Plotly express 기본

학습 내용

- plotly express에 대해 알아본다.

plotly express 소개

- plotly express는 plotly 라이브러리에 내장되어 있음.
- plotly express는 다양한 유형의 수치를 위한 30개 이상의 함수를 제공

기능

- Basic: scatter, line, area, bar, ...
- pie(파이)
- 1D 분포 : histogram, box, violin, strip
- 2D 분포 : density_heatmap
- 행렬 입력 : imshow
- 3차원 : scatter_3d, line_3d
- ...등

In [8]:

```
import plotly
import pandas as pd
import sys
```

프로그램 버전 확인

In [9]:

```
print(sys.version)
print(plotly.__version__)
print(pd.__version__)
```

```
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
5.1.0
1.1.3
```

Plot express 사용한 시각화

- cufflinks보다 좀 더 다양하며, 사용방법은 seaborn과 비슷함.
- plotly_express 이용. plotly 4.1 부터는 별도 설치 없어도 됨. 3.8.1의 경우 설치 필요

In [10]:

```
import plotly.express as px
```

In [11]:

```
# iris 데이터 불러오기
print(px.data.iris.__doc__)
px.data.iris().head()
```

```
Each row represents a flower.

https://en.wikipedia.org/wiki/Iris_flower_data_set (https://en.wikipedia.org/wiki/Ir
is_flower_data_set)

Returns:
    A `pandas.DataFrame` with 150 rows and the following columns:
    `['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species', 'spec
ies_id']`.
```

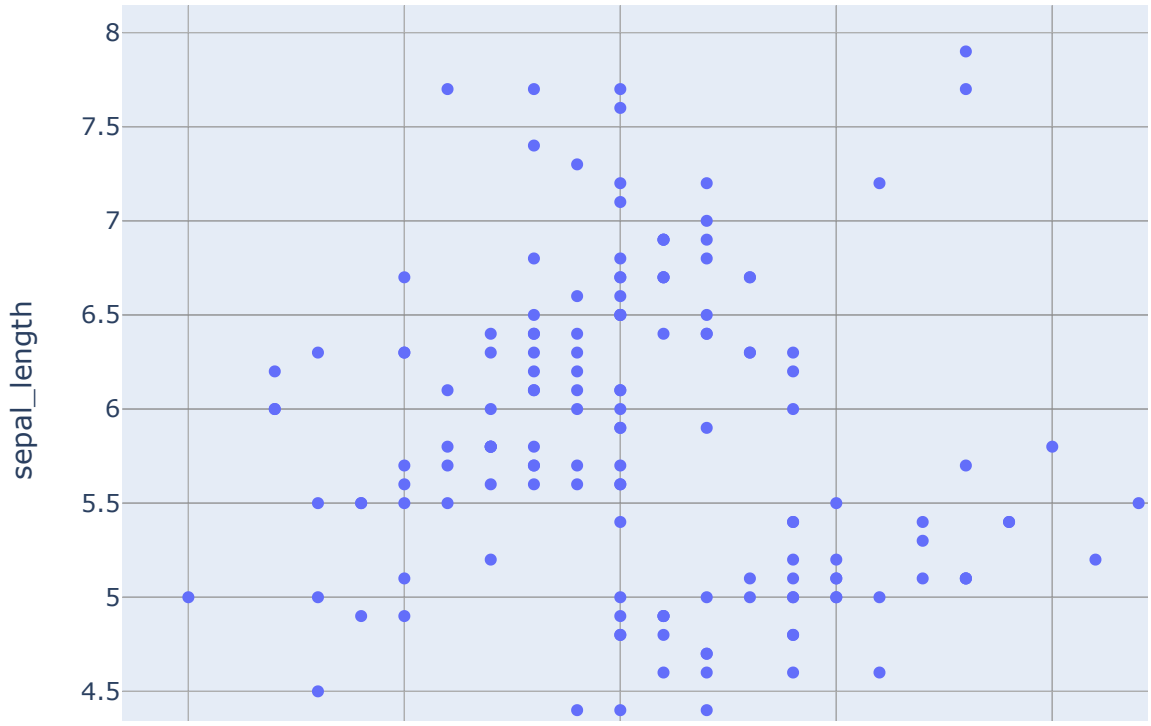
Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	species	species_id
0	5.1	3.5	1.4	0.2	setosa	1
1	4.9	3.0	1.4	0.2	setosa	1
2	4.7	3.2	1.3	0.2	setosa	1
3	4.6	3.1	1.5	0.2	setosa	1
4	5.0	3.6	1.4	0.2	setosa	1

산점도 및 선 그래프

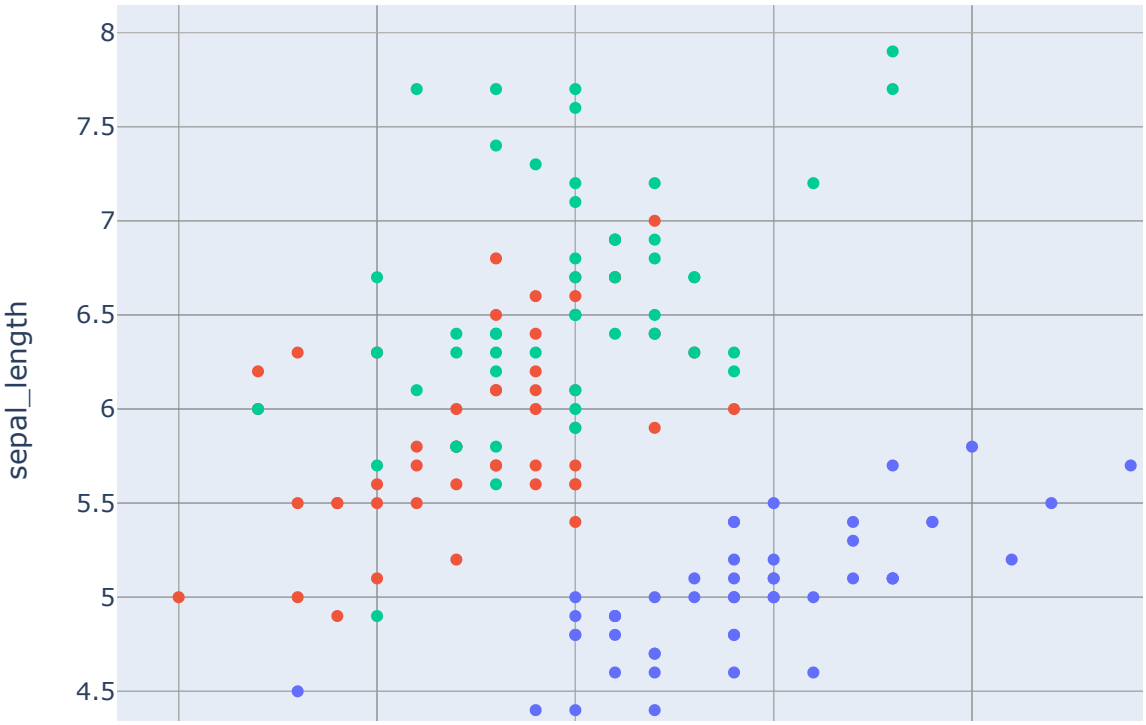
In [13]:

```
iris = px.data.iris()
fig = px.scatter(iris, x="sepal_width", y="sepal_length")
fig.show()
```



