

인터랙티브 시각화(plotly)

최수연 교수

mibm400@hanmail.net

학습목표

- 인터랙티브 시각화에 대해 설명할 수 있다.
- plotly 라이브러리를 사용하여 인터랙티브 차트를 생성하고 활용할 수 있다.

목차

- 인터랙티브 시각화의 정의
- plotly 라이브러리의 정의
- plotly 라이브러리를 활용한 차트 생성
- plotly 라이브러리를 활용하여 버튼 생성하기

인터랙티브 시각화란?

- 마우스 움직임에 따라 실시간으로 모양이 변하는 그래프
- 그래프를 자유롭게 조작하면서 관심 부분을 자세히 살펴 볼 수 있음
- HTML 포맷으로 저장하면 일반 사용자도 웹 브라우저에서 그래프 조작이 가능
- 인터랙티브 시각화를 지원하는 라이브러리

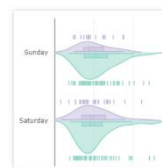
`import plotly`

plotly 라이브러리

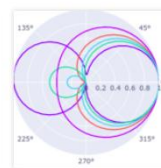
- 인터랙티브 시각화를 지원하는 라이브러리
- JavaScript 기반 시각화 라이브러리
- 마우스를 차트에 올려놓으면 툴팁처럼 실제 데이터 값을 확인 할 수 있음
- 그래프를 확대/축소 가능함
- HTML 로 변환하여 웹 상에서도 확인 가능함
- 약 40여종의 차트를 지원함

<https://plotly.com/python/>

Fundamentals



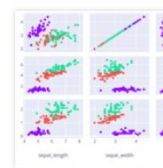
The Figure Data Structure



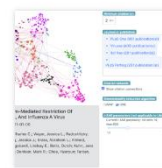
Creating and Updating Figures



Displaying Figures

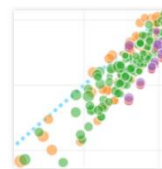


Plotly Express



Analytical Apps with Dash

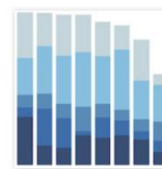
Basic Charts



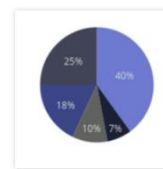
Scatter Plots



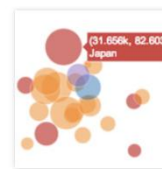
Line Charts



Bar Charts

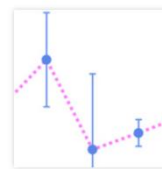


Pie Charts



Bubble Charts

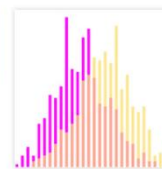
Statistical Charts



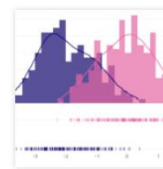
Error Bars



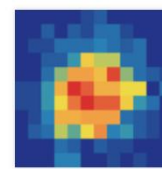
Box Plots



Histograms



Distplots



2D Histograms

More Statistical Charts »

plotly를 활용하여 그래프를 그리는 방법

◆ **graph_objects** 모듈을 사용하는 방법

- 그래프를 세세하게 구성할 때 사용
- low-level interface 를 제공
- 코드가 길고 문법이 복잡

◆ **express** 모듈을 사용하는 방법

- 간단한 코드로 쉽게 표현 가능한 high-level interface 를 제공
- 정해진 템플릿 외에 세세한 표현 어려움
- 세세한 표현을 위해서는 graph_objects와 병행해서 사용

plotly 라이브러리 사용

- 라이브러리 설치
- ✓ colab의 경우 이미 설치되어 있음
- plotly 라이브러리 선언

`pip install plotly`

`pip install jupyter-dash`

- ✓ `express` 모듈 선언

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

- ✓ `graph_objects` 모듈 선언

```
import plotly.graph_objects as go
```

seboarn 라이브러리의 iris 데이터셋

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

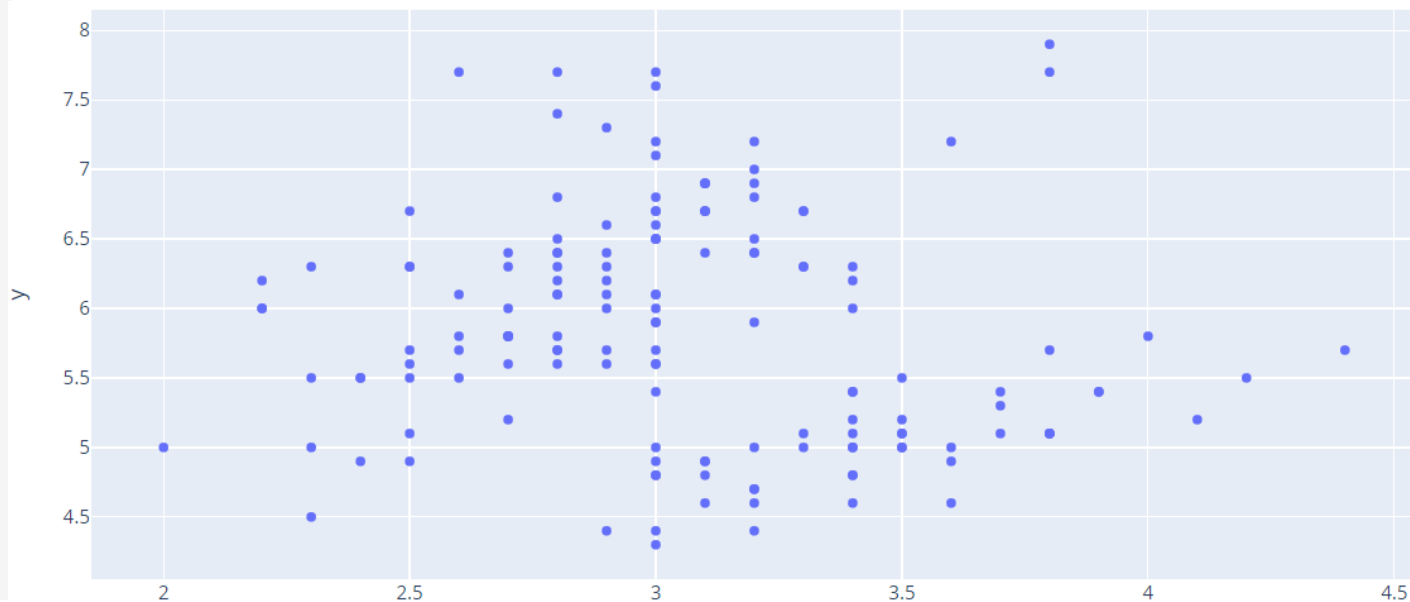
열이름	내용
sepal_length	꽃받침의 길이
sepal_width	꽃받침의 너비
petal_length	꽃잎의 길이
petal_width	꽃잎의 너비
species	붓꽃 종류(setosa, versicolor, virginica)

