2022년 가을학기

# Plotly Express

Day

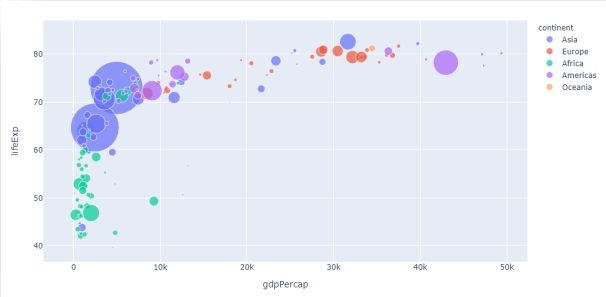
06

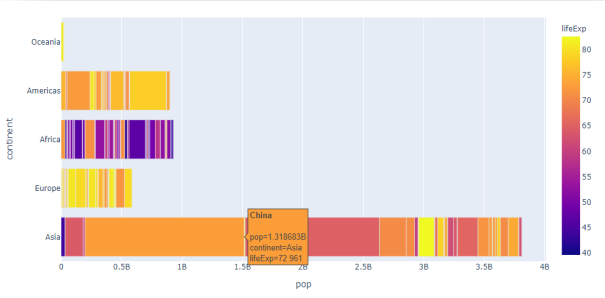
# 데이터시각화 – Plotly Express

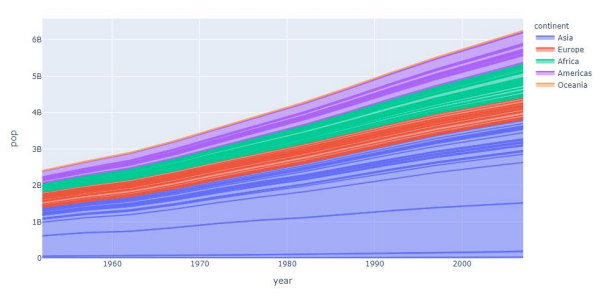
## CONTENTS

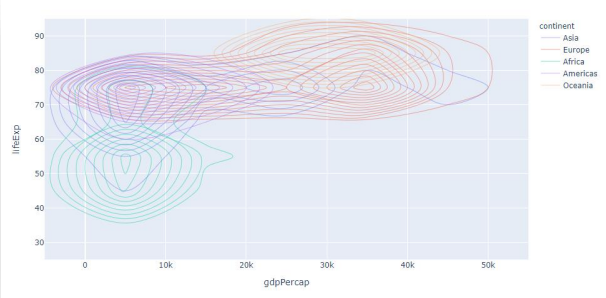
# Plotly Express

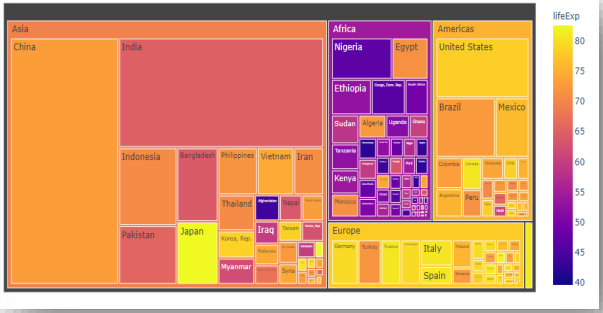
## ❖ Today's plots

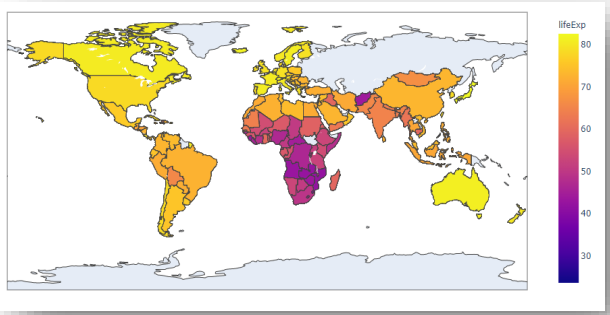


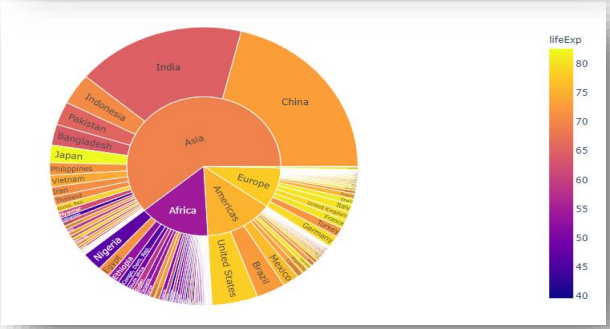



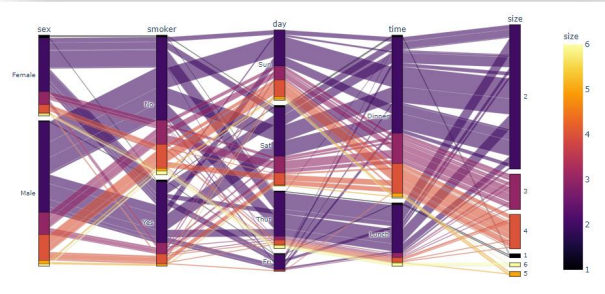



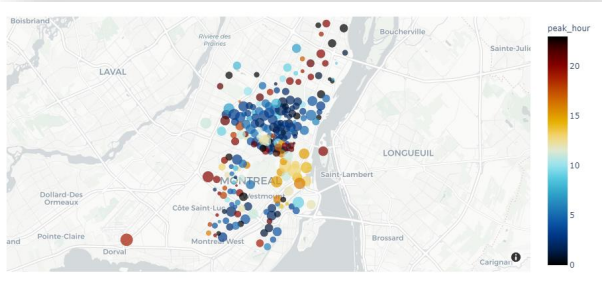



# Dataset

- **Plotly Express provides several types of datasets**
  - carshare()
  - election()
  - election_geojson()
  - experiment()
  - gapminder()
  - iris()
  - medals_long()
  - stocks()
  - tips()
  - wind()

- **Details of the datasets**
  - https://plotly.com/python-api-reference/generated/plotly.express.data.html

# Dataset

- **Plotly Express provides several types of datasets**
  - Gapminder dataset

```
import plotly.express as px

df = px.data.gapminder()

df.head()
```

| | country | continent | year | lifeExp | pop | gdpPercap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| 0 | Afghanistan | Asia | 1952 | 28.801 | 8425333 | 779.445314 | AFG | 4 |
| 1 | Afghanistan | Asia | 1957 | 30.332 | 9240934 | 820.853030 | AFG | 4 |
| 2 | Afghanistan | Asia | 1962 | 31.997 | 10267083 | 853.100710 | AFG | 4 |
| 3 | Afghanistan | Asia | 1967 | 34.020 | 11537966 | 836.197138 | AFG | 4 |
| 4 | Afghanistan | Asia | 1972 | 36.088 | 13079460 | 739.981106 | AFG | 4 |

# A Basic Plots

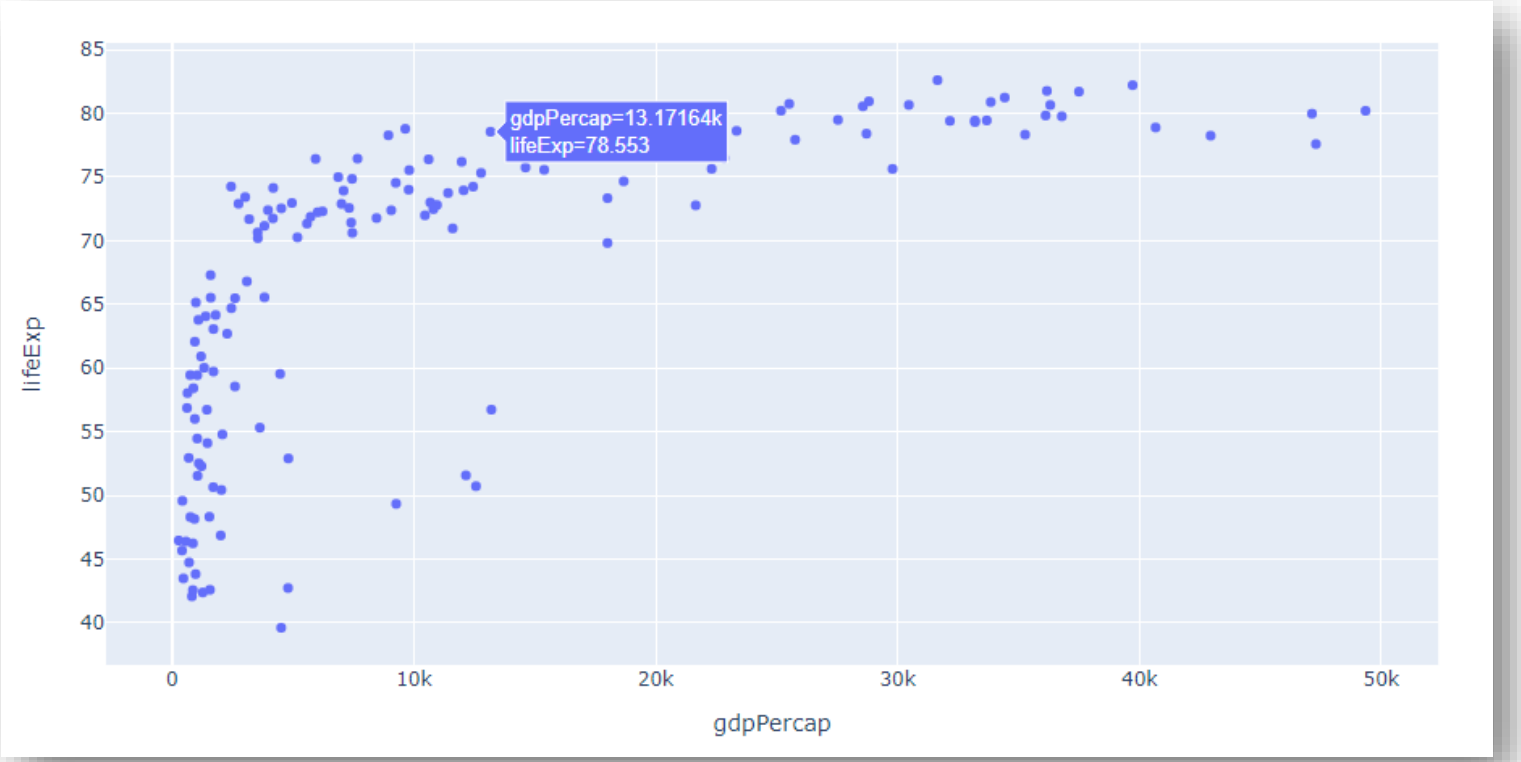
# Scatter Plot

- **px.scatter() function**
  - Each data point is represented as a marker point, whose location is given by the x and y columns

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp')

fig.show()
```

# Scatter Plot

- px.scatter() function



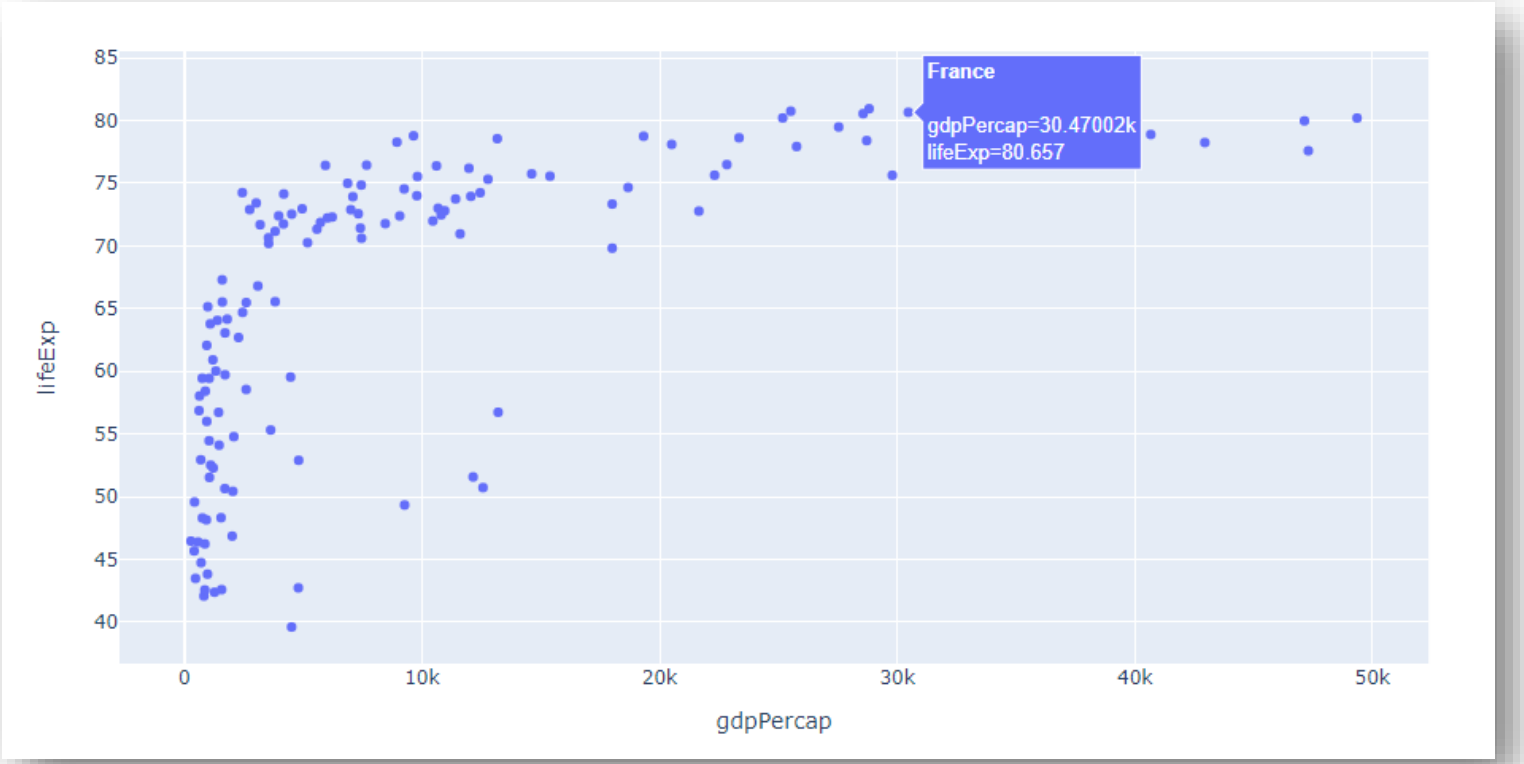
# Scatter Plot

- **px.scatter() function**
  - You can get an interactive information by moving their mouse cursor over the marker
    - Hover_name property

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country')

fig.show()
```