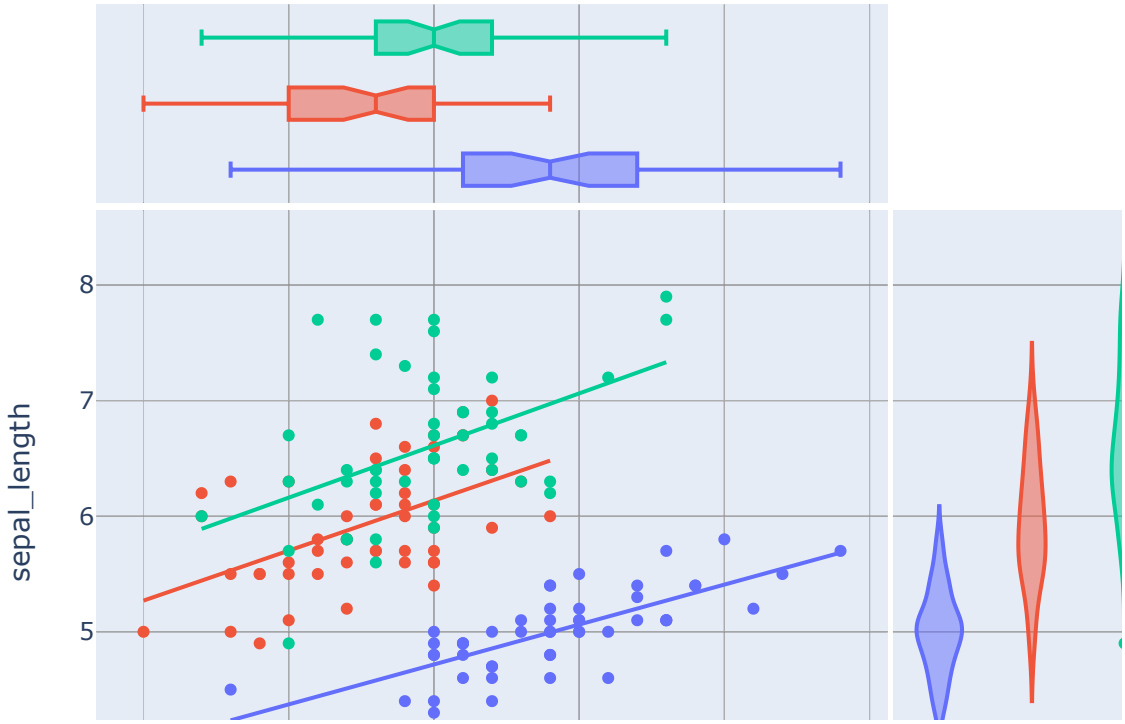
In [14]:

```
fig = px.scatter(iris, x="sepal_width", y="sepal_length", color="species")
fig.show()
```



In [16]:

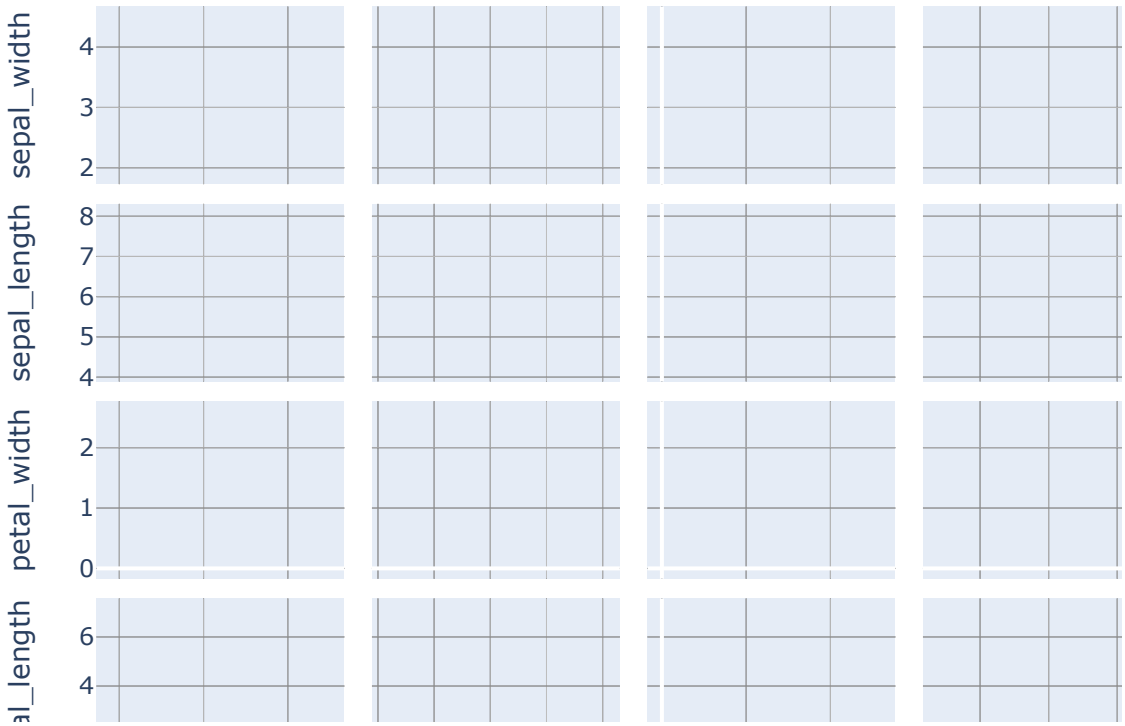
```
fig = px.scatter(iris,
         x="sepal_width", y="sepal_length",
         color="species",
         marginal_y="violin",
         marginal_x="box",
         trendline="ols")
fig.show()
```



산점도 행렬

In [18]:

```
fig = px.scatter_matrix(df, dimensions=["sepal_width",
                                        "sepal_length",
                                       "petal_width",
                                       "petal_length"],
                                       color="species")
fig.show()
```



다차원 범주형 데이터 시각화

- 데이터 집합의 각 변수는 직사각형 열로 표시
- 직사각형은 해당 변수에 의해 취해지는 이산형 값에 해당

In [25]:

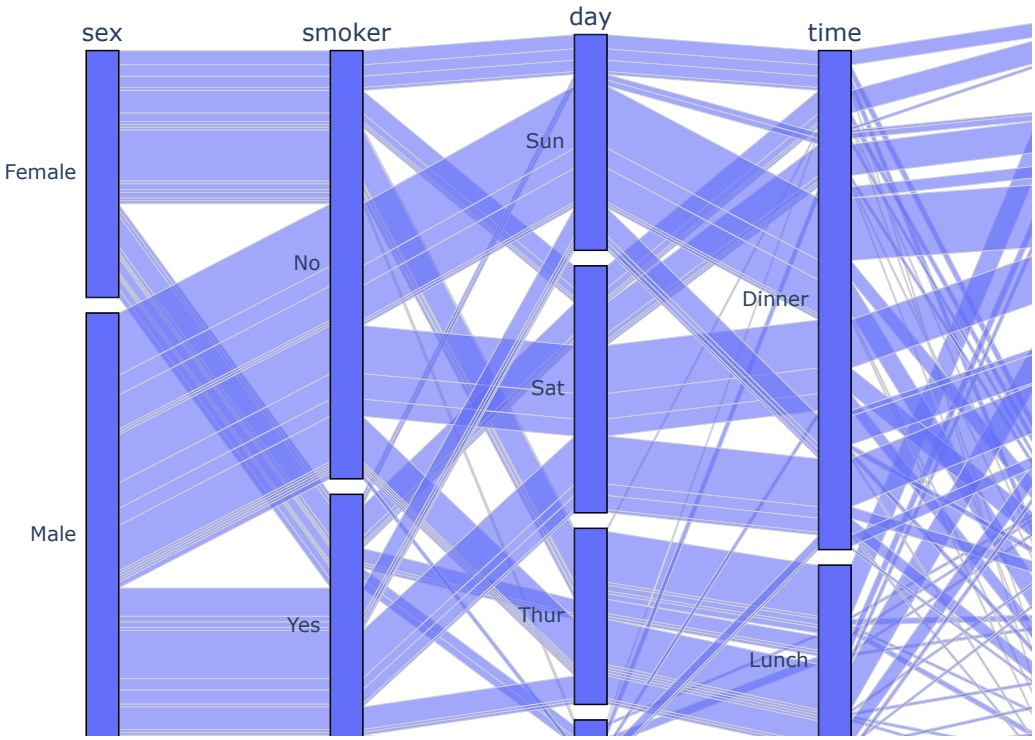
```
tips = px.data.tips()
tips['size'].unique()
```

Out[25]:

```
array([2, 3, 4, 1, 6, 5], dtype=int64)
```

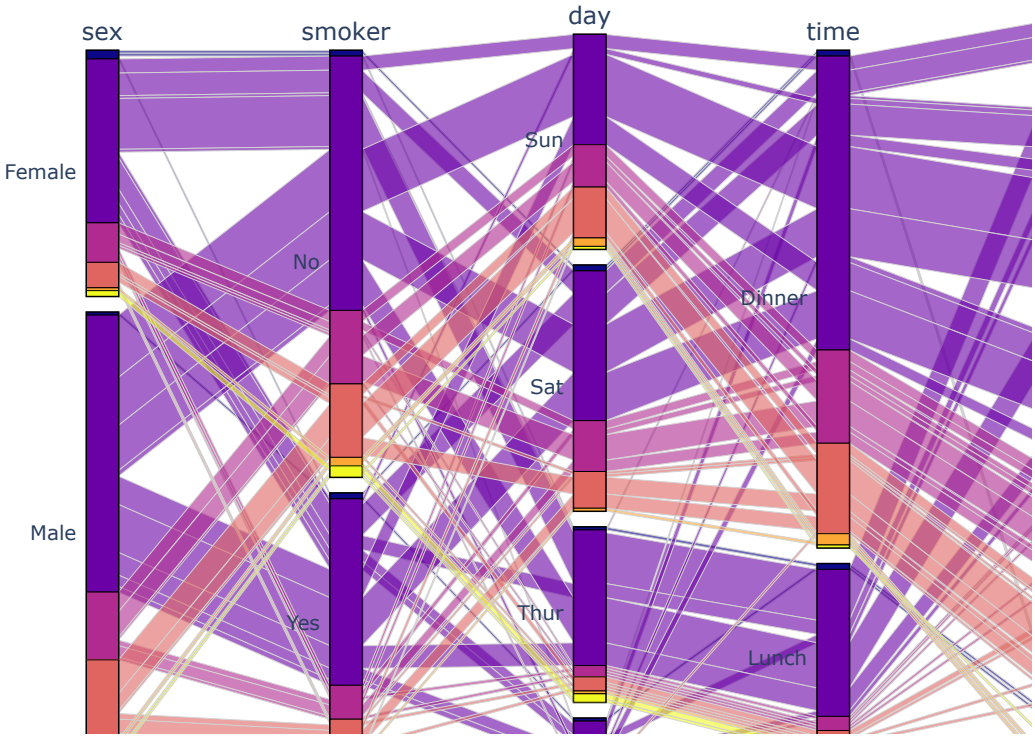
In [28]:

```
fig = px.parallel_categories(tips)
fig.show()
```



In [31]:

```
fig = px.parallel_categories(tips, color="size")
fig.show()
```



gapminder 데이터 셋 시각화

- gapminder 데이터 셋 ?
  - 국가별 경제 수준과 의료 동향 수준을 정리한 DataSet이다.

In [32]:

```
df = px.data.gapminder()
print(df.shape)
print(df.columns)
print(px.data.gapminder.__doc__)
```

```
(1704, 8)
Index(['country', 'continent', 'year', 'lifeExp', 'pop', 'gdpPercap',
       'iso_alpha', 'iso_num'],
      dtype='object')

Each row represents a country on a given year.

https://www.gapminder.org/data/ (https://www.gapminder.org/data/)

Returns:
    A `pandas.DataFrame` with 1704 rows and the following columns:
    `['country', 'continent', 'year', 'lifeExp', 'pop', 'gdpPercap',
    'iso_alpha', 'iso_num']`.
```