scatter 차트

- `px.scatter(data_frame=표이름, x='x축열이름', y='y축열이름')`
- ✓ `data_frame=표이름` : 차트를 생성할 데이터프레임 선택 (생략가능)
- ✓ `x='x축열이름'` : 수치형 데이터를 갖는 x축 열이름
- ✓ `y='y축열이름'` : 수치형 데이터를 갖는 y축 열이름

```
px.scatter(data_frame=iris, x="sepal_width", y="sepal_length")
```

```
px.scatter(x=iris["sepal_width"], y=iris["sepal_length"])
```



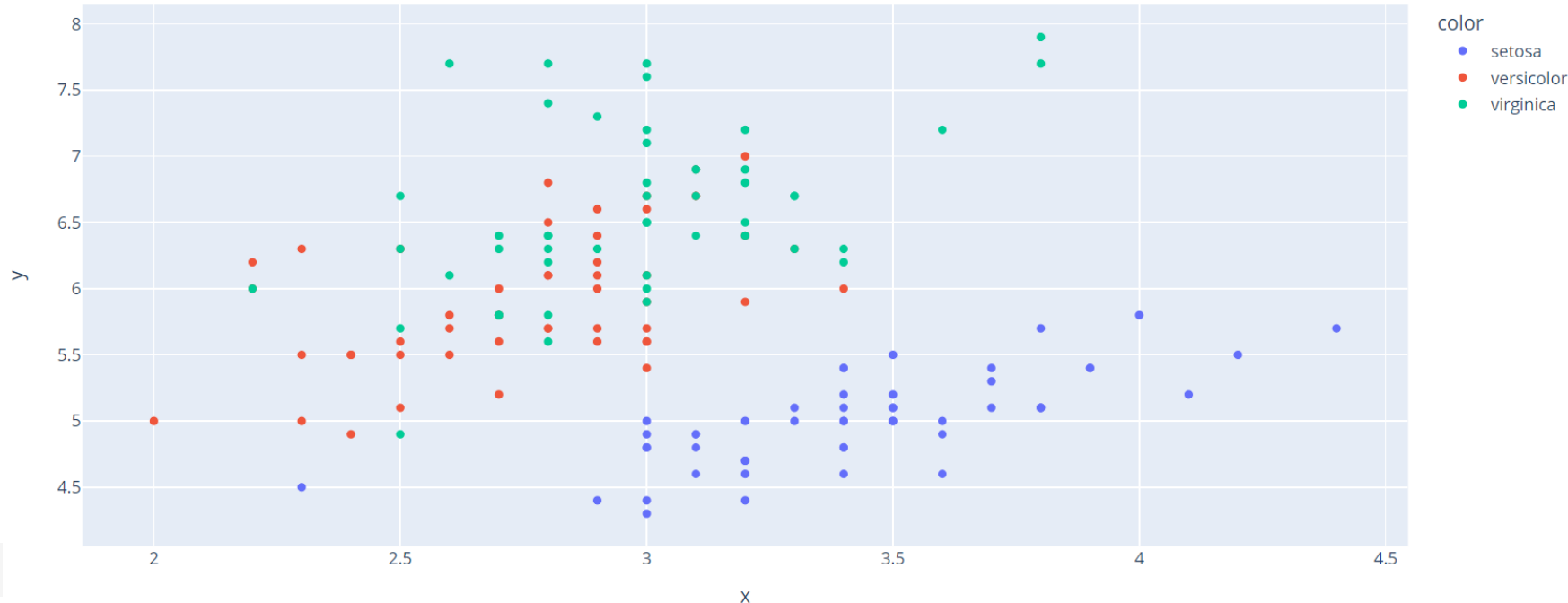
scatter 차트에서 종류 구분하기

- `px.scatter(data=표이름, x='x축열이름', y='y축열이름', color='열이름');`

✓ `color='열이름'`: 종류를 구분하기 위한 데이터의 열이름

- 열이름이 범주형일 경우에는 색상으로 표시
- 열이름이 수치형일 경우에는 그라데이션으로 표시

```
px.scatter(x=iris["sepal_width"], y=iris["sepal_length"], color=iris['species'])
```



scatter 차트에서 종류 구분하기

```
px.scatter(x=iris['sepal_length'], y=iris['sepal_width'], color=iris['petal_length'])
```