# Scatter Plot

- px.scatter() function
  - Hover_name property



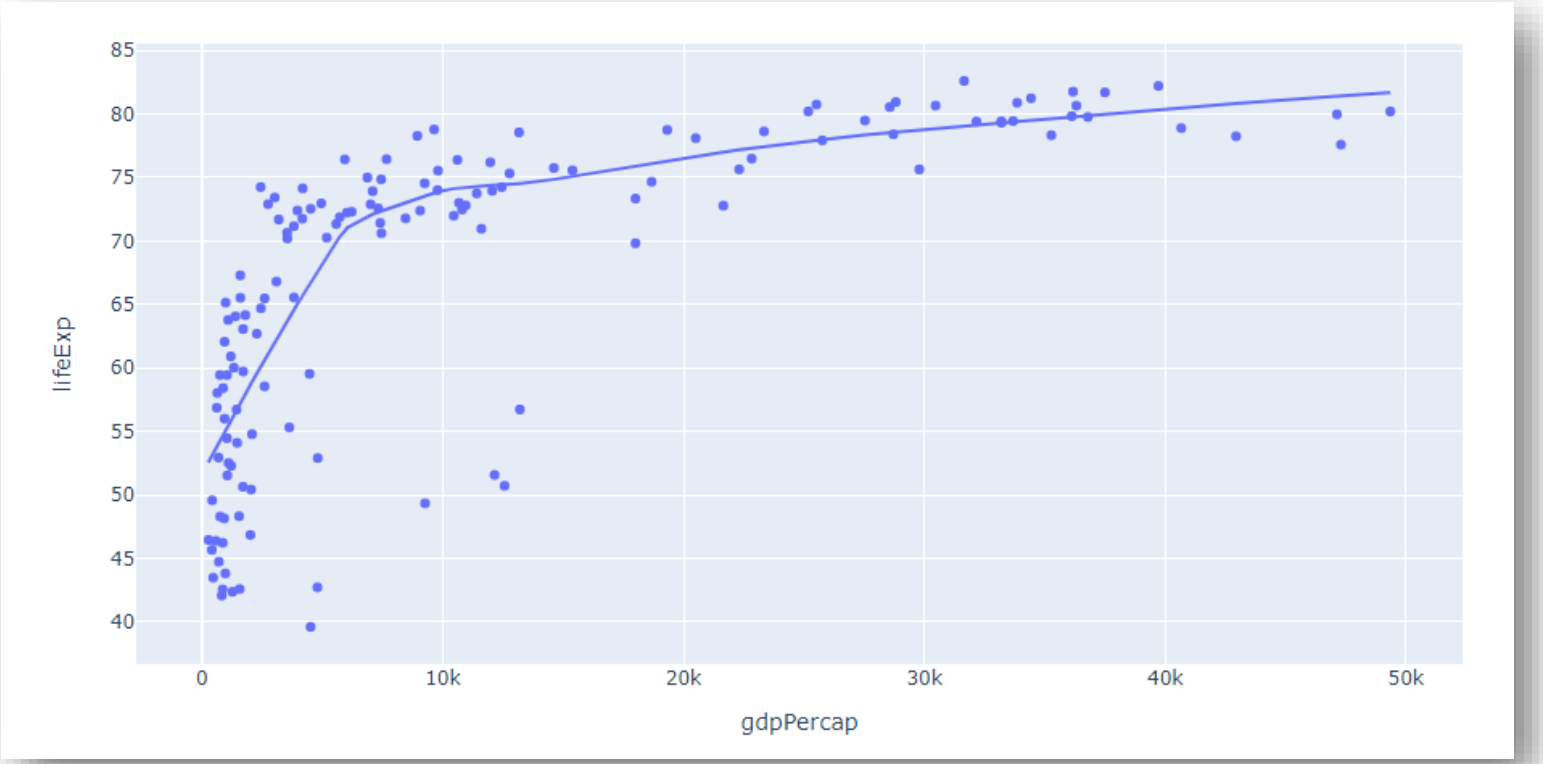
# Scatter Plot

- **px.scatter() function**
  - You can add a trend line to see the trend of the data
    - trendline property with "ols" or "lowess" options

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country',
                 trendline="lowess")

fig.show()
```

# Scatter Plot

- px.scatter() function
  - trendline property



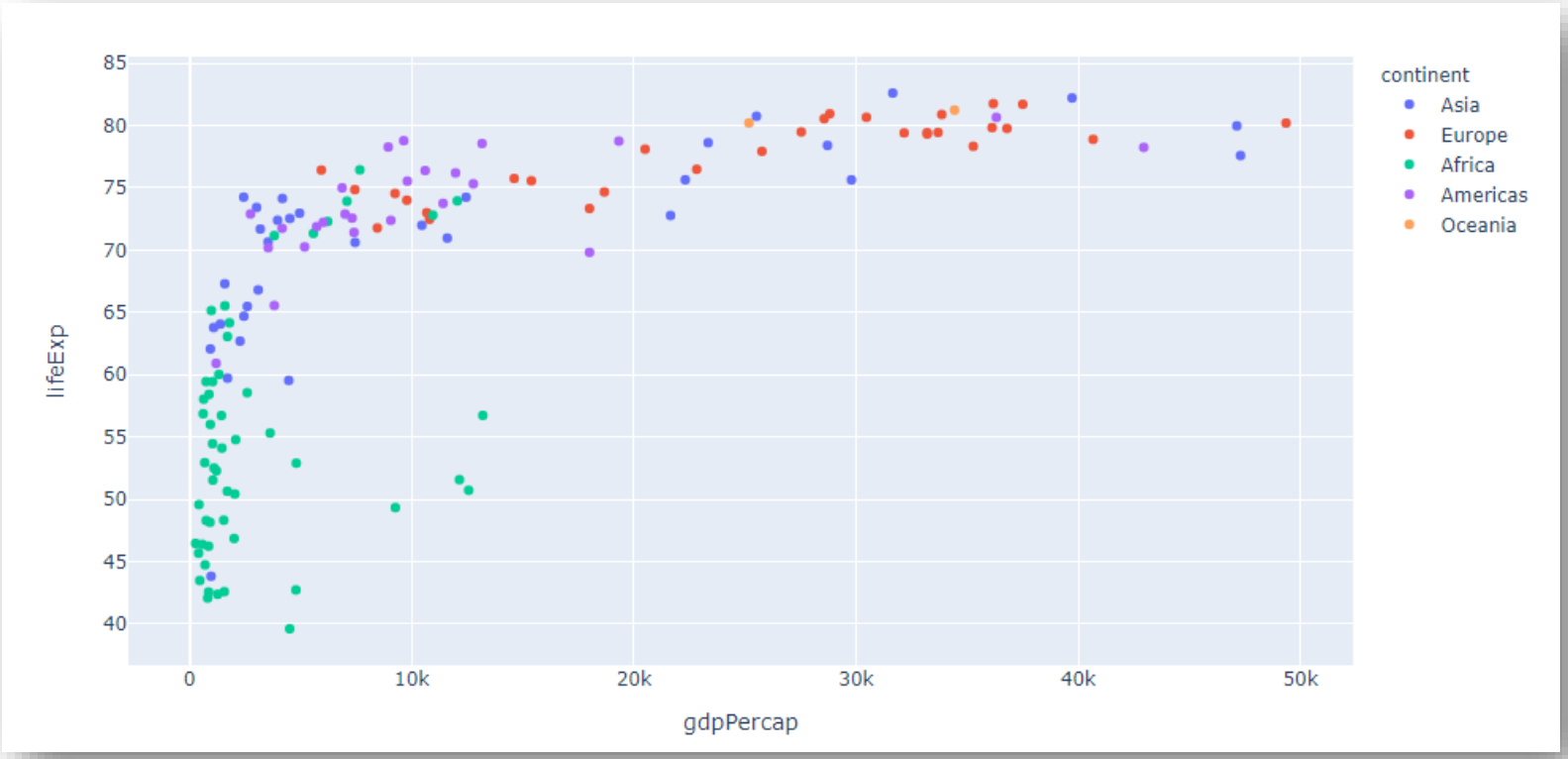
# Scatter Plot

- **px.scatter() function**
  - You can give color to markers
    - color property

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country',
                 color='continent')

fig.show()
```

# Scatter Plot

- px.scatter() function
  - color property



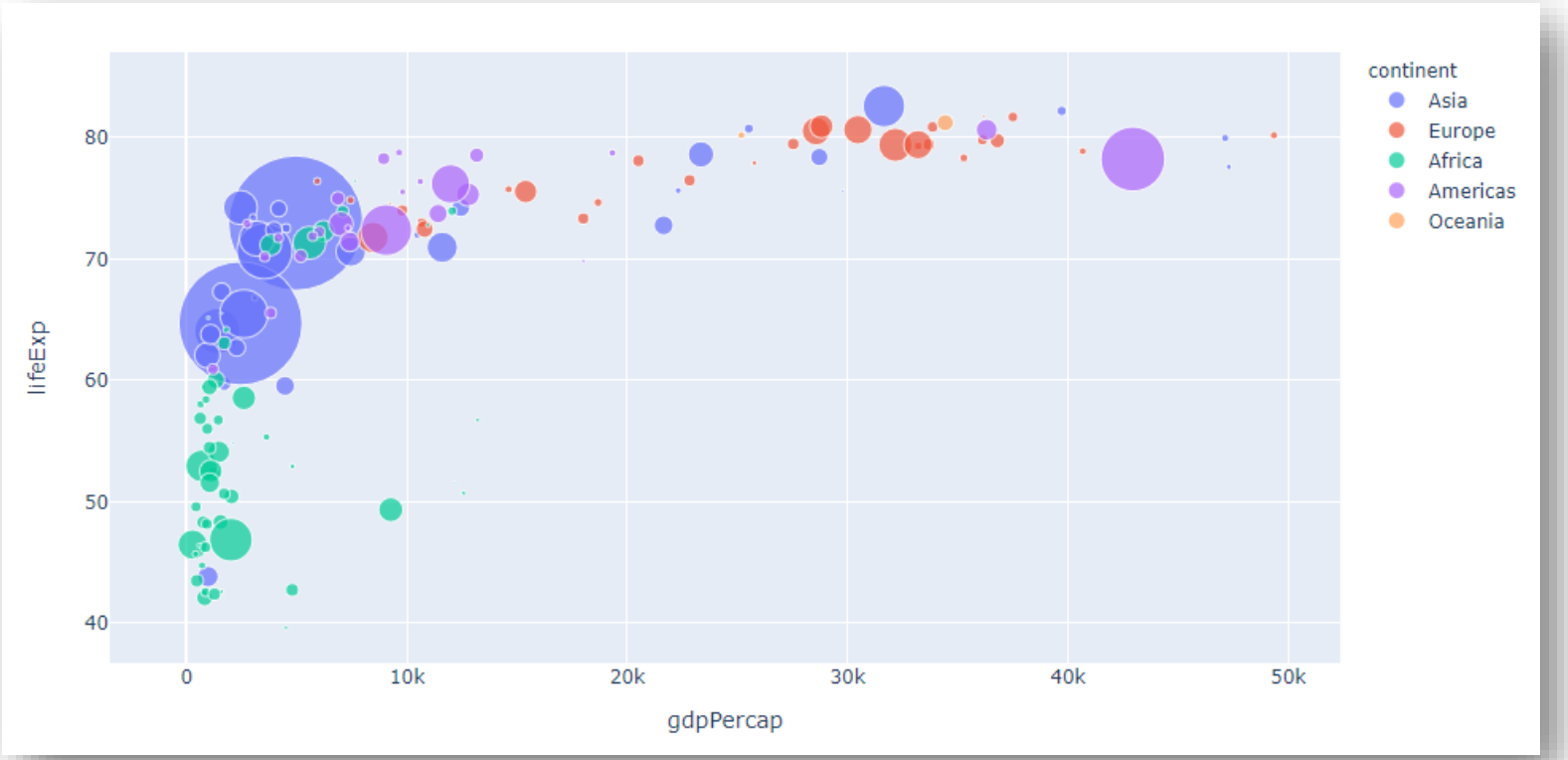
# Scatter Plot

- **px.scatter() function**
  - You can enlarge the marker size
    - size property and size_max property