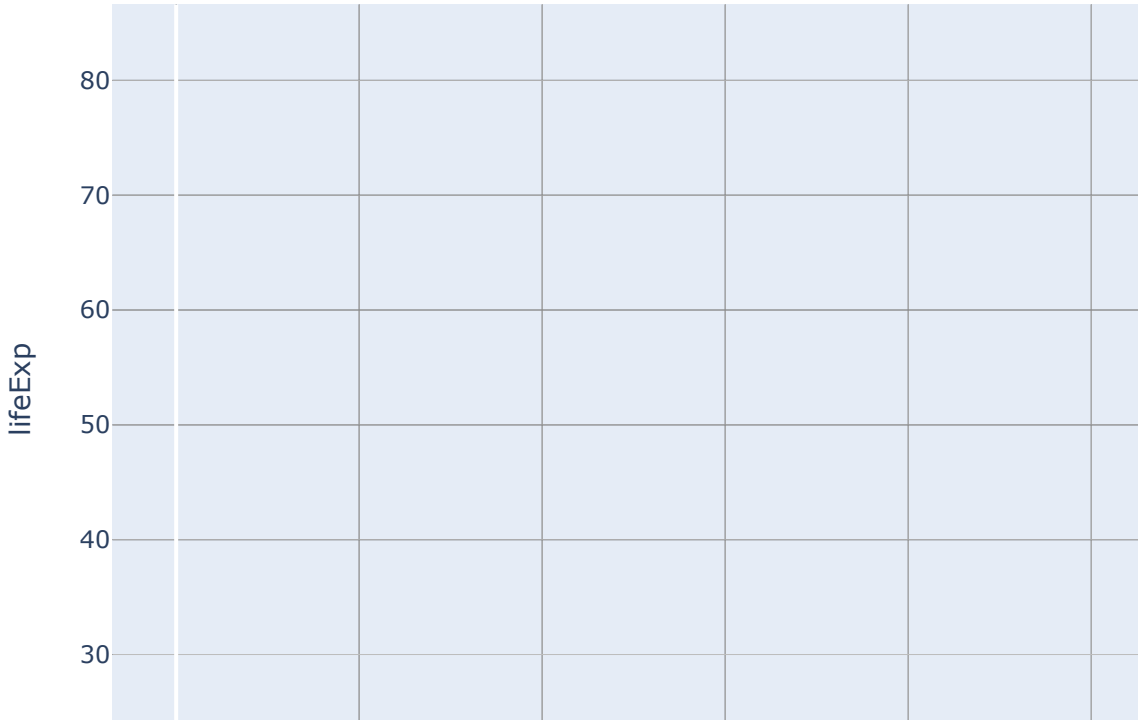
데이터 설명

- continent : 대륙
- country : 나라
- gdpPercap : 1인당 국민소득
- pop : 인구
- lifeExp : 기대수명

1인당 국민소득과 기대 수명은 어떤 관계를 가질까?

In [38]:

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

fig = px.scatter(gapminder, x="gdpPercap", y="lifeExp")
fig.show()
```



1인당 국민소득과 기대 수명은 어떤 관계를 가질까?(2007년)

In [40]:

```
gapminder.describe()
```

Out[40]:

	year	lifeExp	pop	gdpPercap	iso_num
count	1704.00000	1704.000000	1.704000e+03	1704.000000	1704.000000
mean	1979.50000	59.474439	2.960121e+07	7215.327081	425.880282
std	17.26533	12.917107	1.061579e+08	9857.454543	248.305709
min	1952.00000	23.599000	6.001100e+04	241.165877	4.000000
25%	1965.75000	48.198000	2.793664e+06	1202.060309	208.000000
50%	1979.50000	60.712500	7.023596e+06	3531.846989	410.000000
75%	1993.25000	70.845500	1.958522e+07	9325.462346	638.000000
max	2007.00000	82.603000	1.318683e+09	113523.132900	894.000000

다중 그래프 그리기

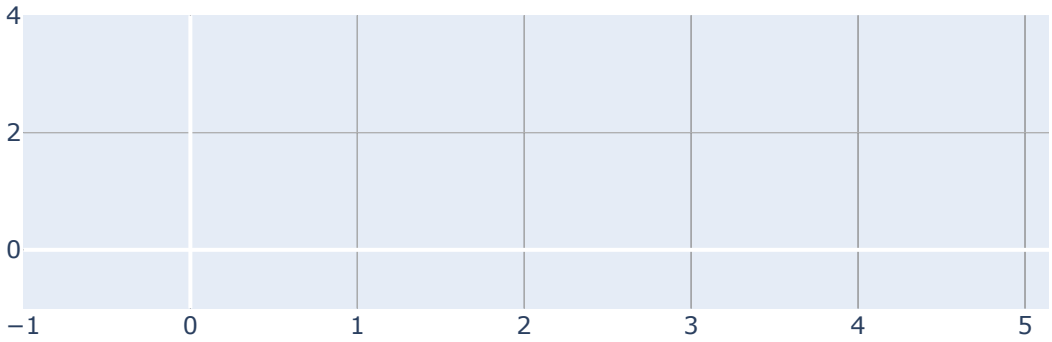
- plotly은 다음과 같음.
  - 01 make_subplots 이용하여 틀(레이아웃 만들기)
  - 02 add_trace 을 이용하여 레이아웃 위에 그래프 그리기
  - 03 update_layout을 이용하여 축제목 등을 추가하기

In [84]:

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

In [85]:

```
fig = make_subplots(rows=2, cols=1, shared_yaxes=True)
fig
```

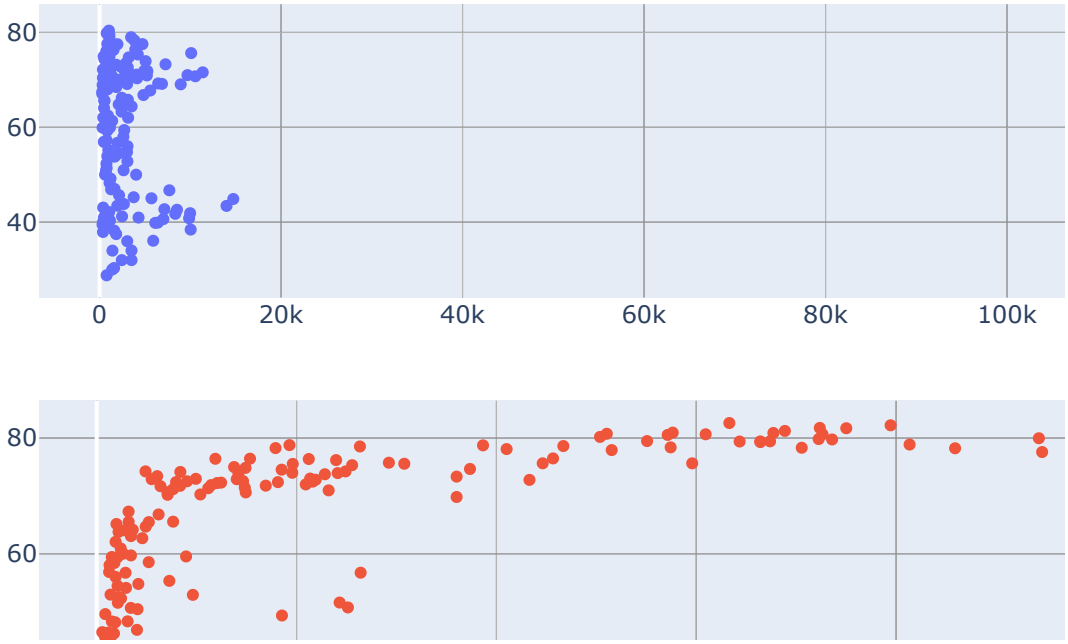


In [78]:

```
gap_1952 = gapminder.query("year==1952")
gap_2007 = gapminder.query("year==2007")

gap_1952_line = go.Scatter(x=gap_1952["gdpPercap"],
                           y=gapminder["lifeExp"],
                           mode='markers')

gap_2007_line = go.Scatter(x=gap_2007["gdpPercap"],
                           y=gap_2007["lifeExp"],
                           mode='markers')
```

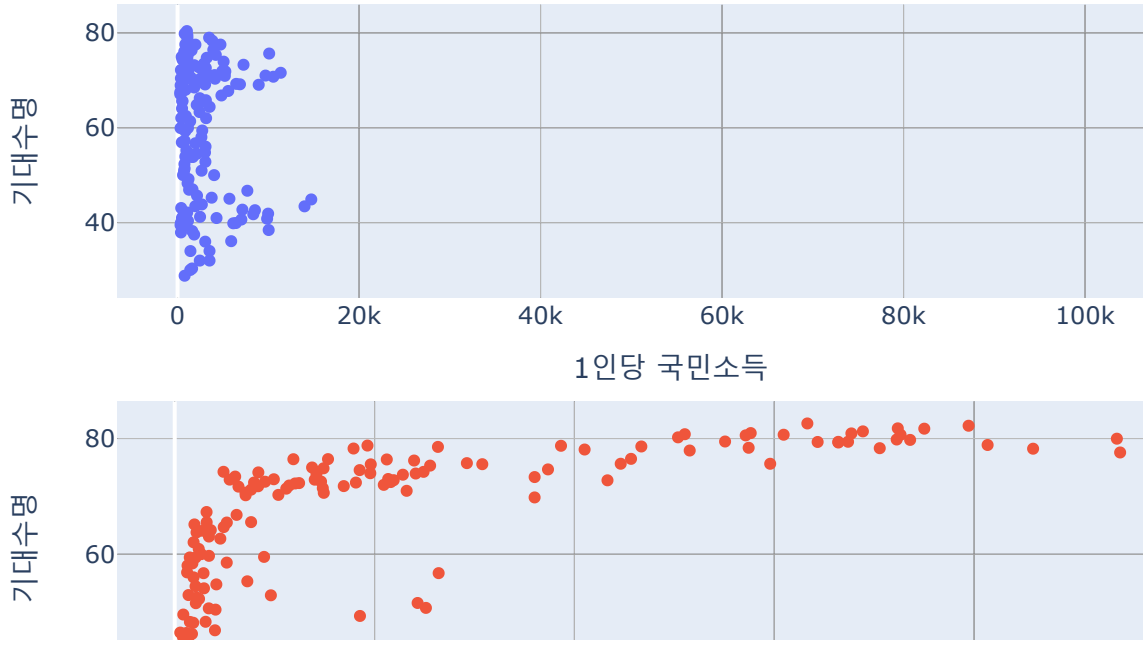
In [79]:

```
fig.add_trace(gap_1952_line, row=1, col=1)
fig.add_trace(gap_2007_line, row=2, col=1)
```



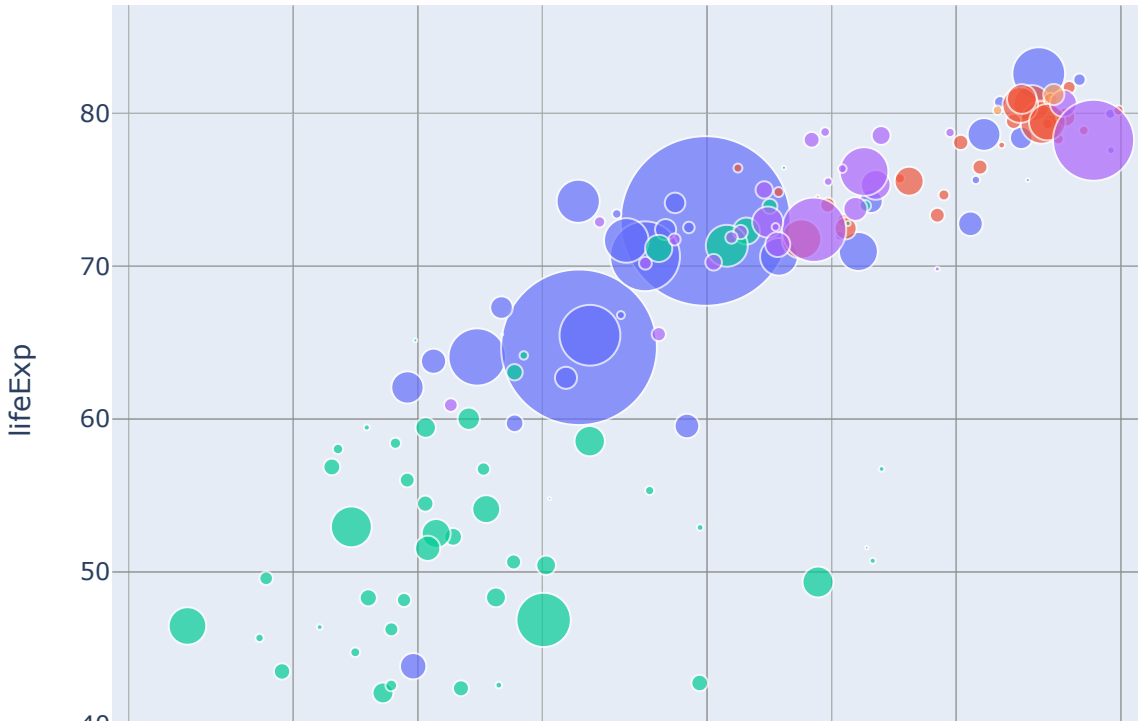
In [83]:

```
layout = {
    "xaxis": {
        "title": "1인당 국민소득",
    },
    "yaxis": {
        "title": "기대수명"
    },

    "xaxis2": {
        "title": "1인당 국민소득",
    },
    "yaxis2": {
        "title": "기대수명"
    }
}

fig.update_layout(layout)
```



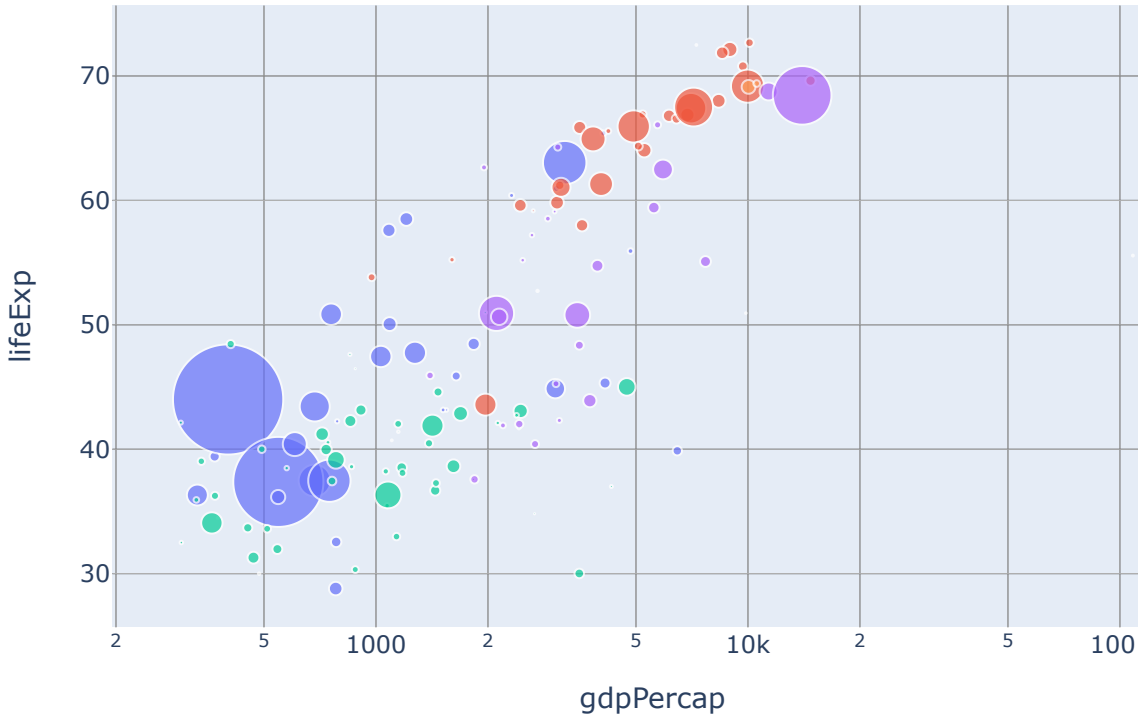
In [87]:

```
gapminder = px.data.gapminder()
fig = px.scatter(gapminder.query("year==2007"),
          x="gdpPercap",
          y="lifeExp",
          size="pop",
          color="continent",     # 색 구분 : 대륙
          hover_name="country",  # 마우스 클릭 나라명
          log_x=True, size_max=60)
fig.show()
```



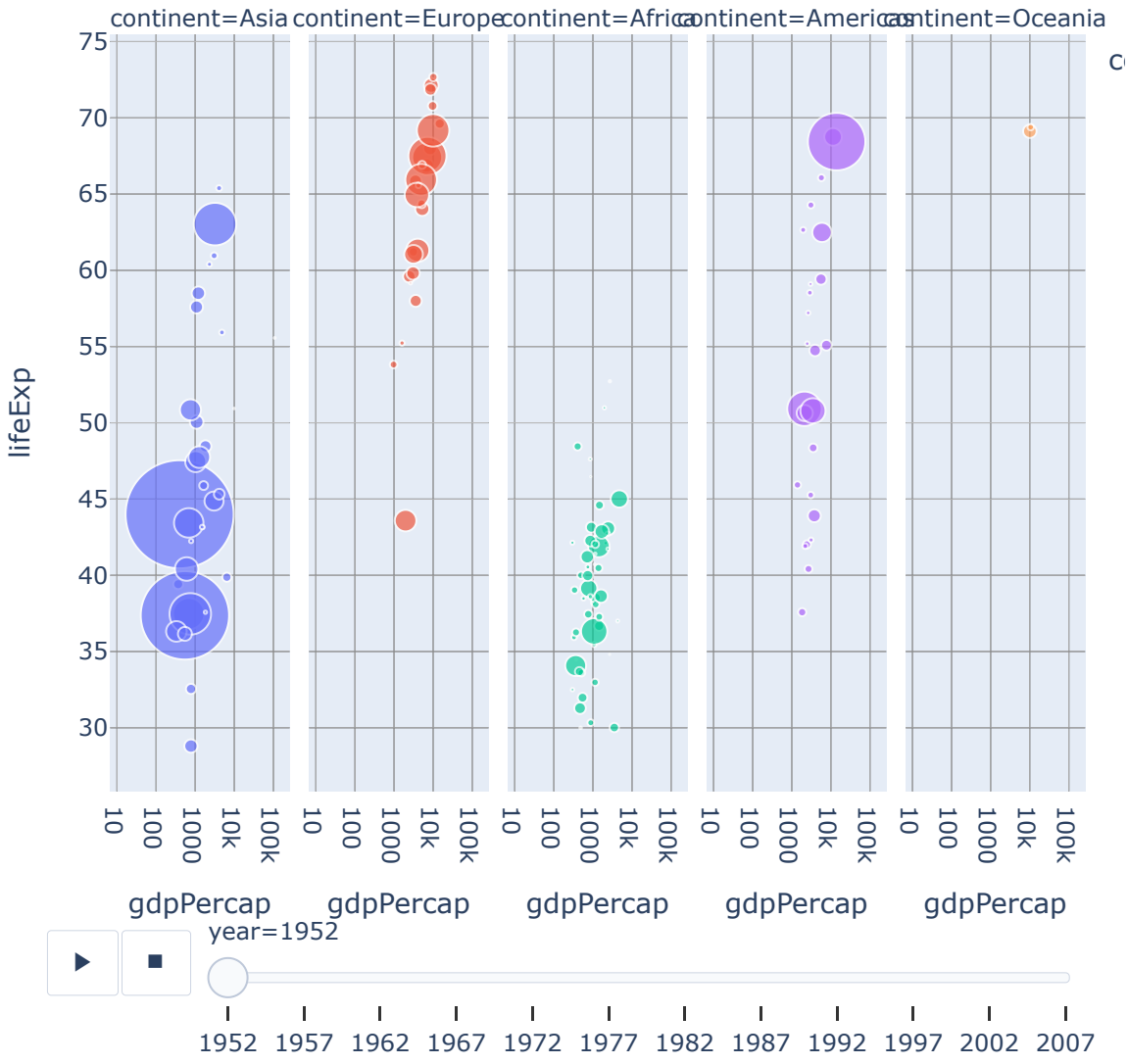
In [90]:

```
fig = px.scatter(gapminder,
          animation_frame="year",
          animation_group="country",
          x="gdpPercap",
          y="lifeExp",
          size="pop",
          color="continent",
          hover_name="country", log_x=True, size_max=60)
fig
```



In [95]:

```
import plotly.express as px
px.defaults.width = 700
px.defaults.height = 600

fig = px.scatter(gapminder,
           animation_frame="year",
           animation_group="country",
           x="gdpPercap",
           y="lifeExp",
           size="pop",
           color="continent",
           hover_name="country",
           facet_col="continent",
           log_x=True, size_max=60)