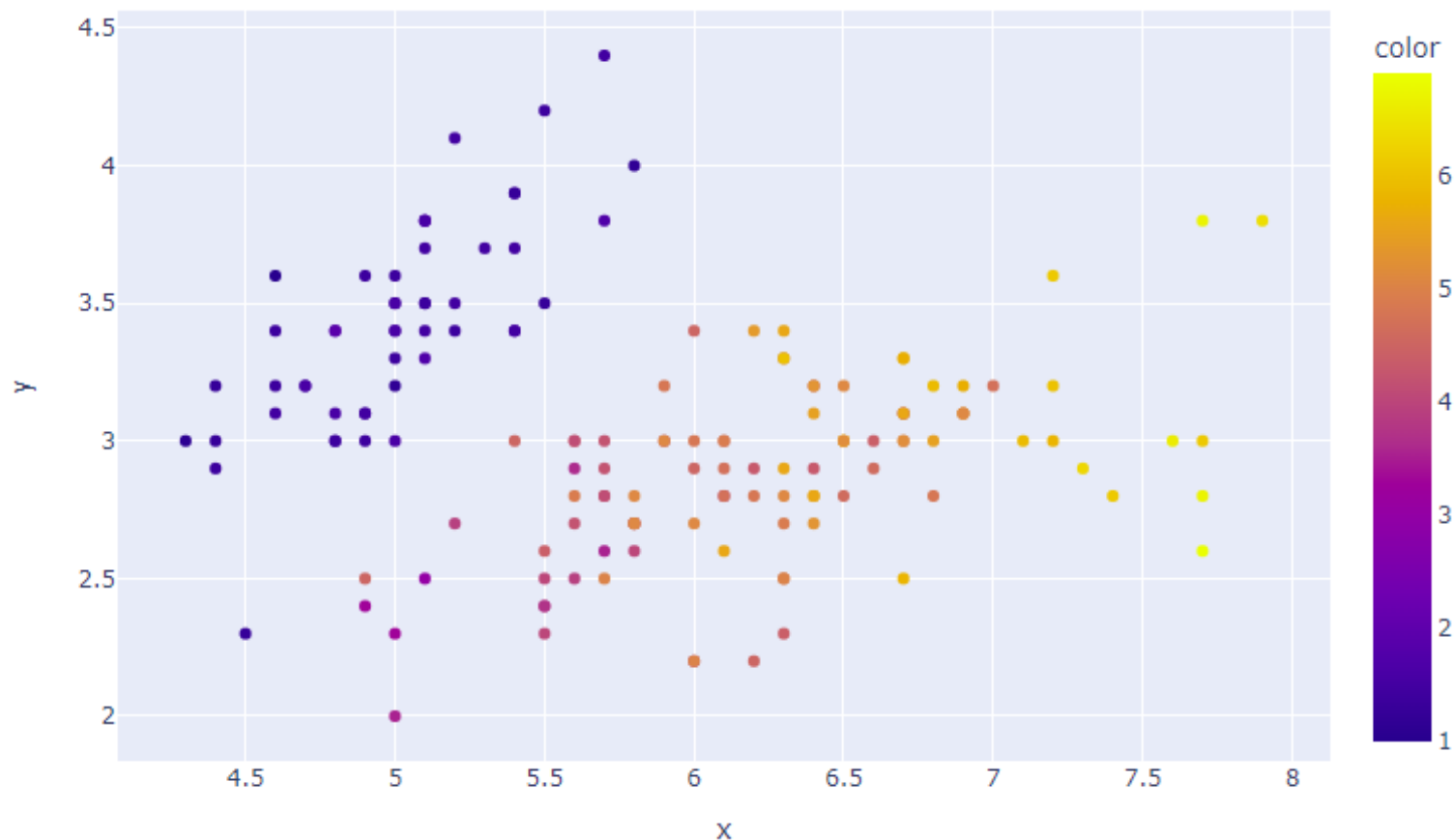


차트 꾸미기

- `fig.update_layout(title_text='')`

✓ `title_text=""` : 차트제목

- `fig.update_xaxes(title='')`

✓ x축 레이블 속성 변경

✓ `title_text=''` : x축 제목 설정

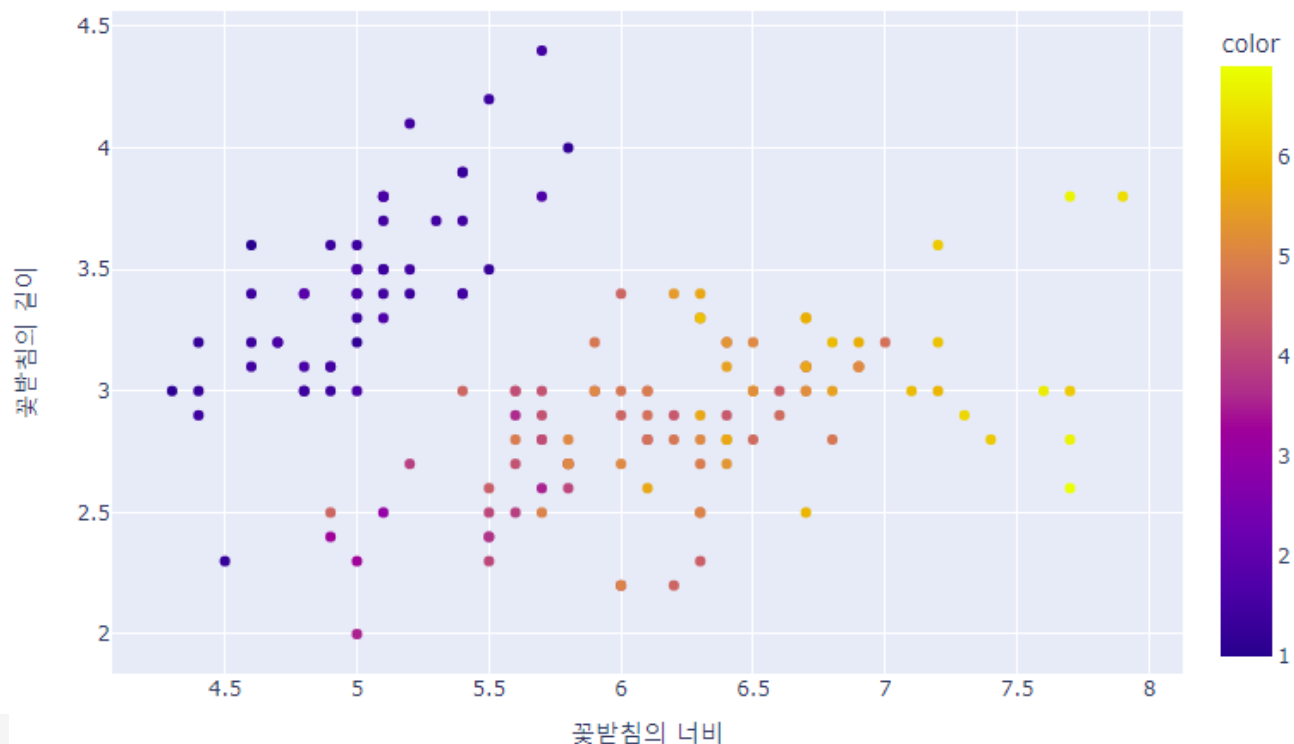
- `fig.update_yaxes(title='')`

✓ y축 레이블 속성 변경

차트 꾸미기

```
fig = px.scatter(x=iris['sepal_length'], y=iris['sepal_width'], color=iris['petal_length'])  
fig.update_layout(title_text = 'Iris 종류')  
fig.update_xaxes(title='꽃받침의 너비')  
fig.update_yaxes(title='꽃받침의 길이')
```

Iris 종류



2001~2023년 인구동향

```
인구동향 = pd.read_excel('/content/인구동향(출생,사망,혼인,이혼통계).xlsx')
인구동향
```

	시점	출생아수 (명)	사망자수 (명)	혼인건수 (건)	이혼건수 (건)
0	2001-01-01	11961	3305	6242	1940
1	2001-02-01	10159	3030	5454	2358
2	2001-03-01	10971	3397	6751	2630
3	2001-04-01	9950	3355	6825	2452