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country',
                 color='continent',
                 size='pop',
                 size_max=60)

fig.show()
```

# Scatter Plot

- px.scatter() function
  - size property



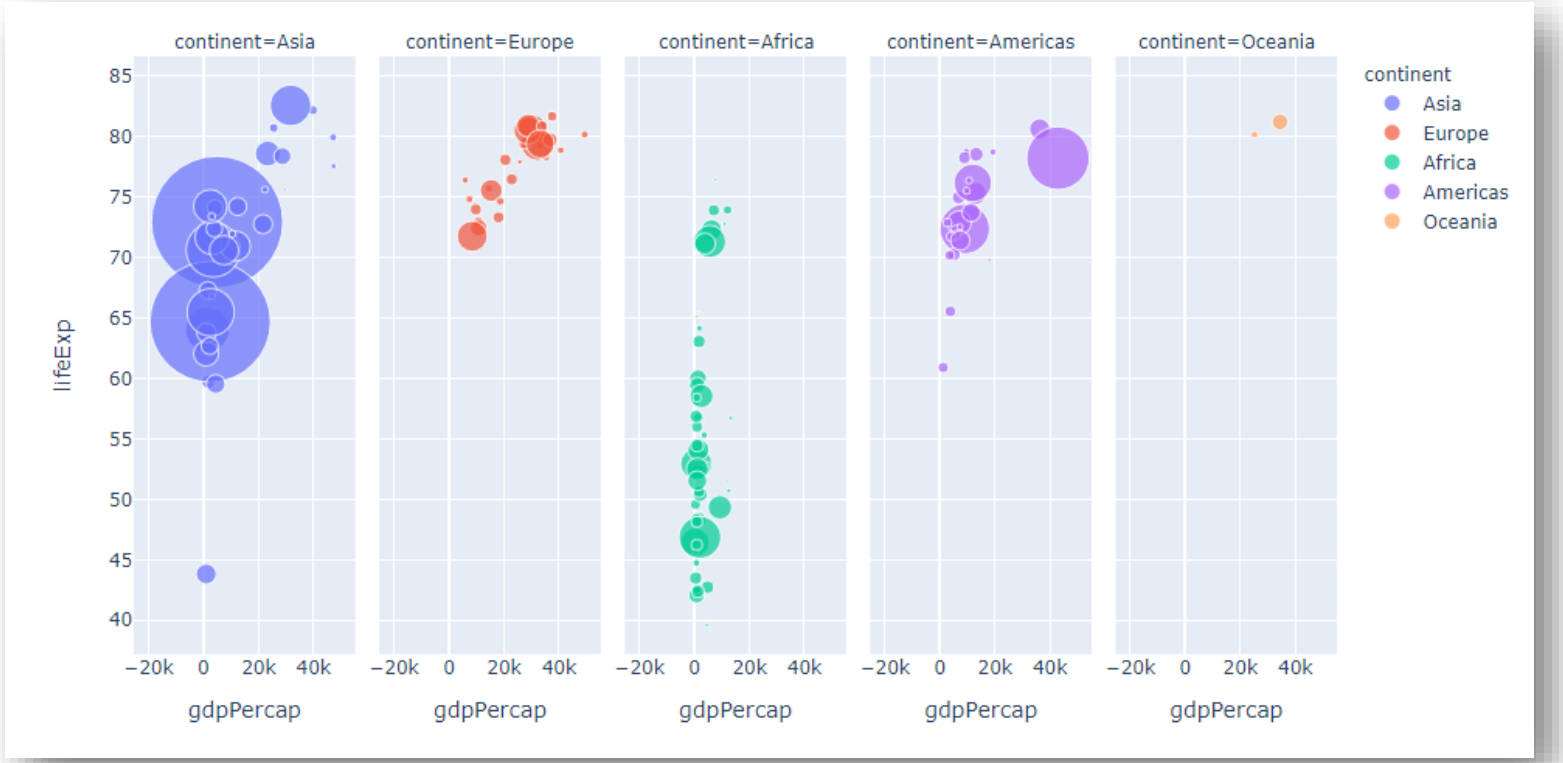
# Scatter Plot

- **px.scatter() function**
  - You can make subplots
    - facet_col property

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country',
                 color='continent',
                 size='pop',
                 size_max=60,
                 facet_col='continent')

fig.show()
```

# Scatter Plot

- **px.scatter() function**
  - facet_col property



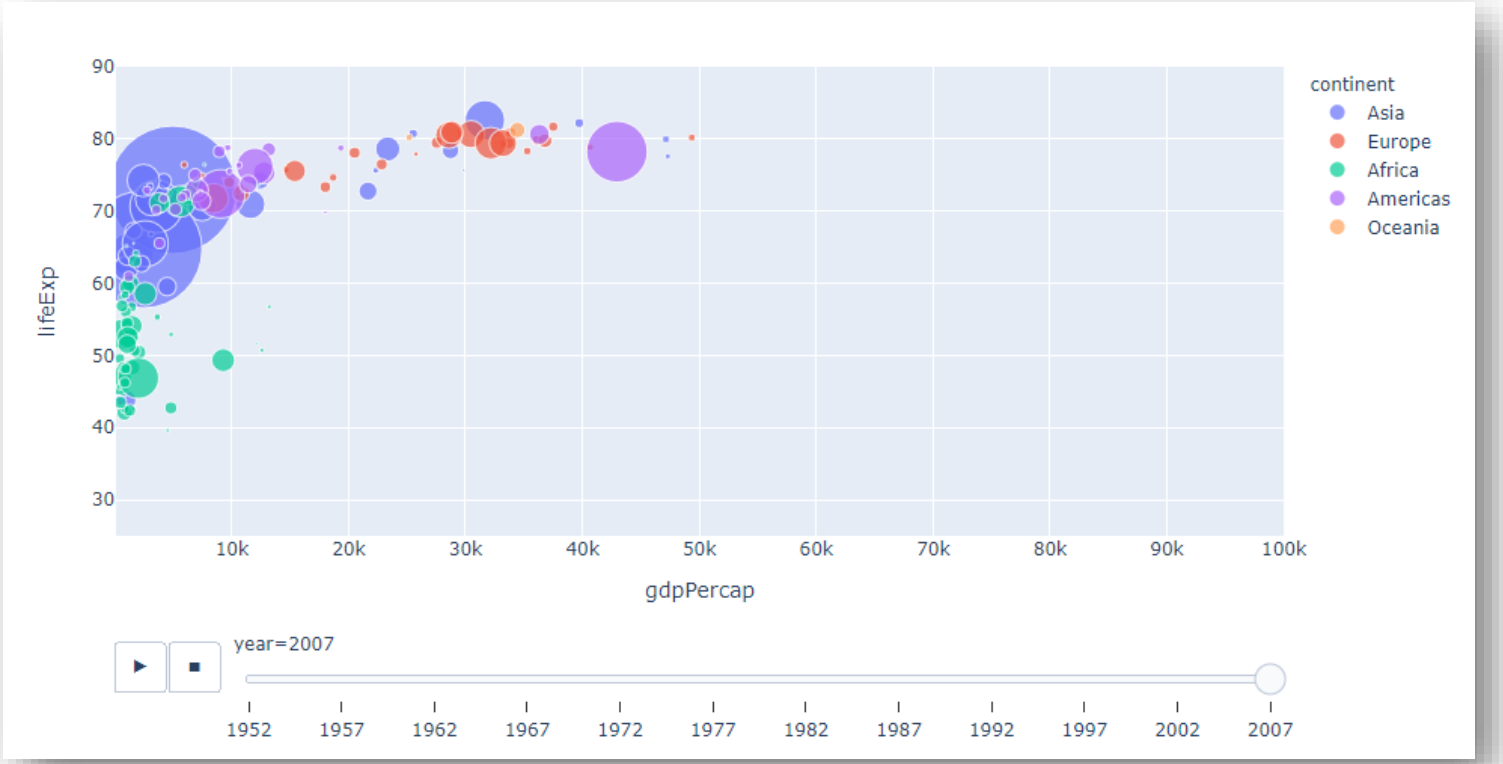
# Scatter Plot

- **px.scatter() function**
  - You can make animated plots
    - animation_frame and animation_group properties

```
df = px.data.gapminder()

fig = px.scatter(df,
                 x='gdpPercap',
                 y='lifeExp',
                 hover_name='country',
                 color='continent',
                 size='pop',
                 size_max=60,
                 animation_frame="year",
                 animation_group="country",
                 range_x=[100,100000],
                 range_y=[25,90])

fig.show()
```

# Scatter Plot

- px.scatter() function
  - animation_frame and animation_group properties



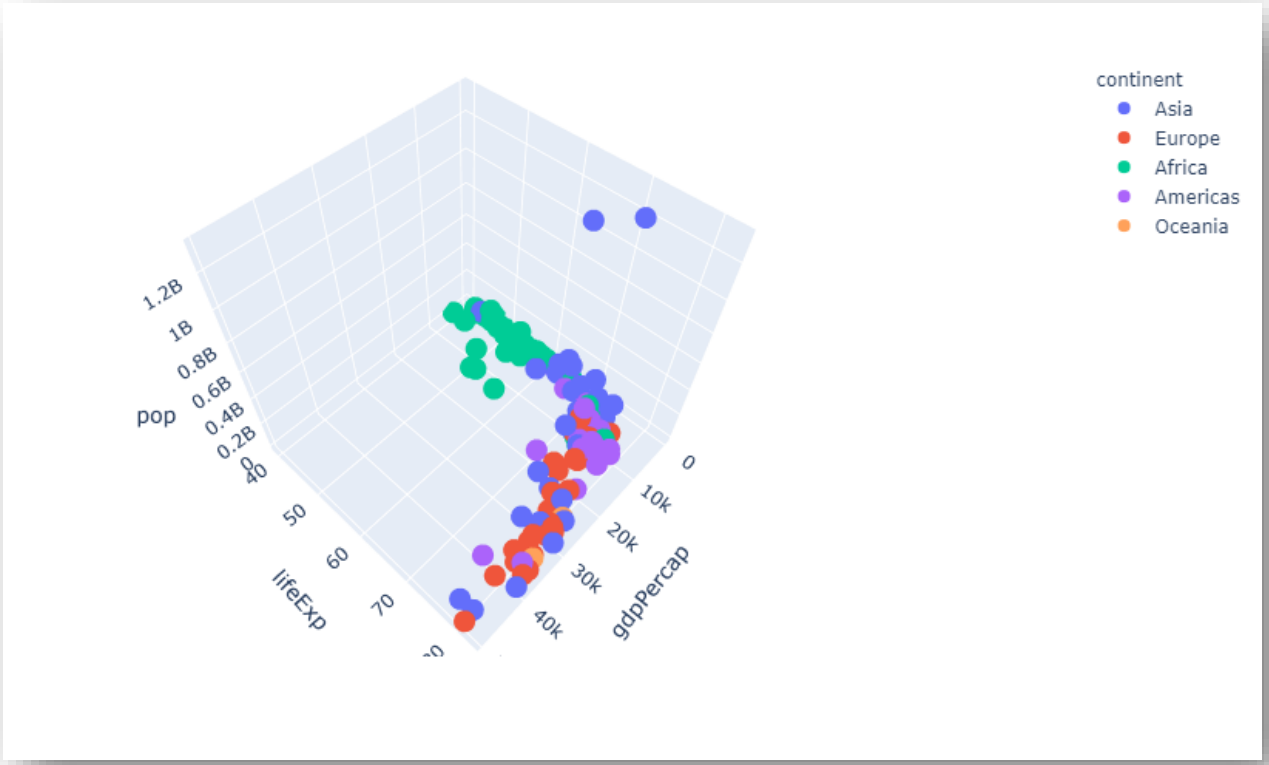
# Scatter Plot

- **px.scatter()_3d function**
  - You can create 3D plots
    - Don't forget to add x, y and z coordinates

```
df = px.data.gapminder().query('year == 2007')

fig = px.scatter_3d(df,
                    x='gdpPercap',
                    y='lifeExp',
                    z = "pop",
                    hover_name='country',
                    color='continent')

fig.show()
```

