ayuda a distinguir entre grupos.

Pivolar el marco de datos

```

1 df_pivot = pd.pivot_table(df_monthly,
2                           values='PM2.5',
3                           index='District',
4                           columns='YM')
5 df_pivot

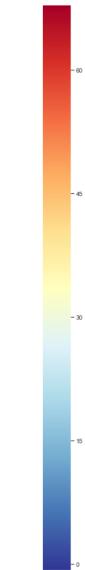
```

	YM	2017-01	2017-02	2017-03	2017-04	2017-05	2017-06	2017-07	2017-08	2017-09	2017-10	...	2019-03	2019-04
District														
Dobong-gu	29.938172	26.101190	33.245968	24.245833	25.389785	24.570833	23.674731	12.673387	17.268056	14.159946	...	55.300766	18.795549	41
Dongdaemun-gu	32.892473	27.238095	32.967742	22.276389	20.774194	19.733333	18.737903	11.646505	17.954167	14.250000	...	49.373563	16.368567	23
Dongjak-gu	34.952957	29.187500	37.931452	26.590278	23.169355	22.398611	23.310484	13.271505	20.622222	14.750000	...	53.065134	19.164117	26
Eunpyeong-gu	29.771505	23.595238	34.607527	24.941667	24.189516	24.125000	22.129032	12.572581	18.712500	14.106183	...	46.705545	24.499305	29
Gangbuk-gu	31.387097	26.311012	33.530914	23.969444	21.713710	19.647222	21.142473	10.440860	14.944444	12.534946	...	48.159004	16.389430	25
Gangdong-gu	33.797043	30.922619	42.892473	27.000000	19.787634	19.504167	18.595430	10.498656	15.556944	12.981183	...	50.542146	19.477051	27
Gwanak-gu	31.529570	28.763393	43.662634	24.642056	23.193548	22.076389	23.450269	12.126344	17.863889	15.145161	...	54.191571	19.324061	26

```

1 plt.figure(figsize = (40,19))
2 plt.title('Average Monthly PM2.5 (microgram/m3)')
3
4 sns.heatmap(df_pivot, annot=True, cmap='RdYlBu_r', fmt= '.4g')
5 plt.xlabel('Year-Month')
6 plt.ylabel('District')
7 plt.show()

```



Year-Month

District

## Aplicar ángulos con un gráfico de radar

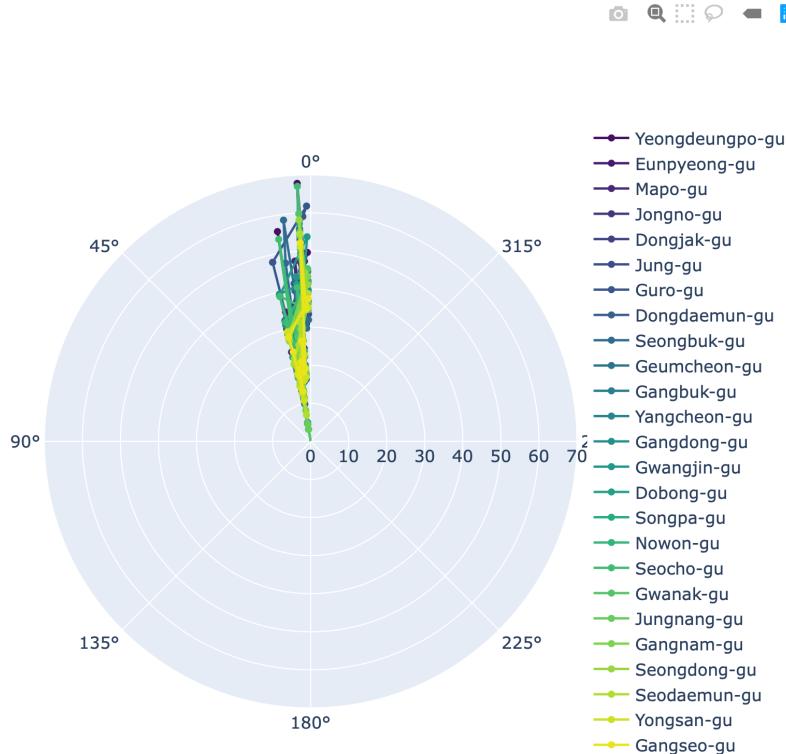
Podemos establecer el eje angular en el diagrama de dispersión en Plotly para crear un gráfico de radar interactivo. Cada mes se seleccionará como una variable en el círculo. Por ejemplo, en este artículo, crearemos un gráfico de radar que compare el promedio mensual de PM2.5 de los 25 distritos en 2019.