4	2001-05-01	9649	3227	8221	2591
...
267	2023-04-01	3062	3998	2660	987
268	2023-05-01	3281	4333	3262	1186
269	2023-06-01	3141	4001	3087	1002
270	2023-07-01	3298	4088	2771	1050
271	2023-08-01	3249	4406	2739	1013

272 rows x 5 columns

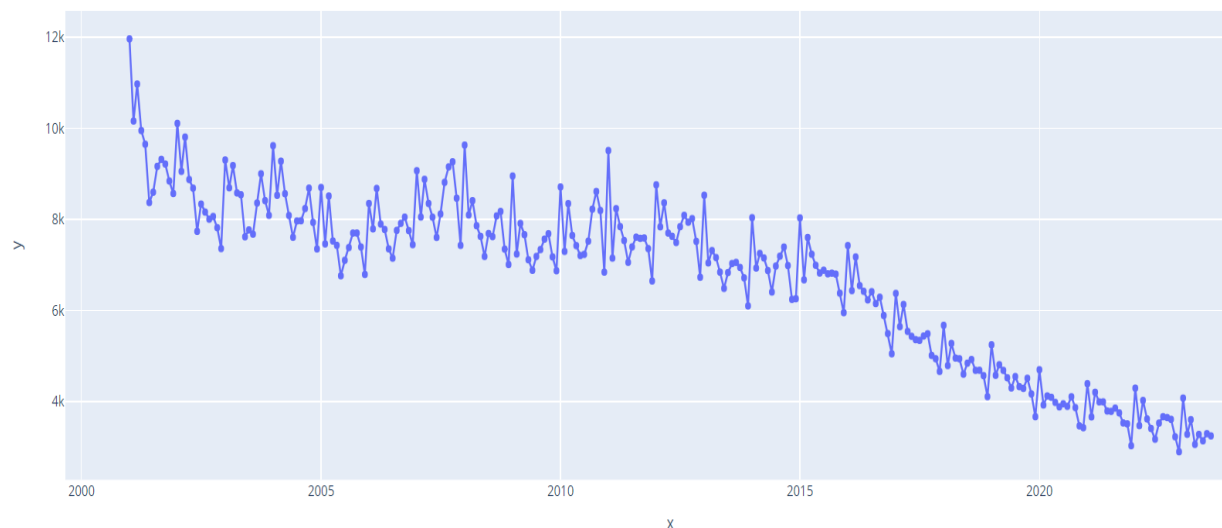
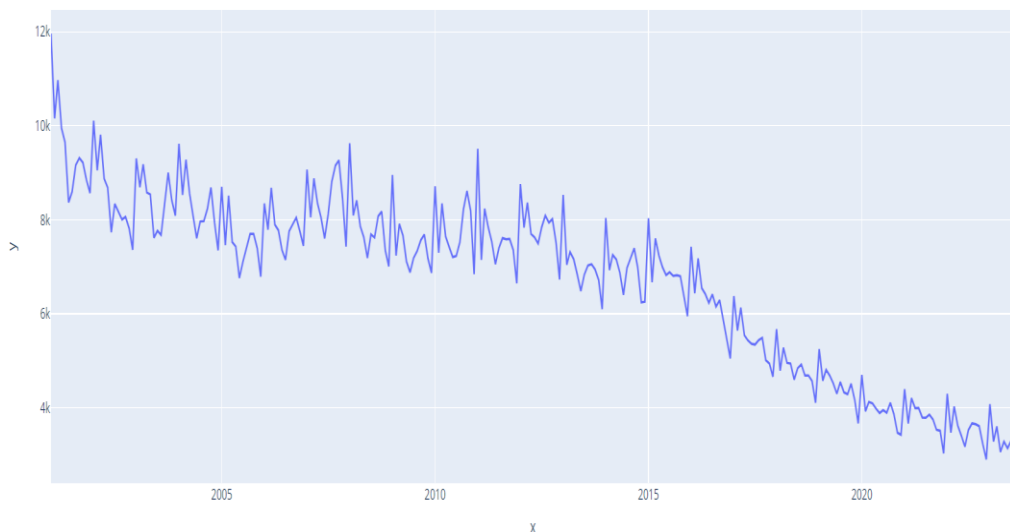
```
인구동향.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 272 entries, 0 to 271
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   시점                  272 non-null   datetime64[ns]
 1   출생아수 (명)         272 non-null   int64  
 2   사망자수 (명)         272 non-null   int64  
 3   혼인건수 (건)         272 non-null   int64  
 4   이혼건수 (건)         272 non-null   int64  
dtypes: datetime64[ns](1), int64(4)
memory usage: 10.8 KB
```