# Scatter Plot

- px.scatter()_3d function



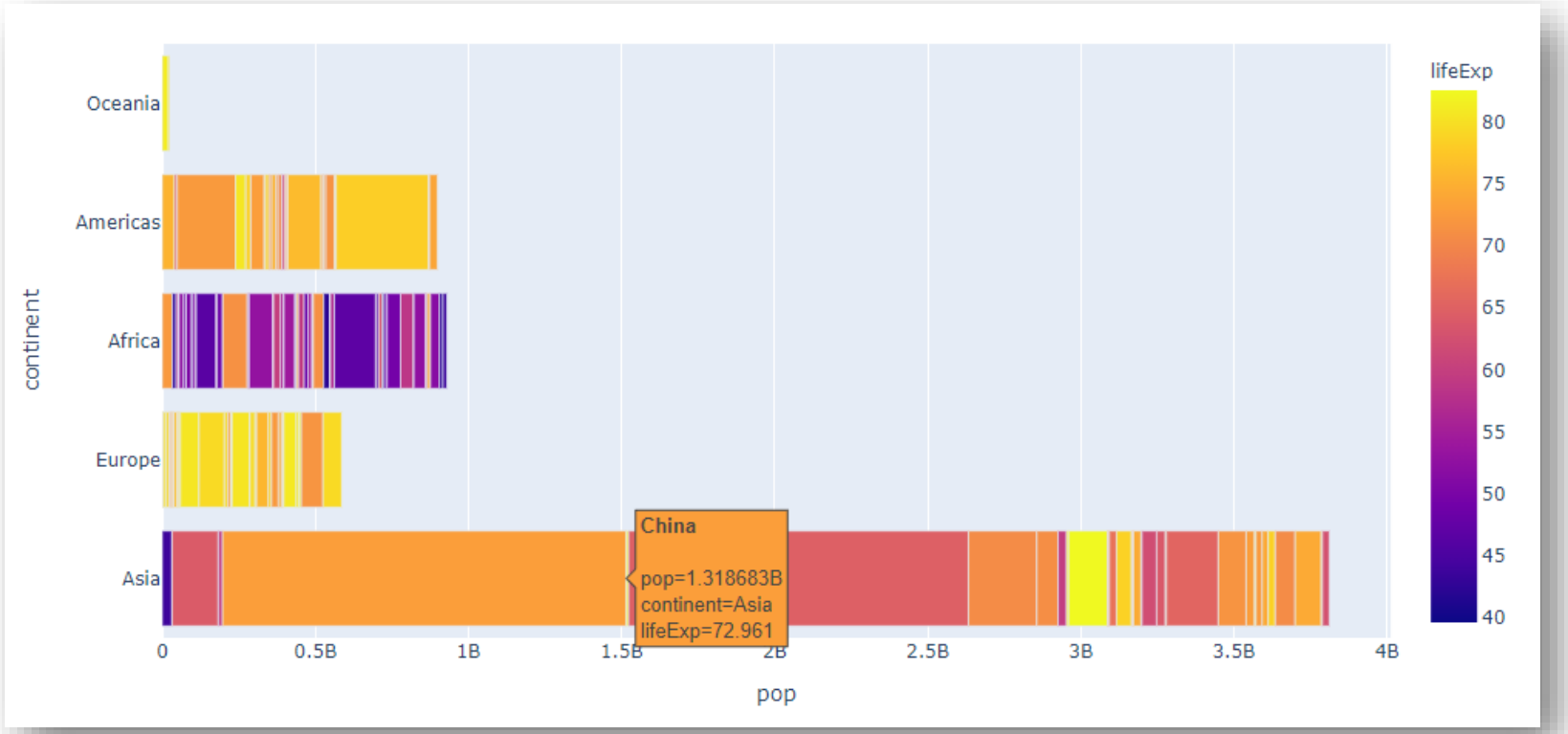
# Bar Plot

- **px.bar() function**
  - Group bar plot

```
df = px.data.gapminder().query('year == 2007')

fig = px.bar(df,
             x='pop',
             y='continent',
             color='lifeExp',
             hover_name='country')

fig.show()
```

# Bar Plot

- px.scatter()_3d function
  - Group bar plot



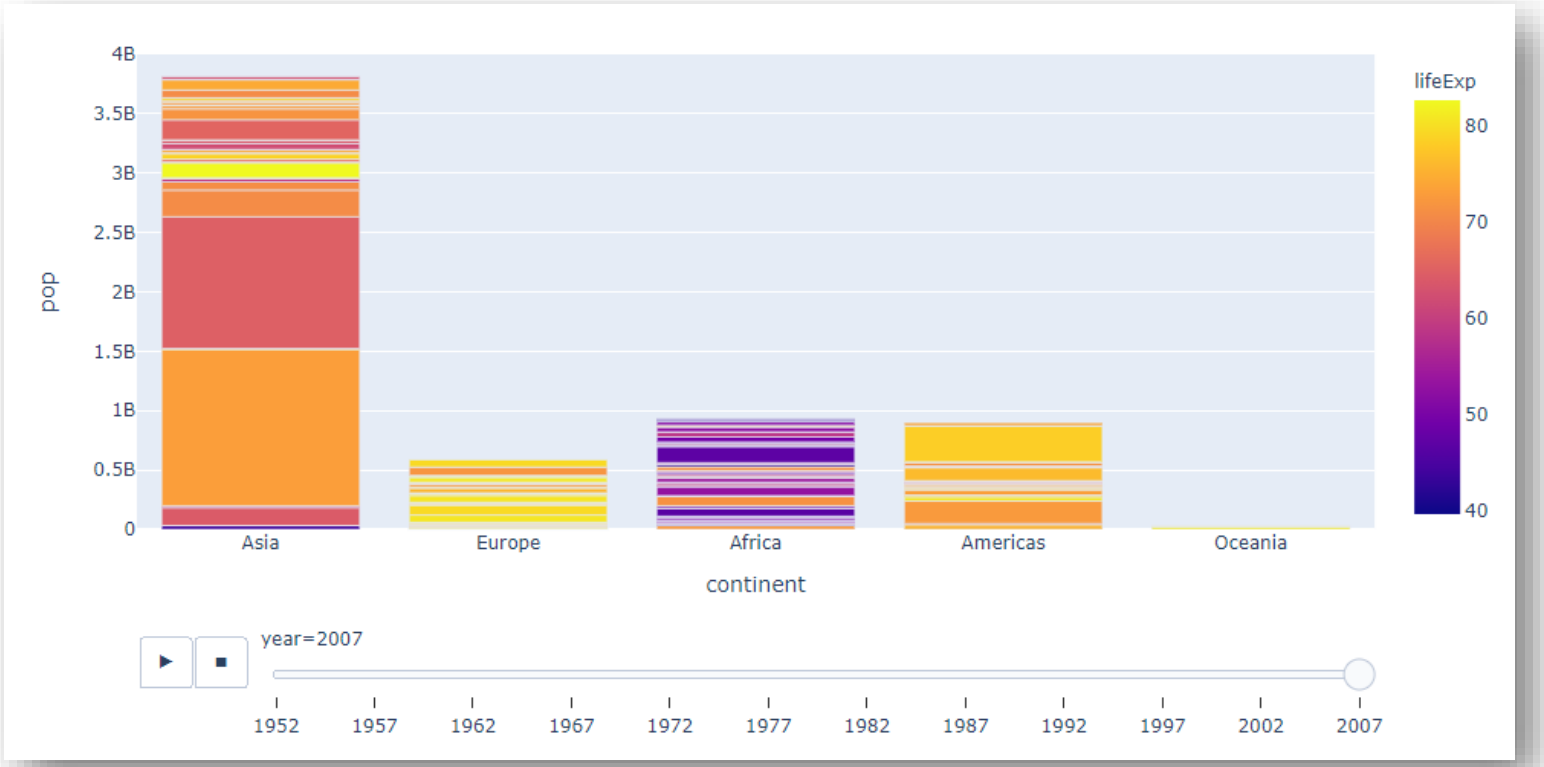
# Bar Plot

- **px.bar() function**
  - Animated group bar plot

```
df = px.data.gapminder()

fig = px.bar(df,
             x ="continent",
             y ="pop",
             color ='lifeExp',
             hover_name ='country',
             animation_frame ='year',
             animation_group="country",
             range_y =[0, 4000000000])

fig.show()
```

# Bar Plot

- px.scatter()_3d function
  - Animated group bar plot



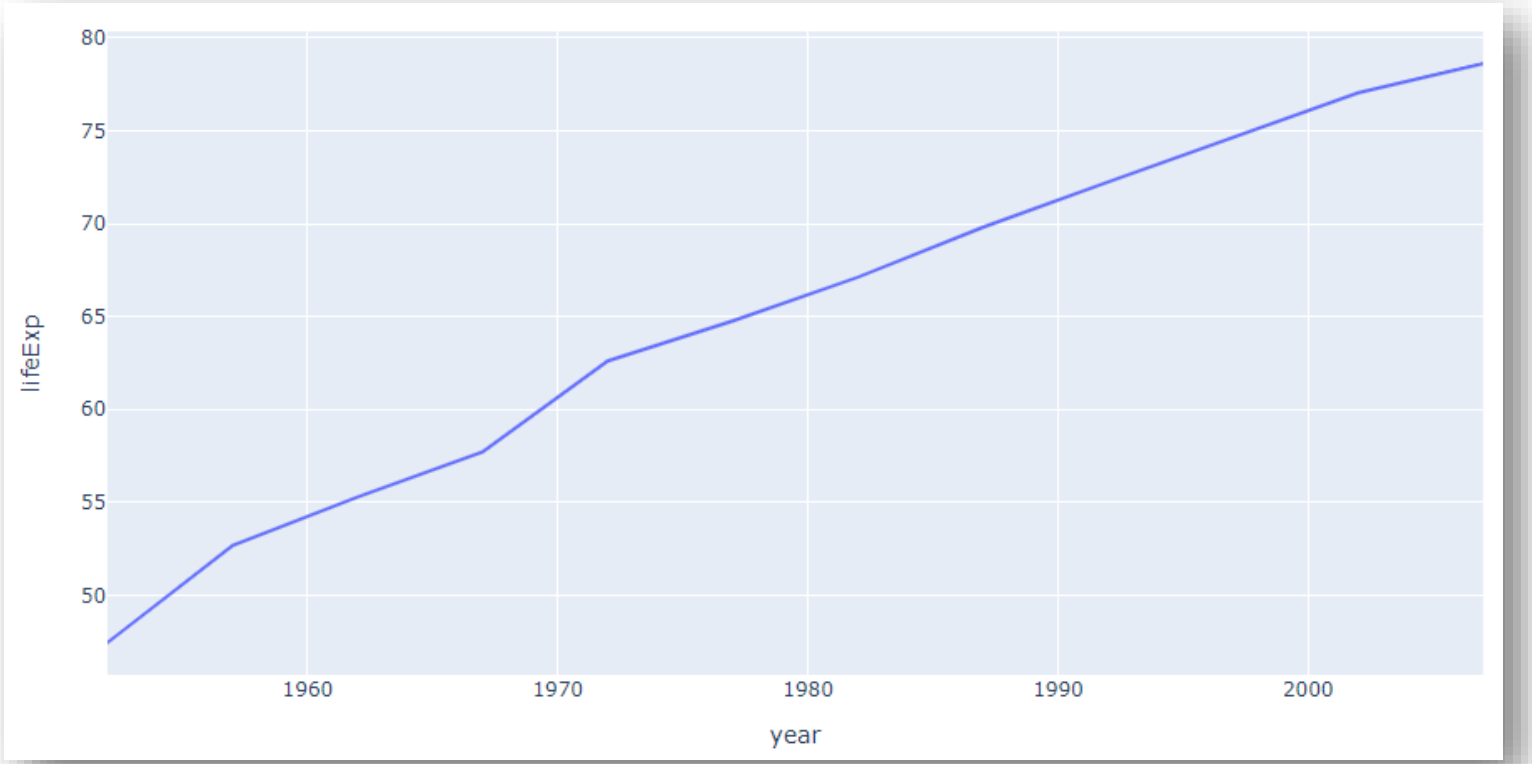
# Line Plot

- **px.line() function**
  - Line plot

```
df = px.data.gapminder().query("country=='Korea, Rep.'")

fig = px.line(df,
              x="year",
              y="lifeExp")

fig.show()
```

# Line Plot

- px.line() function
  - Line plot



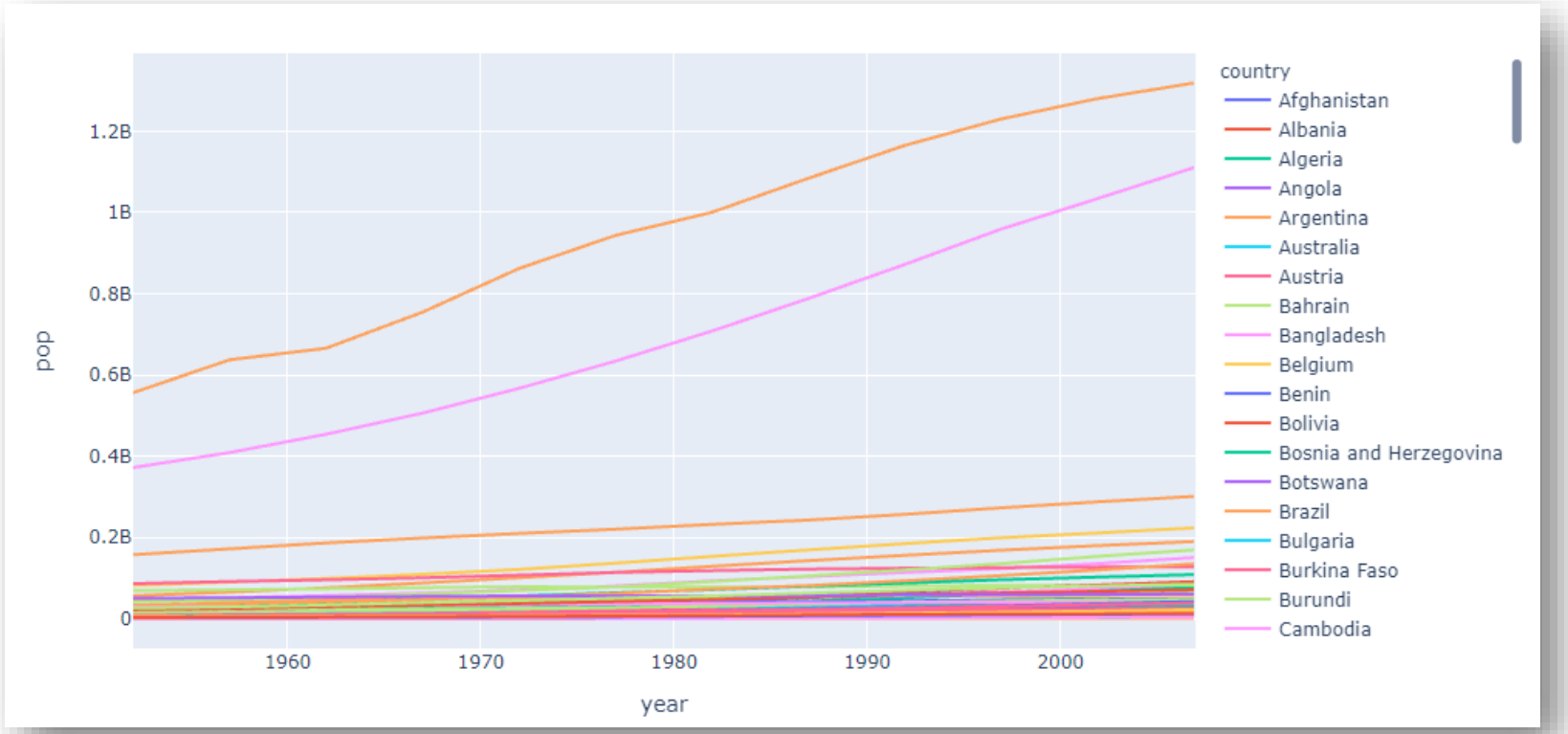
# Line Plot

- **px.line() function**
  - Line plot

```
df = px.data.gapminder()

fig = px.line(df,
              x="year",
              y="pop",
              color = 'country')

fig.show()
```