Filtre el DataFrame con solo datos de 2019

```
1 df_19 = df_monthly[df_monthly['Year']=='2019']
```

Crear gráfico de radar. Lo bueno de usar Plotly es que el gráfico de radar es interactivo. Entonces podemos filtrar fácilmente el gráfico.

```
1 import plotly.graph_objects as go
2
3 #extract color palette, the palette name can be changed
4 pal = list(sns.color_palette(palette='viridis', n_colors=len(list_scode)).as_hex())
5
6 months = list(reversed([str(i) for i in list(range(1,13))])) + ['12']
7 list_PM = [[list(df_19[df_19['District']==i]['PM2.5'])[int(n)-1] for n in months] for i in list_district]
8
9 fig = go.Figure()
10 for pm,d,c in zip(list_PM, list_district, pal):
11     fig.add_trace(go.Scatterpolar(r = pm, theta=months, fill= None,
12                                 name=str(d), marker = dict(color = c)))
13
14 fig.update_layout(polar = dict(radialaxis = dict(visible = True, range=[0, 70]),
15                         angularaxis = dict(rotation=90)),
16                     showlegend=True, width=720, height=720,
17                     font = dict(size=14))
18
19
20 fig.show()
```

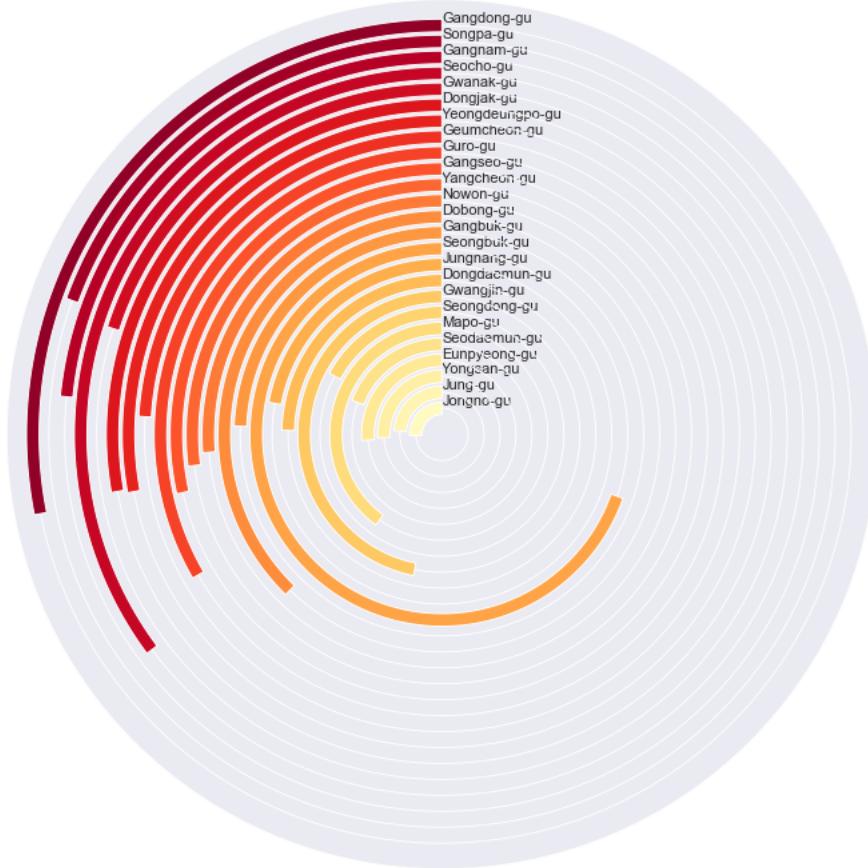


Vayamos más allá llenando el área de radar de cada uno de los distintos uno por uno y comparando cada uno con el resto. Luego crea un collage de fotos.

```
In [23]: 1 max_val = max(df_19['PM2.5'])*1.001           #set max value
2 pal = list(sns.color_palette(palette='YlOrRd', n_colors=len(list_district)).as_hex())
3
4 def circular_bar(input_df, column_name, title):
5     plt.gcf().set_size_inches(12, 12)
6     ax = plt.subplot(projection='polar')
7     input_df.reset_index(inplace=True, drop=True)
8     for i in range(len(input_df)):
9         ax.barh(i, input_df[column_name][i]*2*np.pi/max_val, label=input_df['District'][i], color=pal[i])
10
11     ax.set_theta_zero_location('N')
12     ax.set_theta_direction(1)
13     ax.set_rlabel_position(0)
14     ax.set_thetagrids([], labels[])
15     ax.set_rgrids(range(len(input_df)), labels= input_df['District'])
16
17     ax = plt.subplot(projection='polar')
18     plt.title("Average PM2.5 // " + title)
19
20     return ax
```

```
In [25]: 1 list_month19 = list(set(df_19['Month']))
2 list_YM19 = list(set(df_19['YM']))
3 list_YM19.sort()
4 listdf_monthly19 = [df_19[df_19['Month']==str(i)] for i in list_month19]
5
6 keep_sname = []
7 order = range(len(listdf_monthly19))
8 for i in order:
9     circular_bar(listdf_monthly19[i], 'PM2.5', list_YM19[i])
10    keep_sname.append('cir_bar_' + str(i) + '.png')
11    plt.savefig('cir_bar_' + str(i) + '.png')
12    plt.show()
```