line 차트

- `px.line(data_frame=표이름, x='x축열이름', y='y축열이름', markers=True)`
- ✓ `data_frame=표이름` : 차트를 생성할 데이터프레임 선택 (생략가능)
- ✓ `x='x축열이름'` : 수치형 데이터를 갖는 x축 열이름
- ✓ `y='y축열이름'` : 수치형 데이터를 갖는 y축 열이름
- ✓ `markers=True` : 선차트에 marker 표시

```
px.line(x=인구동향['시점'], y=인구동향['출생아수 (명)']) px.line(x=인구동향['시점'], y=인구동향['출생아수 (명)'], markers=True)
```

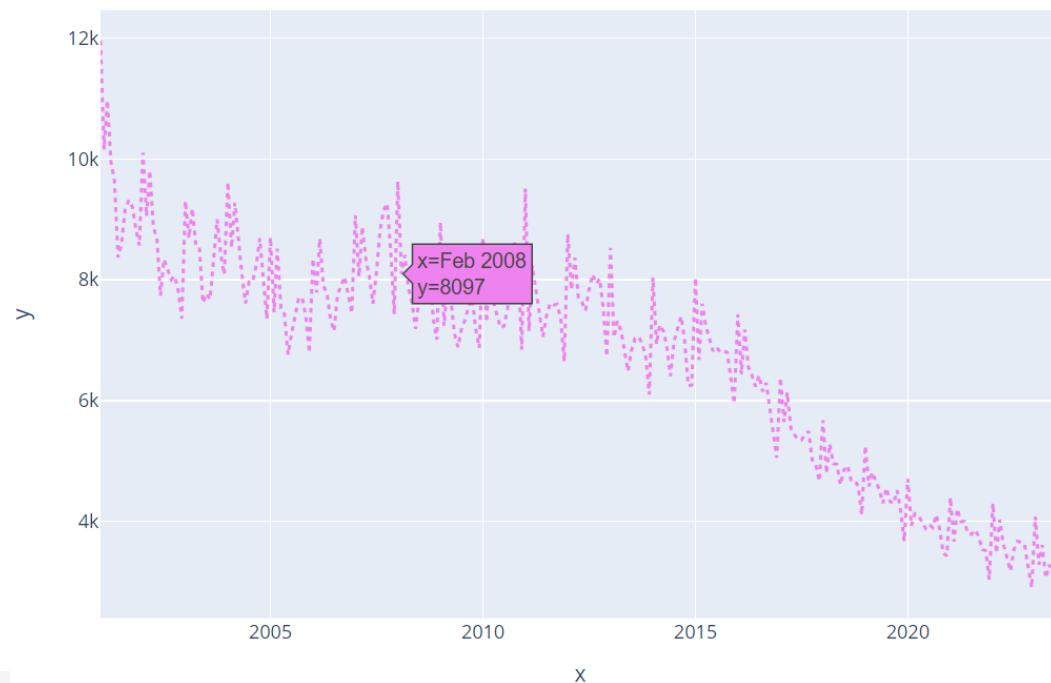


line 차트 선 변경하기

- `fig.update_traces`(선의 속성들)

- ✓ `line_color = ''` : 선 색 설정
- ✓ `line_width=2` : 선의 두께 설정
- ✓ `line_dash='dash'` : 선 종류 설정(dash, dot, dashdot)

```
fig = px.line(x=인구동향['시점'], y=인구동향['출생아수 (명)'])  
fig.update_traces(line_color='violet', line_width=2, line_dash='dot')
```



2000~2022년 혼인건수 데이터

```
import pandas as pd
혼인건수 = pd.read_excel('/content/2000_2022 혼인건수.xlsx')
혼인건수.head()
```

	자치구별	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
0	종로구	1550	1426	1368	1354	1160	1155	1043	1162	1069	...	877	840	806	760	700	676	590	565	478	486
1	중구	1123	1192	991	968	1010	910	1035	1052	984	...	860	851	916	799	792	778	722	653	590	511
2	용산구	2046	1839	1654	1699	1669	1702	1842	1979	1959	...	1795	1560	1515	1366	1357	1400	1303	1175	995	948
3	성동구	2655	2975	2555	2578	2659	2686	2686	2763	2339	...	2432	2125	2083	1944	2096	2000	1758	1573	1272	1179
4	광진구	3281	3213	3216	3000	2844	2801	3059	3023	2725	...	2685	2589	2576	2285	2082	2164	1893	1661	1399	1317