# Line Plot

- **px.line() function**
  - Line plot



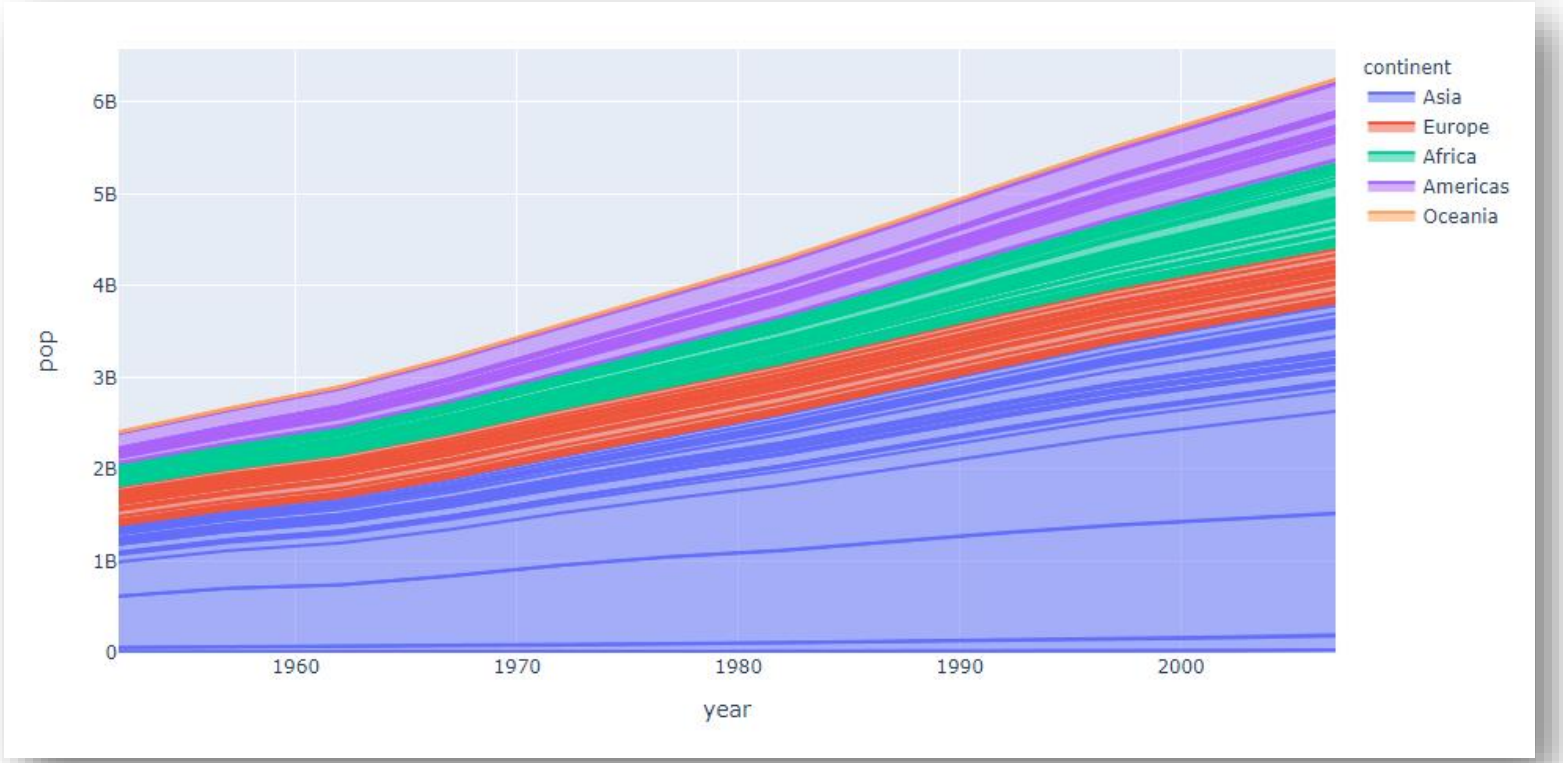
# Area Plot

- **px.area() function**
  - You can plot lines in the form of combined area

```
df = px.data.gapminder()

fig = px.area(df,
              x="year",
              y="pop",
              color="continent",
              line_group="country")

fig.show()
```

# Line Plot

- **px.area() function**
  - Area plot



B

# Part-of-Whole

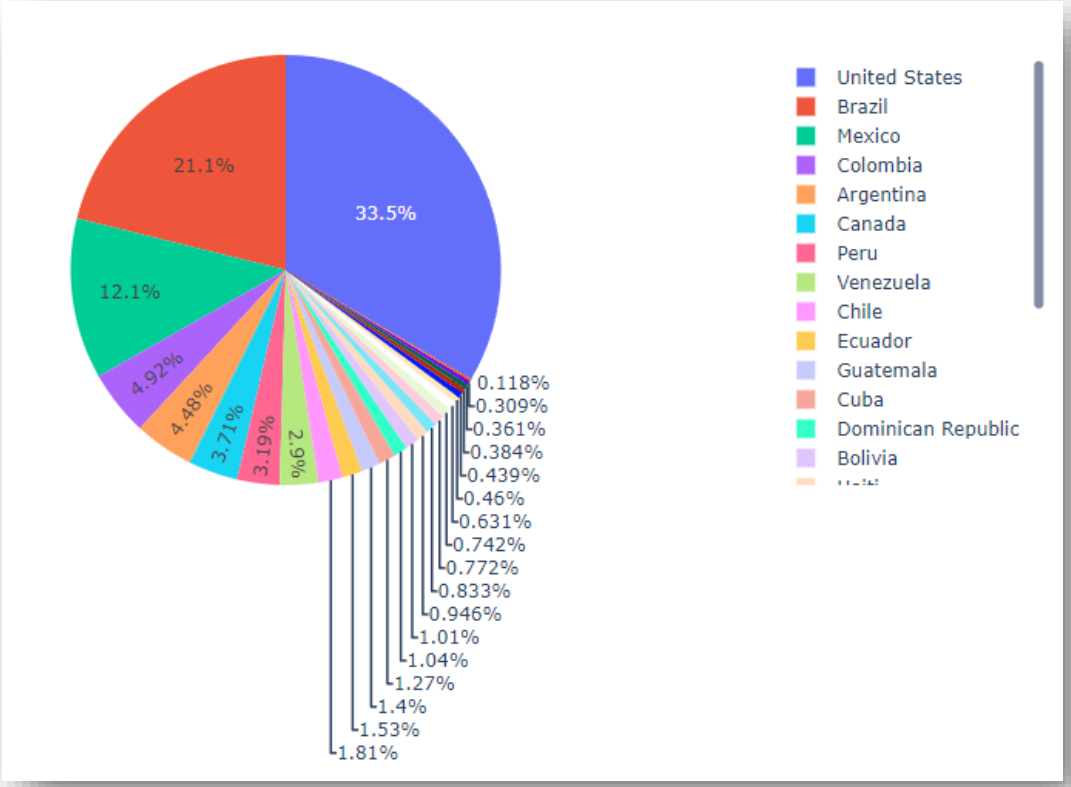
# Pie Plot

- **px.pie() function**
  - Pie plot are used to show the composition of the dataset

```
df = px.data.gapminder().query("continent=='Americas'").
                                    query("year == 2007")

fig = px.pie(df,
             values='pop',
             names='country')

fig.show()
```

# Pie Plot

- px.pie() function
  - Pie plot



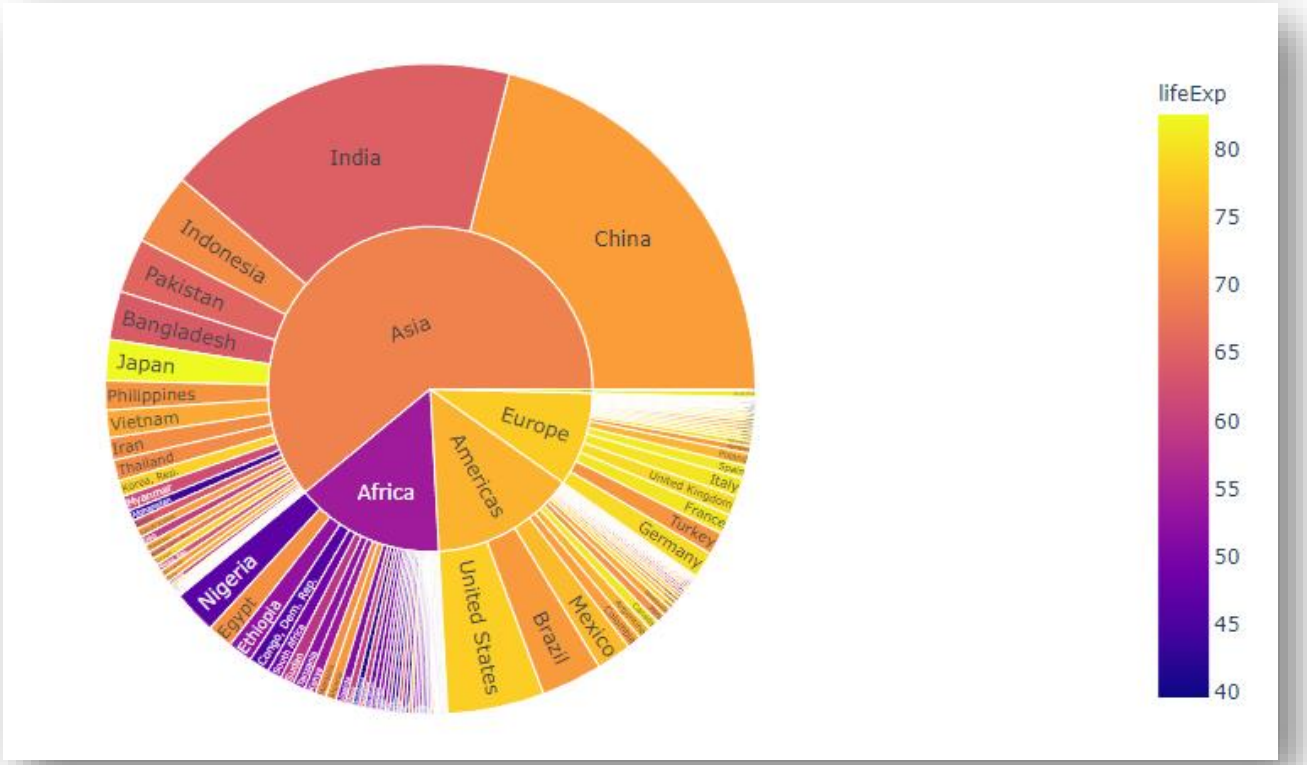
# Sunburst Plot

- **px.sunburst() function**
  - Visualize hierarchical data using pie

```
df = px.data.gapminder().query("year == 2007")

fig = px.sunburst(df,
                  path=['continent', 'country'],
                  values='pop',
                  color='lifeExp')

fig.show()
```