Average PM2.5 // 2019-01

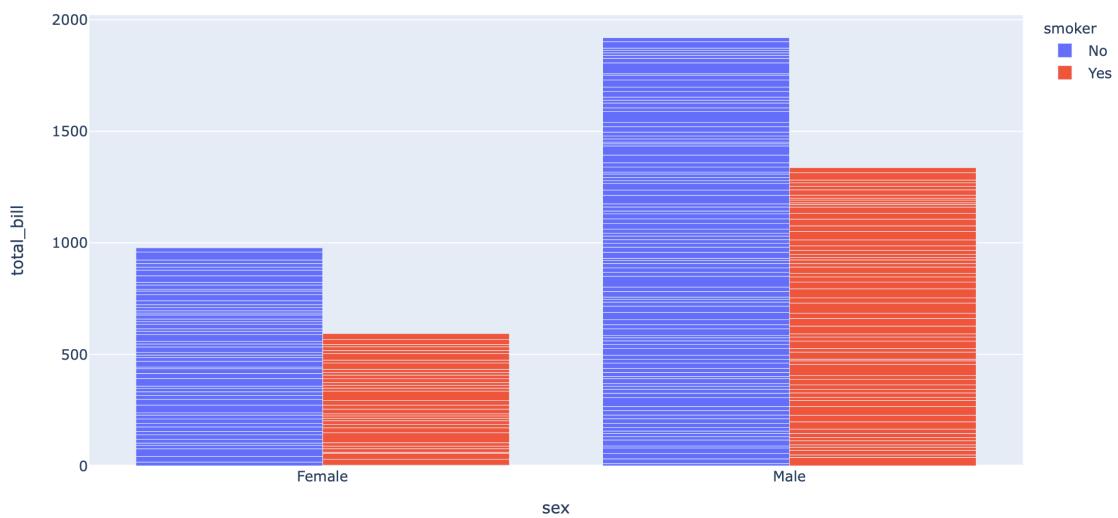


## Gráficos con mapas

### plotly.express

El módulo plotly.express (generalmente importado como px) contiene funciones que pueden crear figuras completas a la vez y se conoce como Plotly Express o PX. Plotly Express es una parte integrada de la biblioteca plotly y es el punto de partida recomendado para crear las figuras más comunes. Cada función de Plotly Express usa objetos gráficos internamente y devuelve una instancia de plotly.graph\_objects

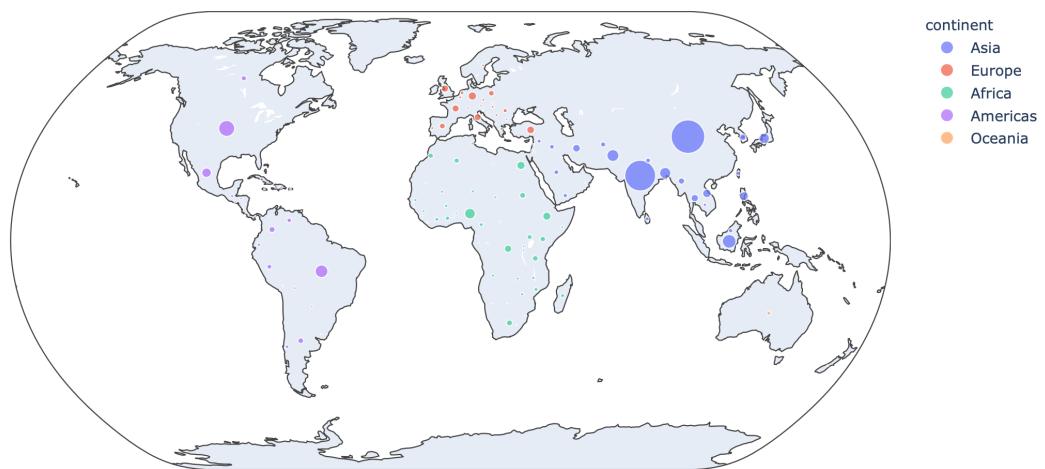
```
1 import plotly.express as px
2 df = px.data.tips()
3 fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group")
4 fig.show()
```



```

1 import plotly.express as px
2 df = px.data.gapminder().query("year==2007")
3 fig = px.scatter_geo(df, locations="iso_alpha", color="continent",
4                      hover_name="country", size="pop",
5                      projection="natural earth")
6 fig.show()
7

```



## Grafico Dinámico

```

1 df = px.data.gapminder()
2 fig = px.scatter_geo(df, locations="iso_alpha", color="continent",
3                      hover_name="country", size="pop",
4                      animation_frame="year",
5                      projection="natural earth")
6
7 fig.show()

```



```
1 import plotly.express as px
2
3 df = px.data.gapminder().query("year == 2007")
4 fig = px.sunburst(df, path=['continent', 'country'], values='pop',
5 color='lifeExp', hover_data=['iso_alpha'])
6 fig.show()
```