5 rows x 24 columns

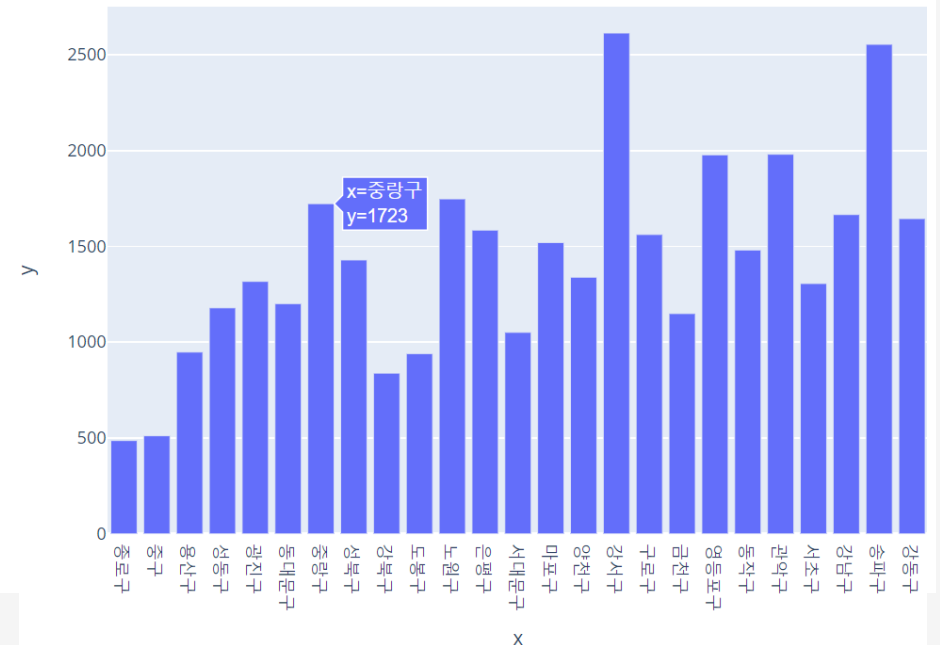
혼인건수.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 24 columns):
#   Column  Non-Null Count  Dtype  
---  -
0   자치구별  25 non-null      object  
1   2000      25 non-null      int64  
2   2001      25 non-null      int64  
3   2002      25 non-null      int64  
4   2003      25 non-null      int64  
5   2004      25 non-null      int64  
6   2005      25 non-null      int64  
7   2006      25 non-null      int64  
8   2007      25 non-null      int64  
9   2008      25 non-null      int64  
10  2009      25 non-null      int64  
11  2010      25 non-null      int64  
12  2011      25 non-null      int64  
13  2012      25 non-null      int64  
14  2013      25 non-null      int64  
15  2014      25 non-null      int64  
16  2015      25 non-null      int64  
17  2016      25 non-null      int64  
18  2017      25 non-null      int64  
19  2018      25 non-null      int64  
20  2019      25 non-null      int64  
21  2020      25 non-null      int64  
22  2021      25 non-null      int64  
23  2022      25 non-null      int64  
dtypes: int64(23), object(1)
memory usage: 4.8+ KB
```

bar 차트

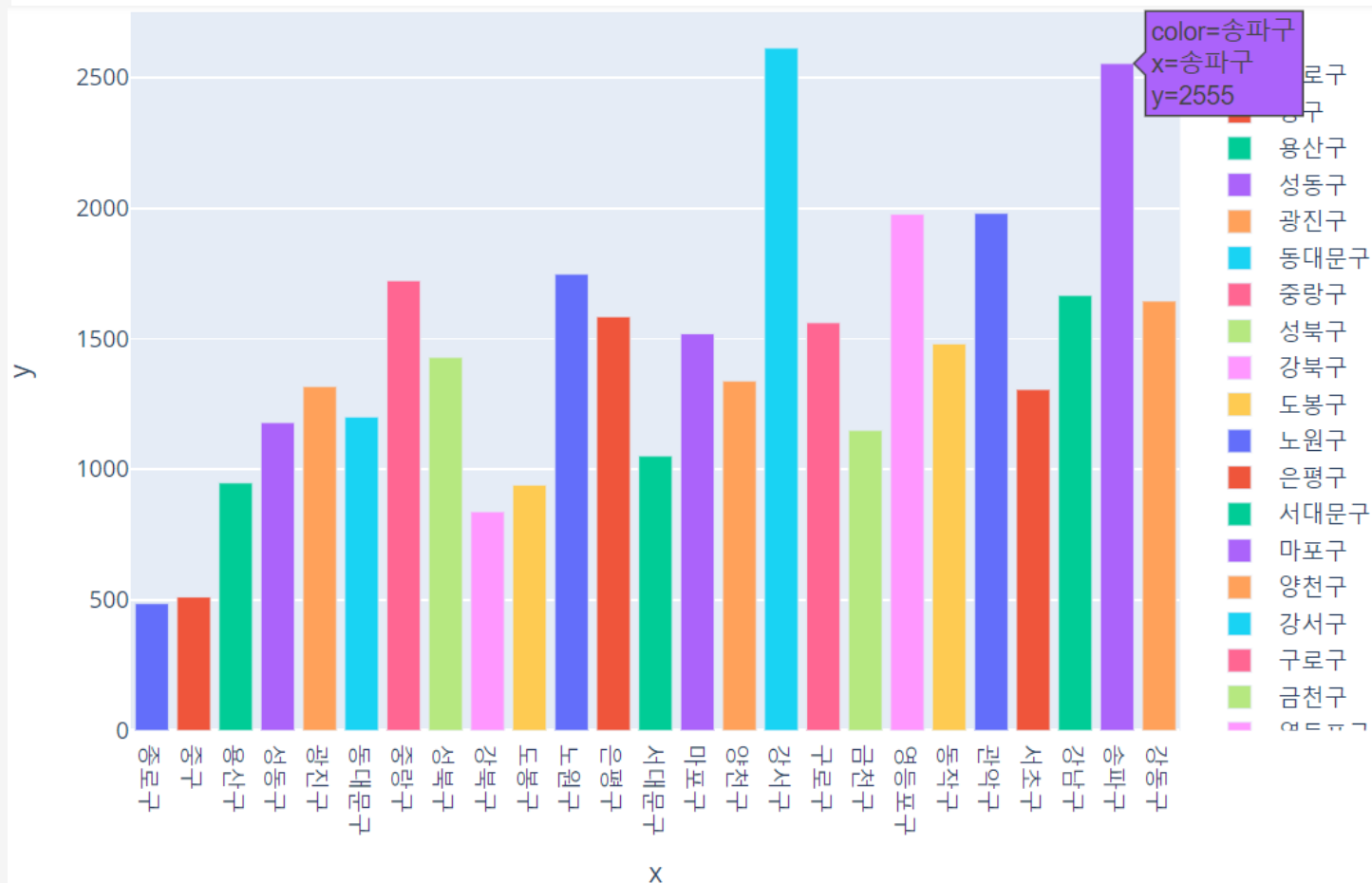
- `px.bar(data_frame=표이름, x='x축열이름', y='y축열이름', 속성들)`
 - ✓ `data_frame=표이름` : 차트를 생성할 데이터프레임 선택 (생략가능)
 - ✓ `x='x축열이름'` : 수치형 데이터를 갖는 x축 열이름
 - ✓ `y='y축열이름'` : 수치형 데이터를 갖는 y축 열이름
 - ✓ `color = '열이름'` : 종류를 구분하기 위한 데이터의 열이름
 - ✓ `text_auto=True` : 막대차트 내에 데이터 값 표시