fig
```



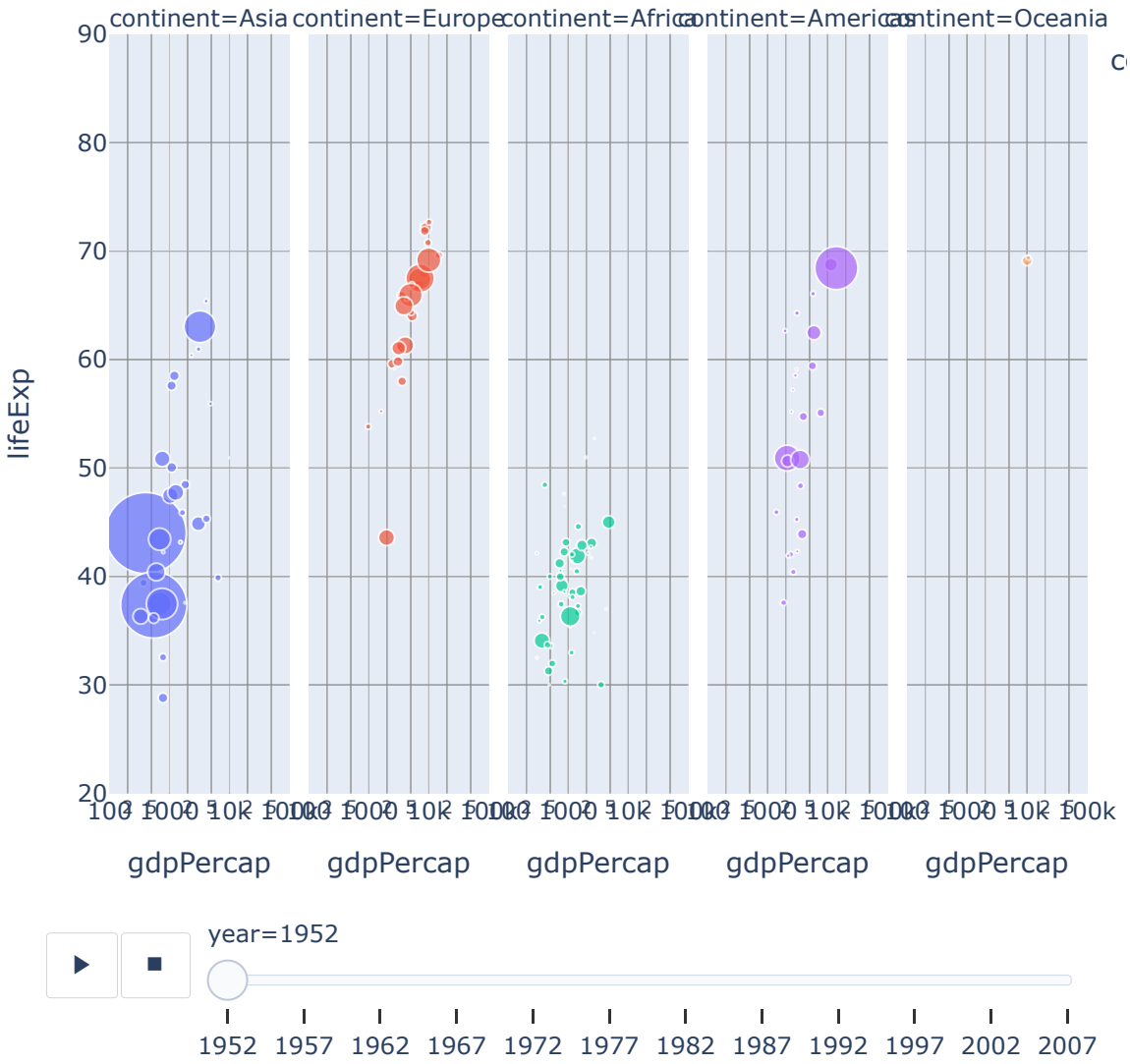
In [97]:

```
gapminder.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1704 entries, 0 to 1703
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   country    1704 non-null   object 
 1   continent  1704 non-null   object 
 2   year       1704 non-null   int64  
 3   lifeExp    1704 non-null   float64
 4   pop        1704 non-null   int64  
 5   gdpPercap  1704 non-null   float64
 6   iso_alpha  1704 non-null   object 
 7   iso_num    1704 non-null   int64  
dtypes: float64(2), int64(3), object(3)
memory usage: 106.6+ KB
```

In [96]:

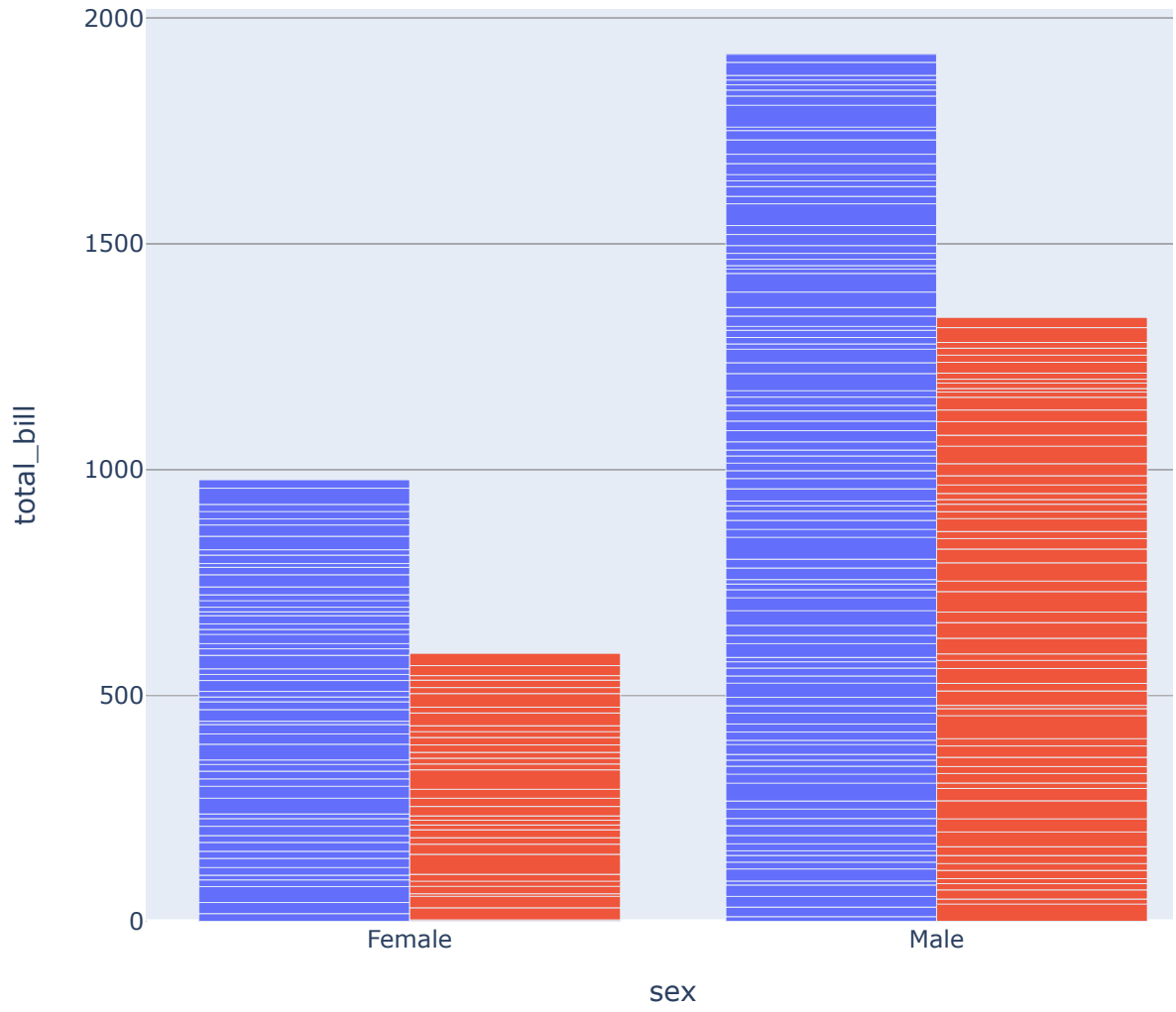
```
fig = px.scatter(gapminder, x="gdpPercap", y="lifeExp",
           animation_frame="year",
           animation_group="country",
           size="pop",
           color="continent",
           hover_name="country",
           facet_col="continent",
           log_x=True, size_max=45, range_x=[100,100000], range_y=[20,90])
fig.show()
```



막대 그래프

In [99]:

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



In [100]:

```
df = px.data.election()
print(df.shape)
print(df.head())
print(df.columns)
print(px.data.election.__doc__)
```

```
(58, 8)
                district  Coderre  Bergeron  Joly  total    winner     result  ₩
0     101-Bois-de-Liesse     2481      1829  3024   7334      Joly  plurality
1  102-Cap-Saint-Jacques     2525      1163  2675   6363      Joly  plurality
2   11-Sault-au-Récollet     3348      2770  2532   8650   Coderre  plurality
3          111-Mile-End     1734      4782  2514   9030  Bergeron   majority
4        112-DeLorimier     1770      5933  3044  10747  Bergeron   majority

   district_id
0          101
1          102
2           11
3          111
4          112
Index(['district', 'Coderre', 'Bergeron', 'Joly', 'total', 'winner', 'result',
       'district_id'],
      dtype='object')

Each row represents voting results for an electoral district in the 2013 Montreal
mayoral election.

Returns:
    A `pandas.DataFrame` with 58 rows and the following columns:
    `['district', 'Coderre', 'Bergeron', 'Joly', 'total', 'winner', 'result', 'distr
ict_id']`.
```

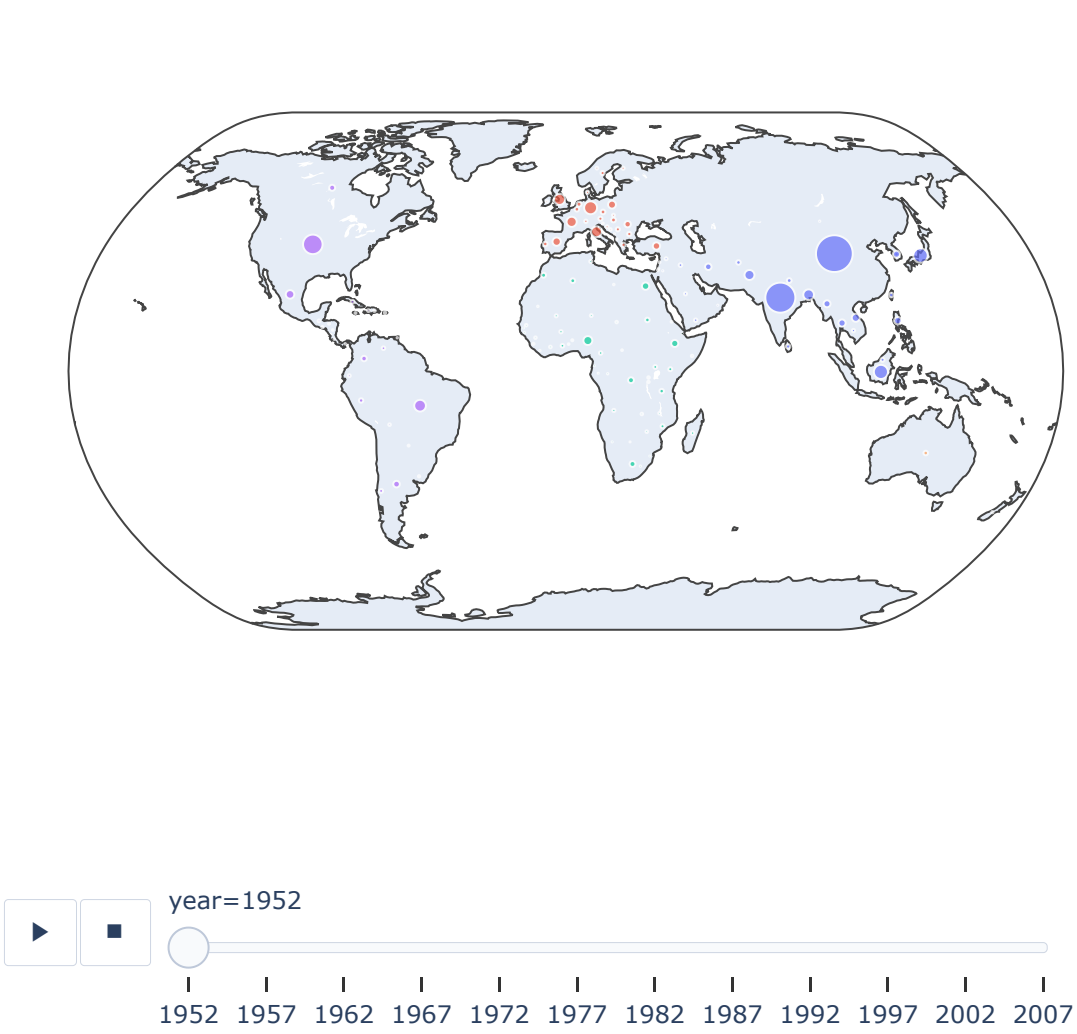
In [101]:

```
import plotly.express as px
election = px.data.election()
fig = px.line_3d(election,
                 x="Joly", y="Coderre", z="Bergeron",
                 color="winner", line_dash="winner")
fig.show()
```

Maps

In [102]:

```
fig = px.scatter_geo(gapminder,
                     locations="iso_alpha",
                     color="continent",
                     hover_name="country",
                     size="pop",
                    animation_frame="year", projection="natural earth")
fig.show()
```



In [103]:

```
import plotly.express as px
df = px.data.gapminder().query("year == 2007")
fig = px.line_geo(df, locations="iso_alpha",
                  color="continent", # "continent" is one of the columns of gapminder
                  projection="orthographic")
fig.show()
```

REF

- cufflinks.datagen module
- https://jpoles1.github.io/cufflinks/html/cufflinks.datagen.html (https://jpoles1.github.io/cufflinks/html/cufflinks.datagen.html) (https://jpoles1.github.io/cufflinks/html/cufflinks.datagen.html (https://jpoles1.github.io/cufflinks/html/cufflinks.datagen.html))
- Plotly Express in Python
- https://plot.ly/python/plotly-express/#plotly-express (https://plot.ly/python/plotly-express/#plotly-express) (https://plot.ly/python/plotly-express/#plotly-express (https://plot.ly/python/plotly-express/#plotly-express))