# Sunburst Plot

- `px.sunburst()` function



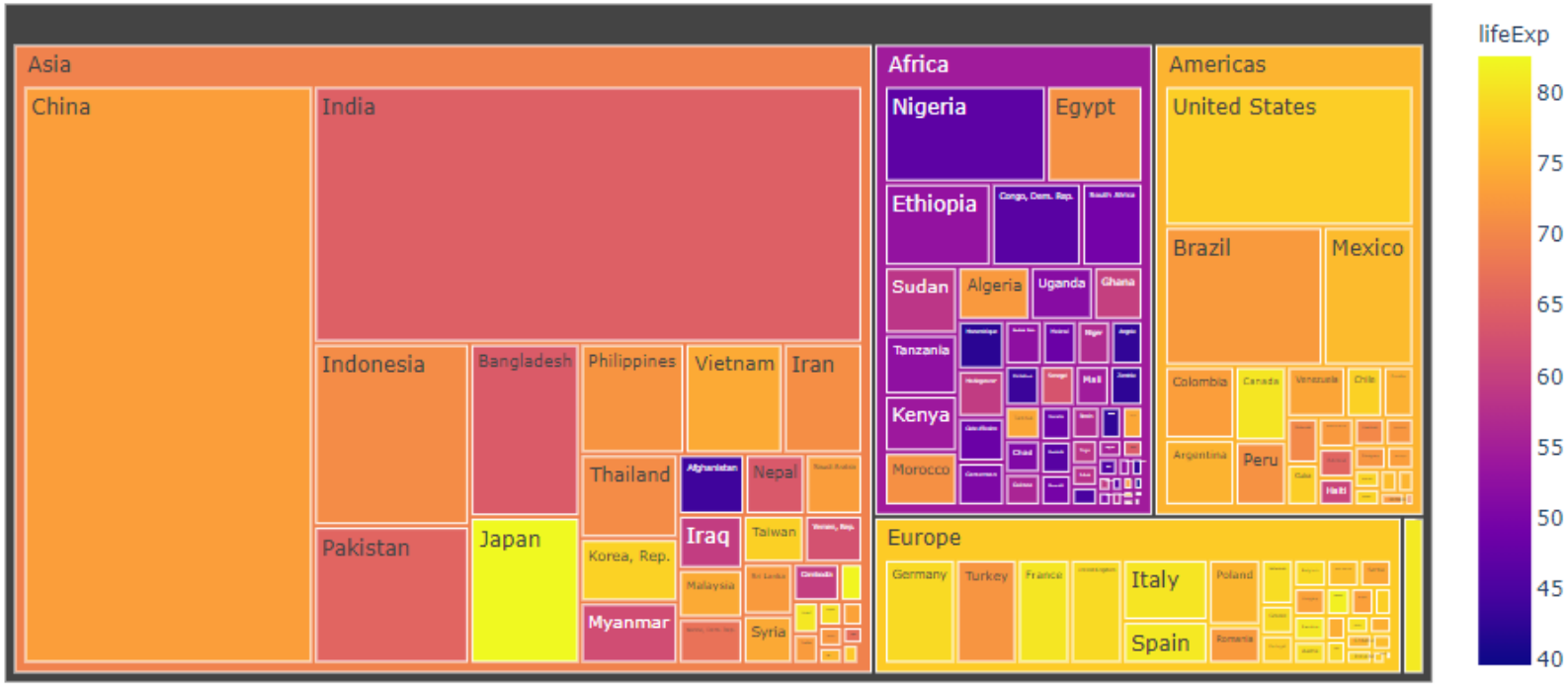
# TreeMap Plot

- **px.treemap() function**
  - Visualize hierarchical data using nested rectangles

```
df = px.data.gapminder().query("year == 2007")

fig = px.treemap(df,
                 path=['continent', 'country'],
                 values='pop',
                 color='lifeExp')
fig.show()
```

# Sunburst Plot

- px.treemap() function



C

# Distributions

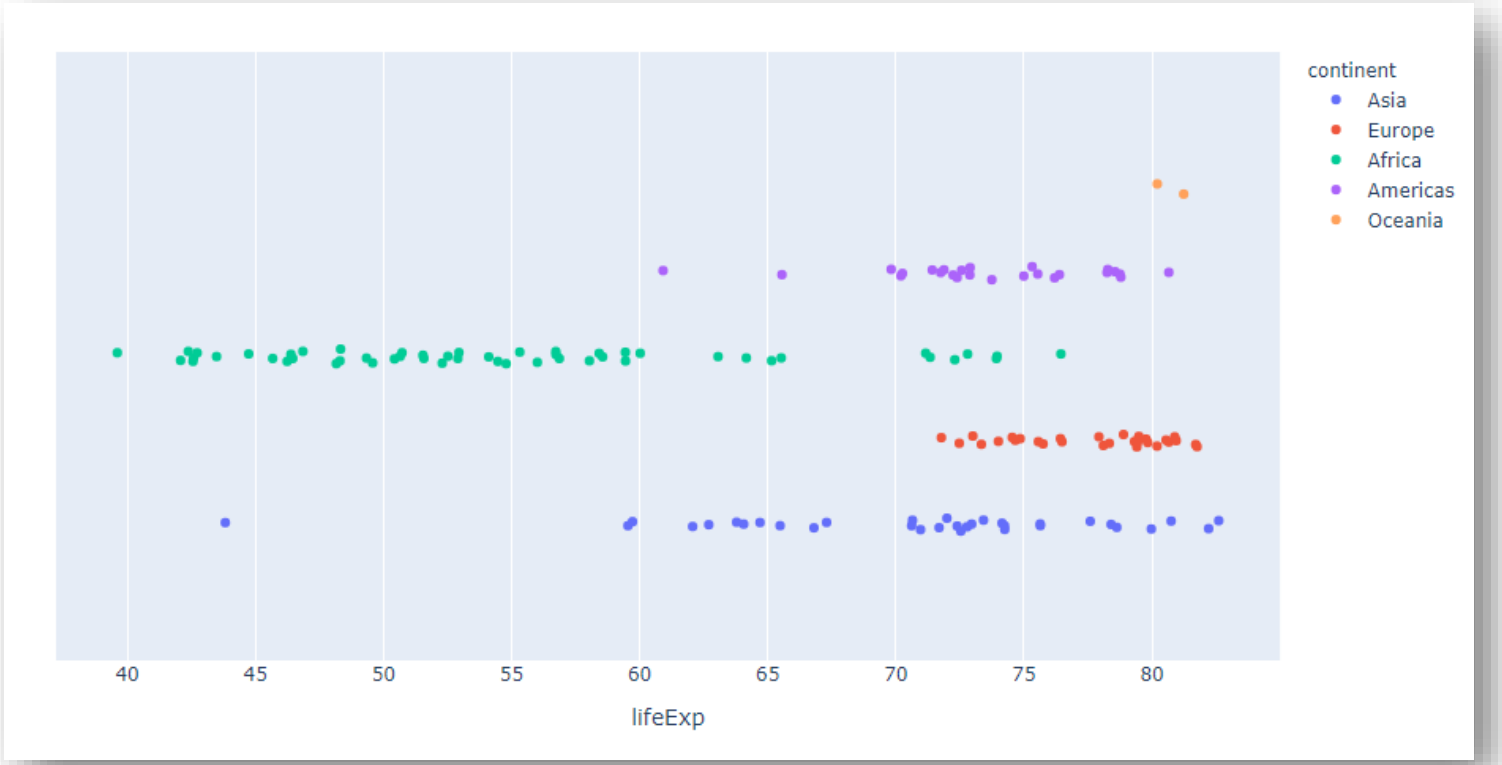
# Strip Plot

- **px.strip()**
  - Strip charts are like 1-dimensional jittered scatter plots

```
df = px.data.gapminder().query("year == 2007")

fig = px.strip(df,
        x = "lifeExp",
        hover_name = "country",
        color = "continent")

fig.show()
```

# Strip Plot

- px.strip()



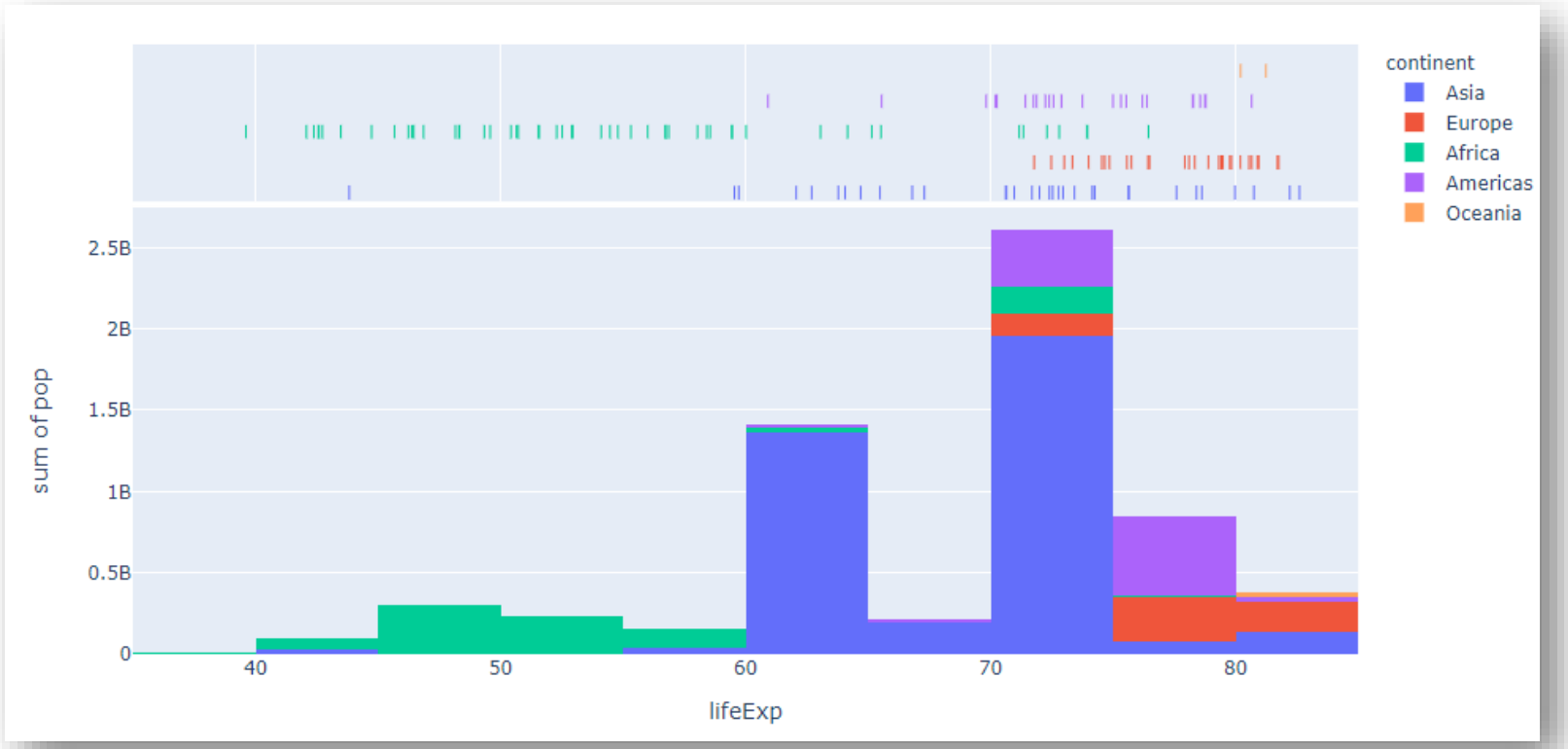
# Histogram Plot

- **px.histogram() function**
  - Used to check data distribution
    - With marginal property, a marginal is drawn alongside the histogram

```
df = px.data.gapminder().query("year == 2007")

fig = px.histogram(df,
                   x = "lifeExp",
                   y ="pop",
                   hover_name = "country",
                   color = "continent",
                   marginal = "rug")

fig.show()
```