```
px.bar(x=혼인건수['자치구별'], y=혼인건수['2022'])
```



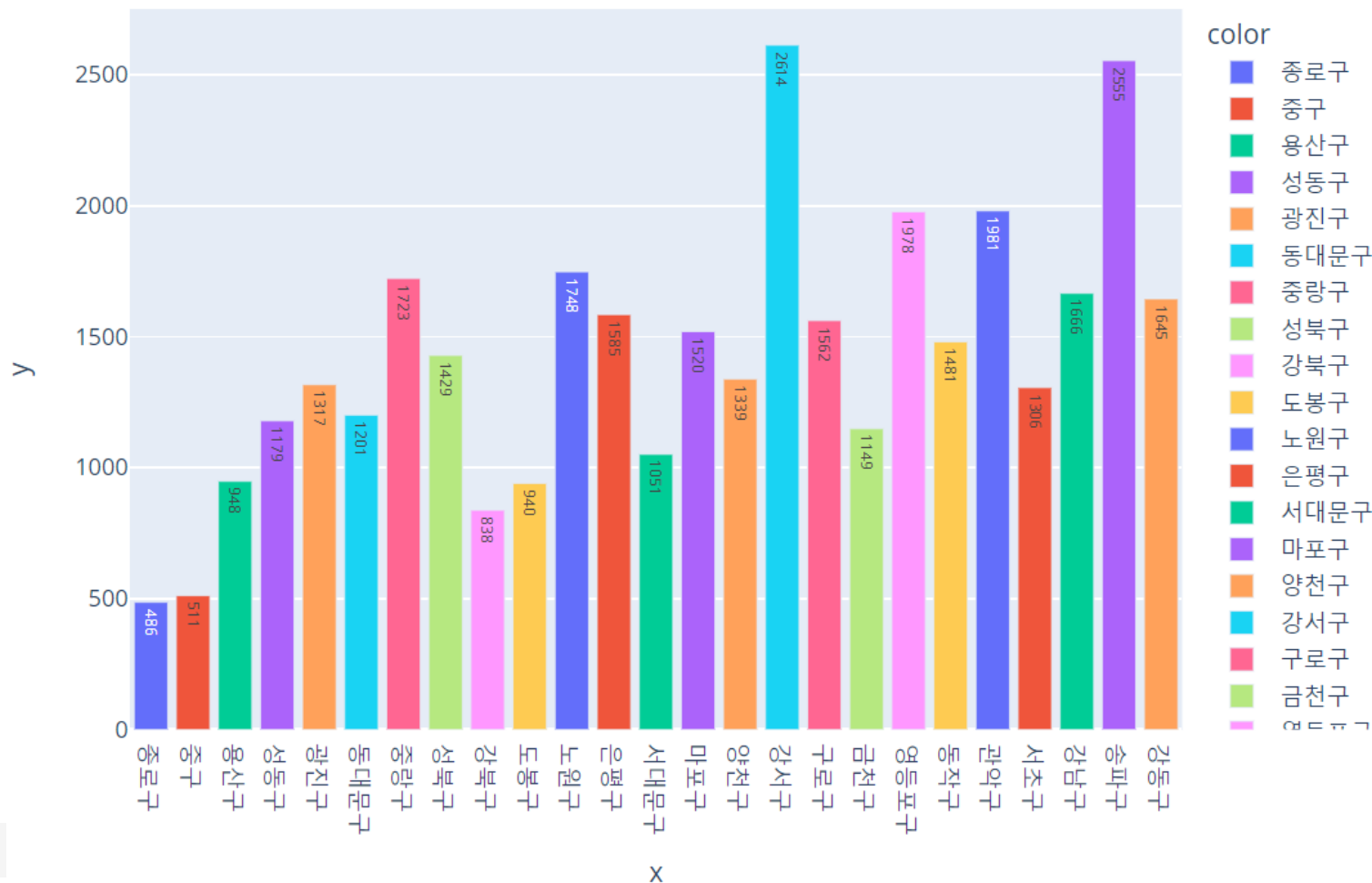
bar 차트에서 색상 지정하기

`px.bar(x=혼인건수['자치구별'], y=혼인건수['2022'], color=혼인건수['자치구별'])`



bar 차트에 데이터 표시하기

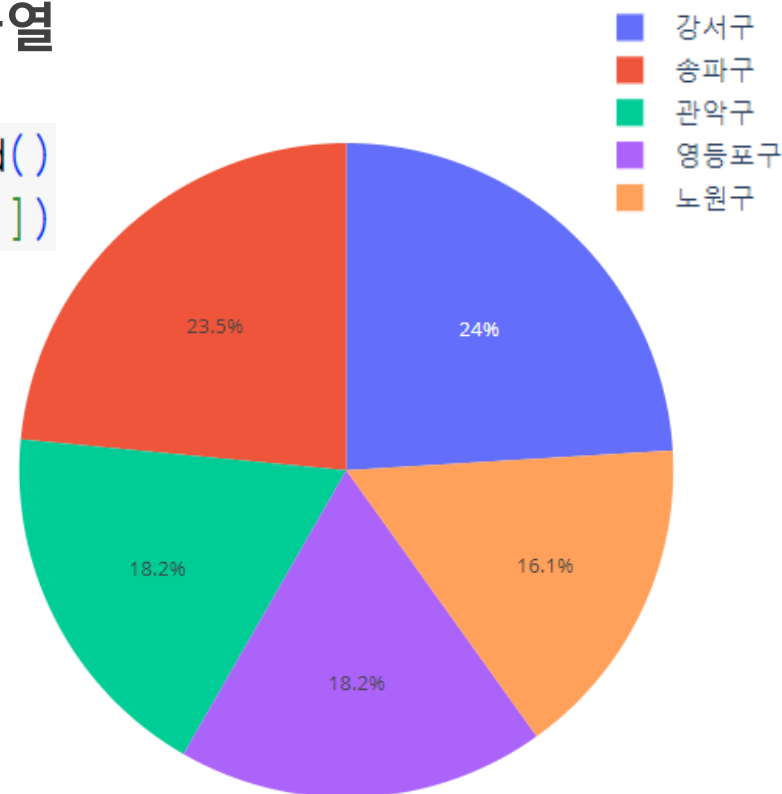
```
px.bar(x=혼인건수['자치구별'], y=혼인건수['2022'], color=혼인건수['자치구별'], text_auto=True)
```



pie 차트

- `px.pie(values=데이터, names='레이블명들')`
 - ✓ `valuse=데이터` : 데이터의 구성비를 확인하기 위한 데이터
 - ✓ `names='레이블명들'` : 각 영역의 값을 나타내는 레이블 문자열