# Histogram Plot

- px.histogram() function



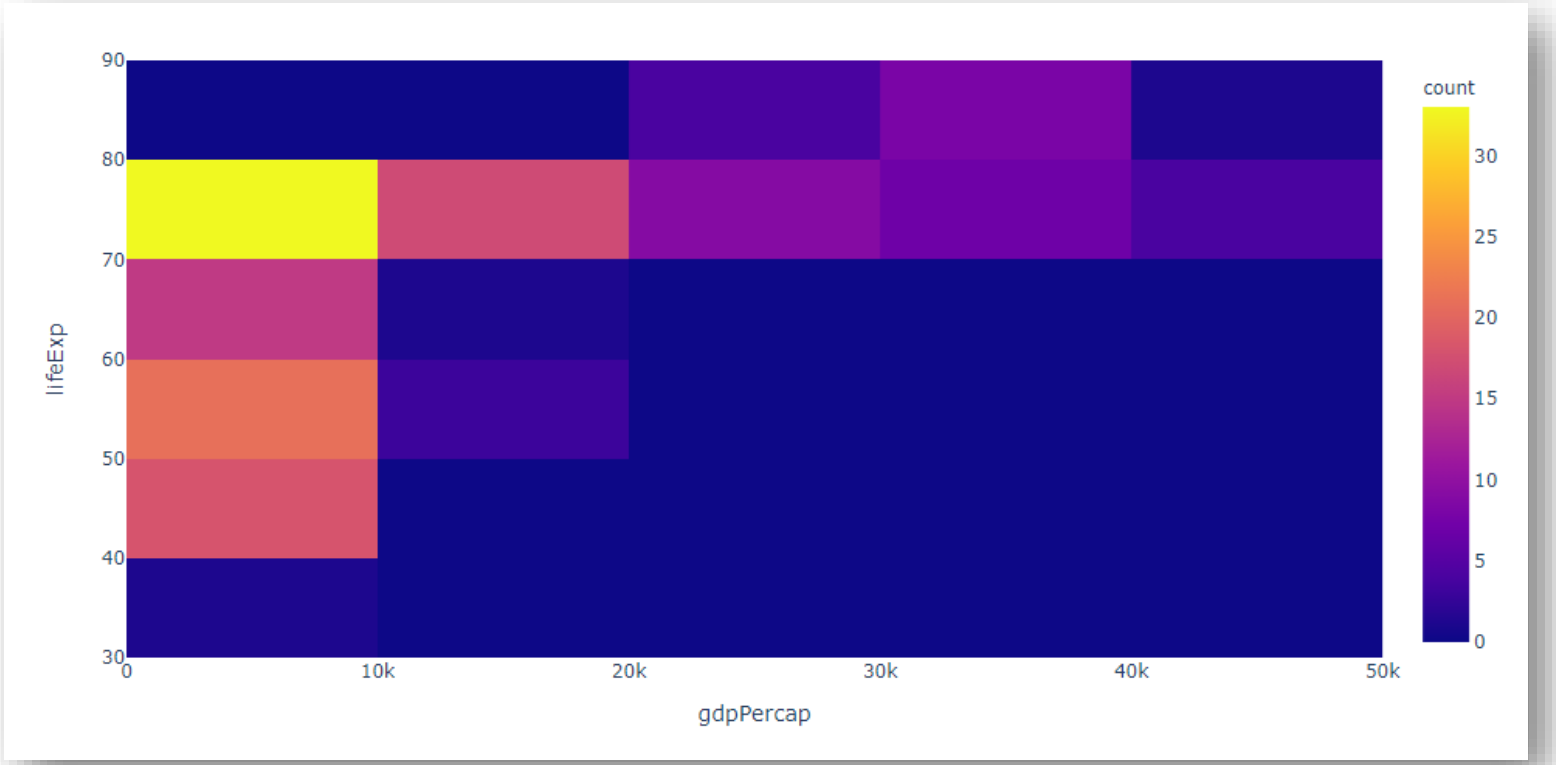
# Density HeatMap Plot

- **px.density_heatmap()**
  - Used to see the density between two variables

```
df = px.data.gapminder().query("year == 2007")

fig = px.density_heatmap(df,
                         x="gdpPercap",
                         y="lifeExp")

fig.show()
```

# Histogram Plot

- px.density_heatmap()



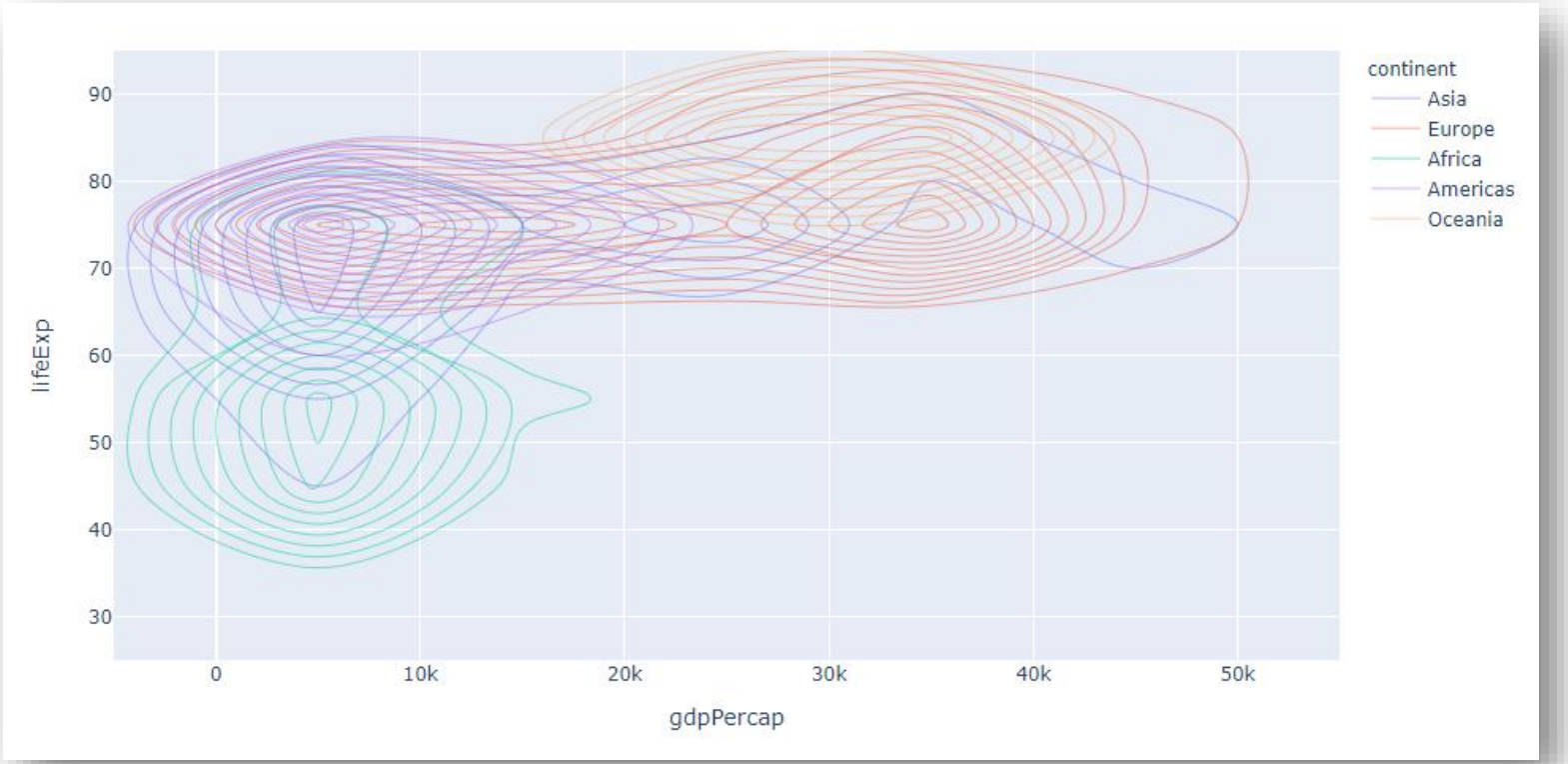
# Density Contour Plot

- **px.density_contour()**
  - Used to see the density between two variables

```
df = px.data.gapminder().query("year == 2007")

fig = px.density_contour(df,
                         x="gdpPercap",
                         y="lifeExp",
                         color ="continent")

fig.show()
```

# Density Contour Plot

- px.density_contour()



# D

# Advanced Plots

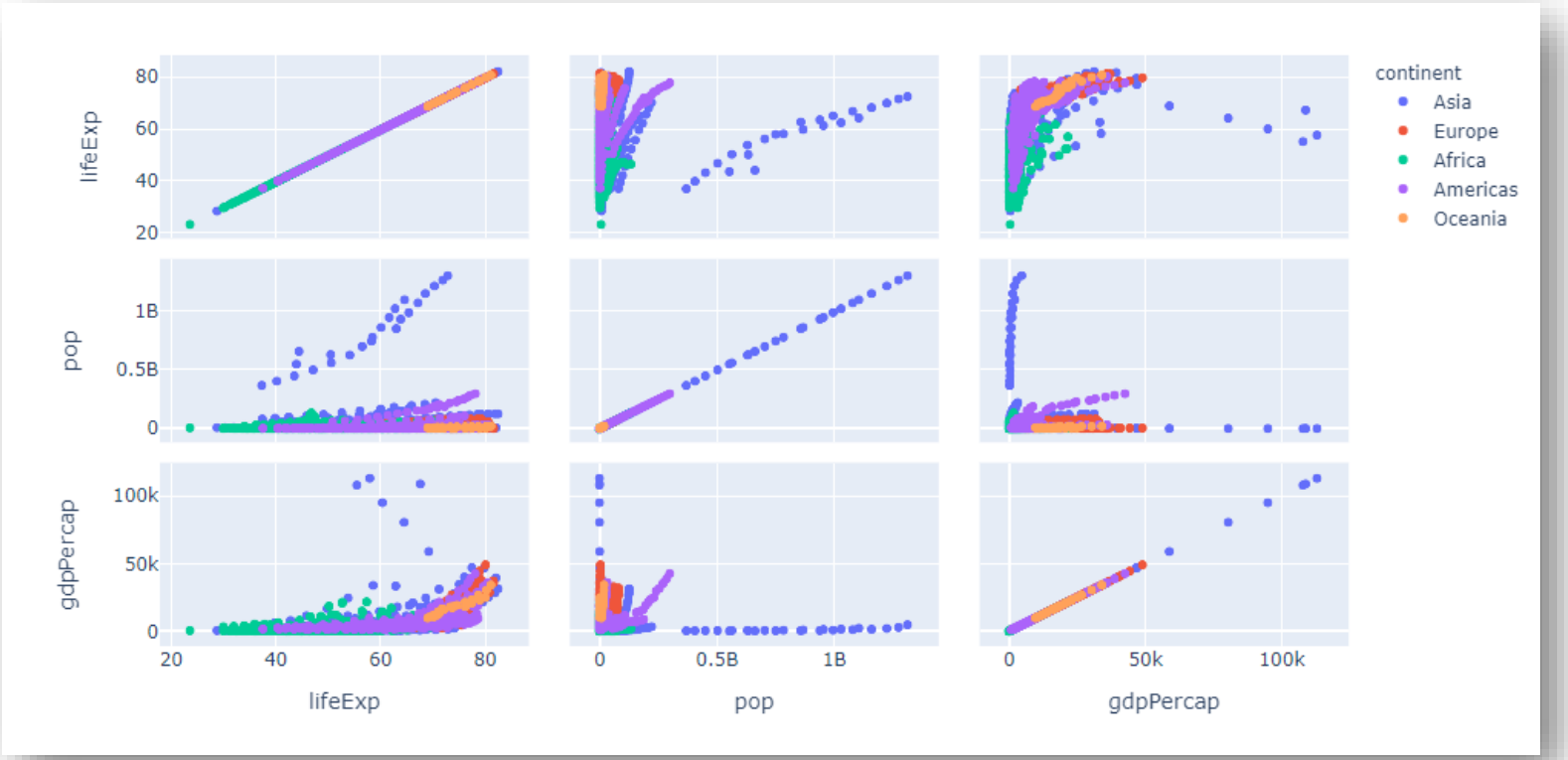
# Scatter Matrix

- px.scatter_matrix() function
  - Used to check the relationship between several variables

```
df = px.data.gapminder()

fig = px.scatter_matrix(df,
                        dimensions=["lifeExp", "pop", "gdpPercap"],
                        color="continent")

fig.show()
```

# Scatter Matrix

- px.scatter_matrix() function



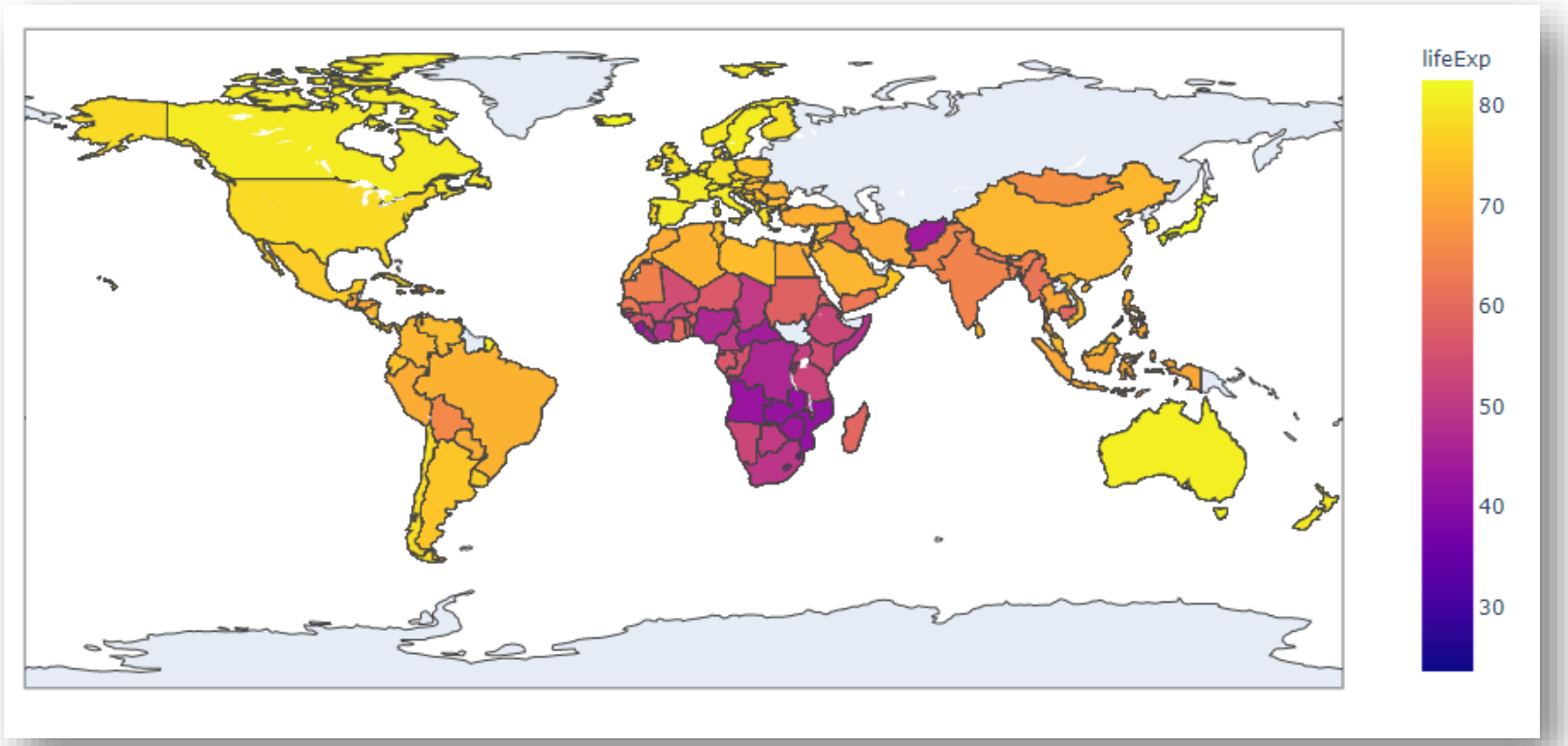
# Choropleth Maps

- **px.choropleth() function**
  - A map composed of colored polygons

```
df = px.data.gapminder()

fig = px.choropleth(df,
                    locations="iso_alpha",
                    color="lifeExp",
                    hover_name="country")

fig.show()
```

# Choropleth Maps

- px.choropleth() function



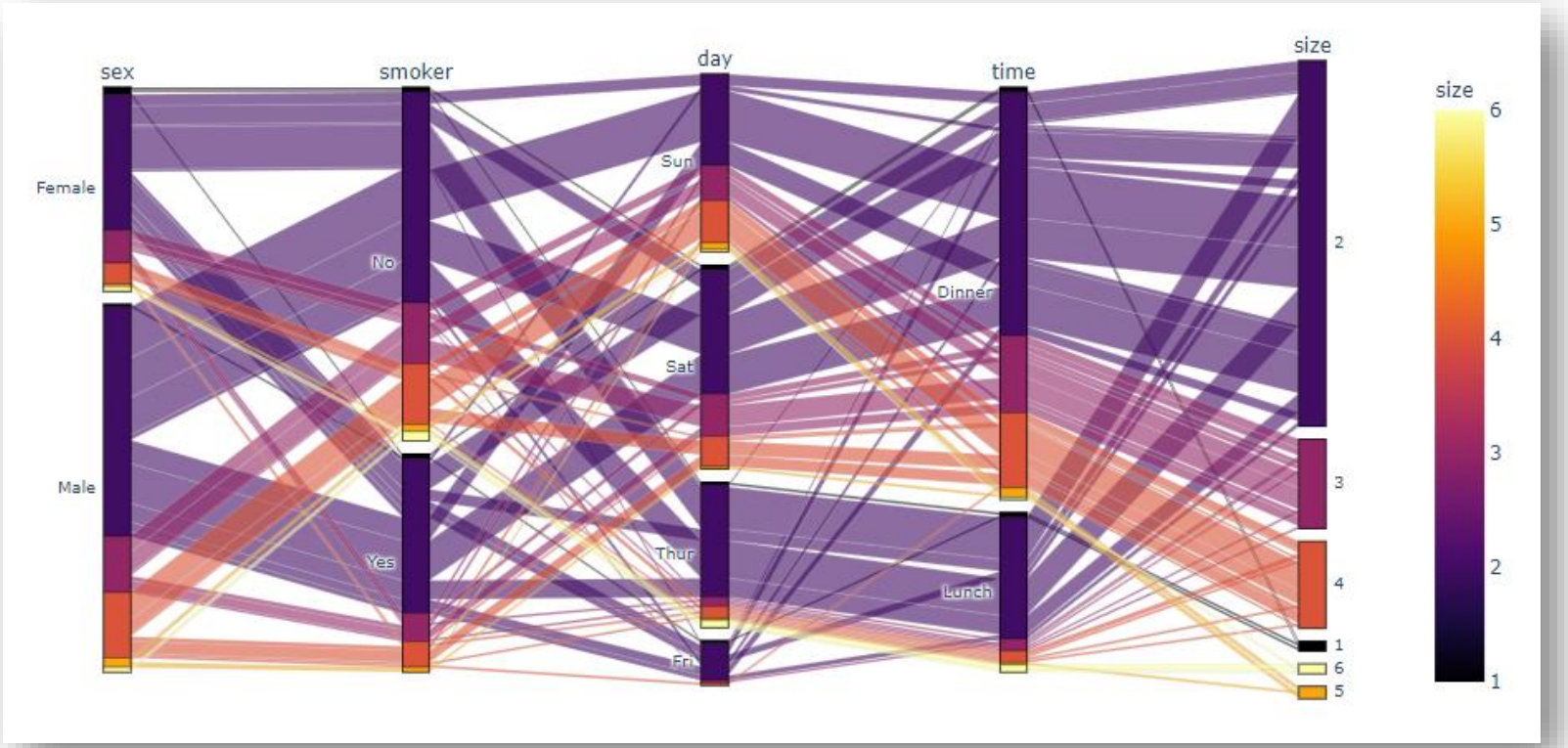
# Parallel Categories

- **px.parallel_categories() function**
  - A visualization of multi-dimensional categorical data sets

```
df = px.data.tips()


fig = px.parallel_categories(df,
                             color="size",
                             color_continuous_scale=
                                    px.colors.sequential.Inferno)

fig.show()
```

# Parallel Categories

- px.parallel_categories() function



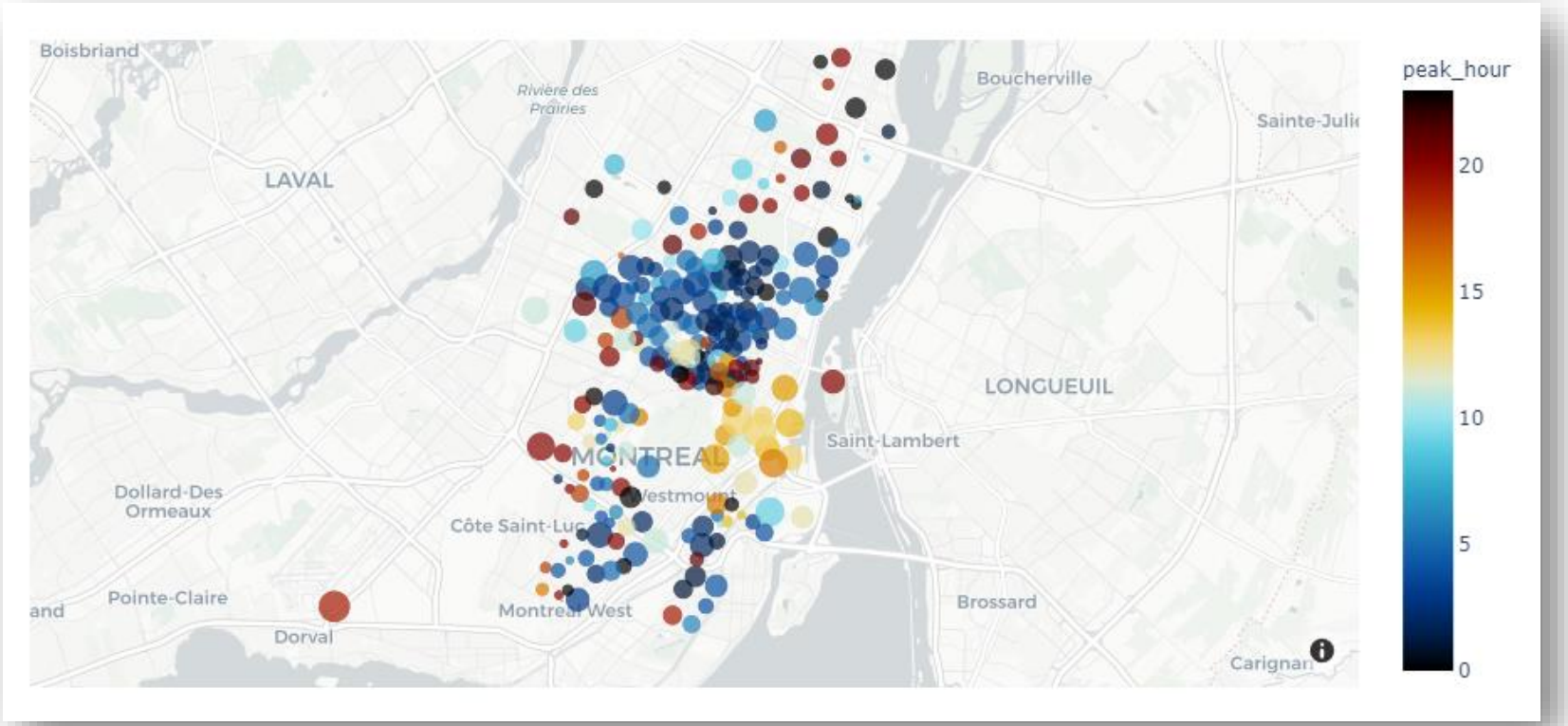
# Scatter Mapbox

- **px.scatter_mapbox() function**
  - Used to scatter data points over the map

```
df = px.data.carshare()

fig = px.scatter_mapbox(df,
                        lat="centroid_lat",
                        lon="centroid_lon",
                        color="peak_hour",
                        size="car_hours",

color_continuous_scale=px.colors.cyclical.IceFire,
                        size_max=15,
                        zoom=10,
                        mapbox_style="carto-positron")
fig.show()
```

# Scatter Mapbox

- px.scatter_mapbox() function



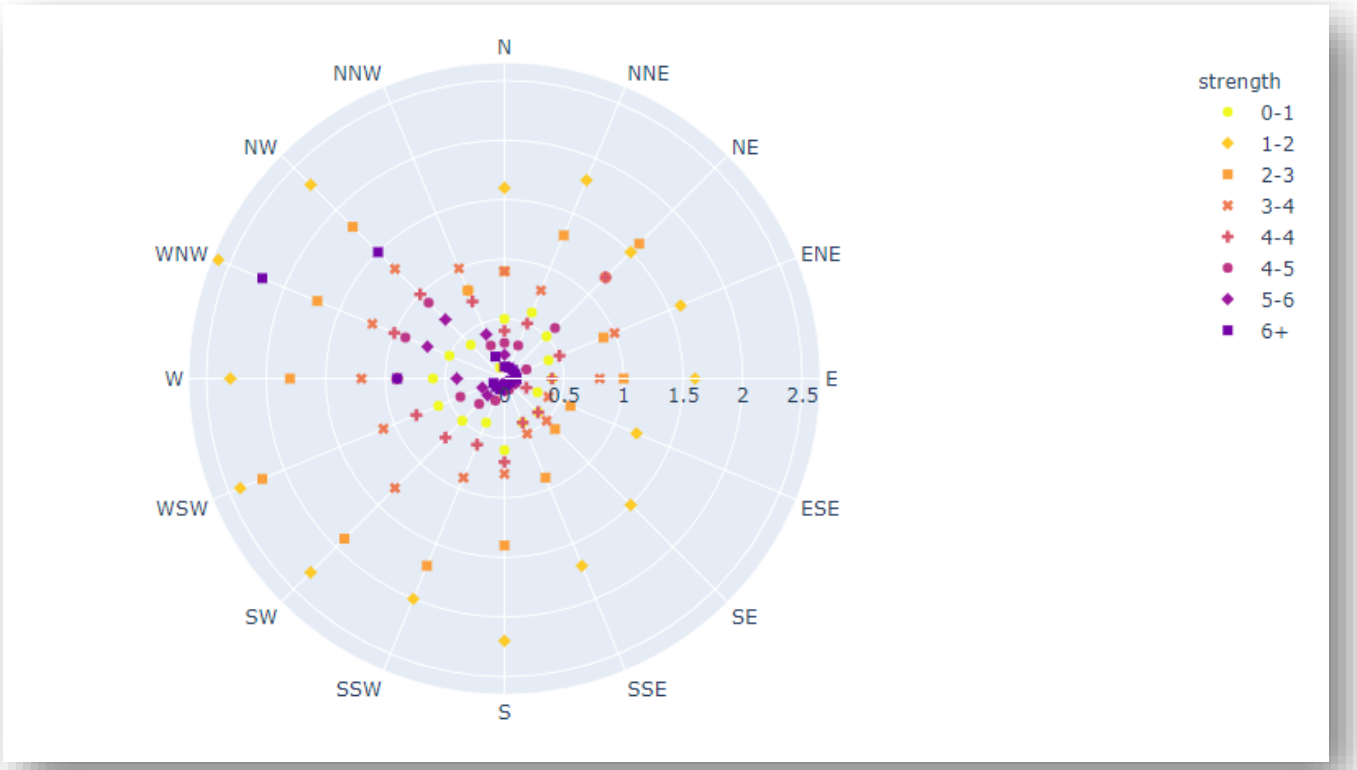
# Polar Plots

- **px.scatter_polar() functions**
  - Represents data along radial and angular axes

```
df = px.data.wind()

fig = px.scatter_polar(df,
                       r="frequency",
                       theta="direction",
                       color="strength",
                       symbol="strength",

color_discrete_sequence=px.colors.sequential.Plasma_r)

fig.show()
```

# Polar Plots

- px.scatter_polar() functions



# Homework for Lecture 6

- **Submit your source code for the following task:**
  1. Use any of the dataset provided by Plotly Express (Slide 3)
  2. Draw two graphs provided by Plotly Express
  3. Explain meaning of each plot result

- **Submission: source code, result screenshots and result explanation**

# 감사합니다!