```
혼인건수2022 = 혼인건수.sort_values('2022', ascending=False).head()  
px.pie(values=혼인건수2022['2022'], names=혼인건수2022['자치구별'])
```



pie 차트 도넛 모양으로 변경하기

- `fig.update_traces(hole=실수)`
 - ✓ 도넛 모양의 차트 설정
 - ✓ `hole=실수` : 0~1사이의 실수로 가운데 도넛 모양의 크기

```
fig = px.pie(values=혼인건수2022['2022'], names=혼인건수2022['자치구별'])  
fig.update_traces(hole=.3)
```

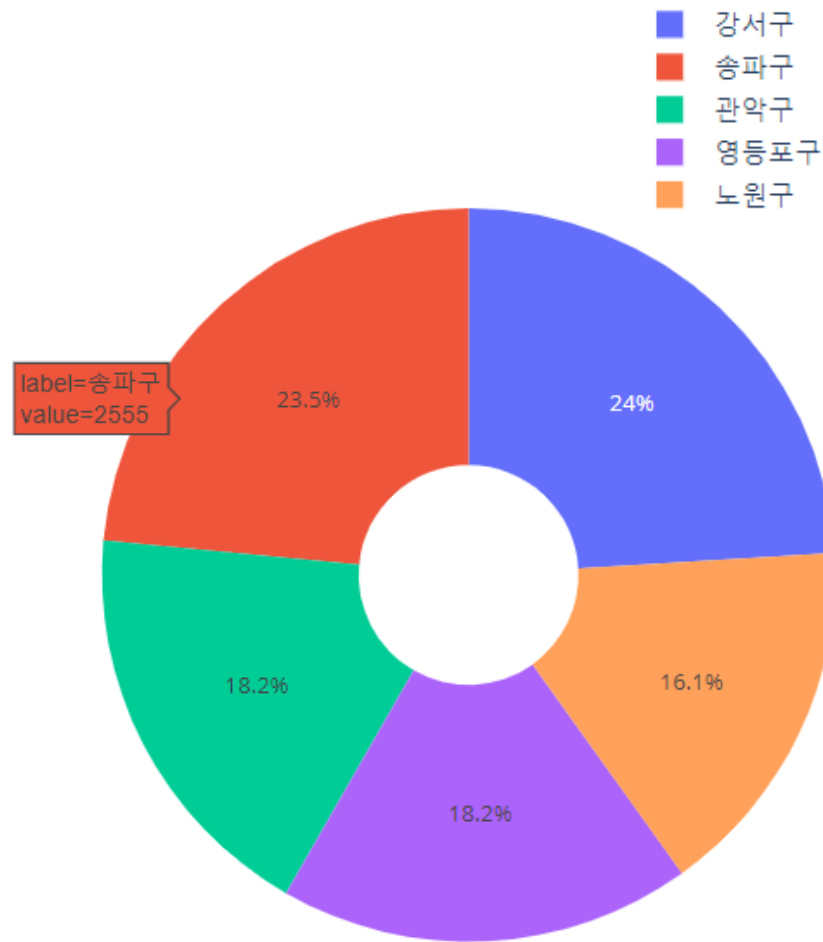


차트 여러 개 그리기

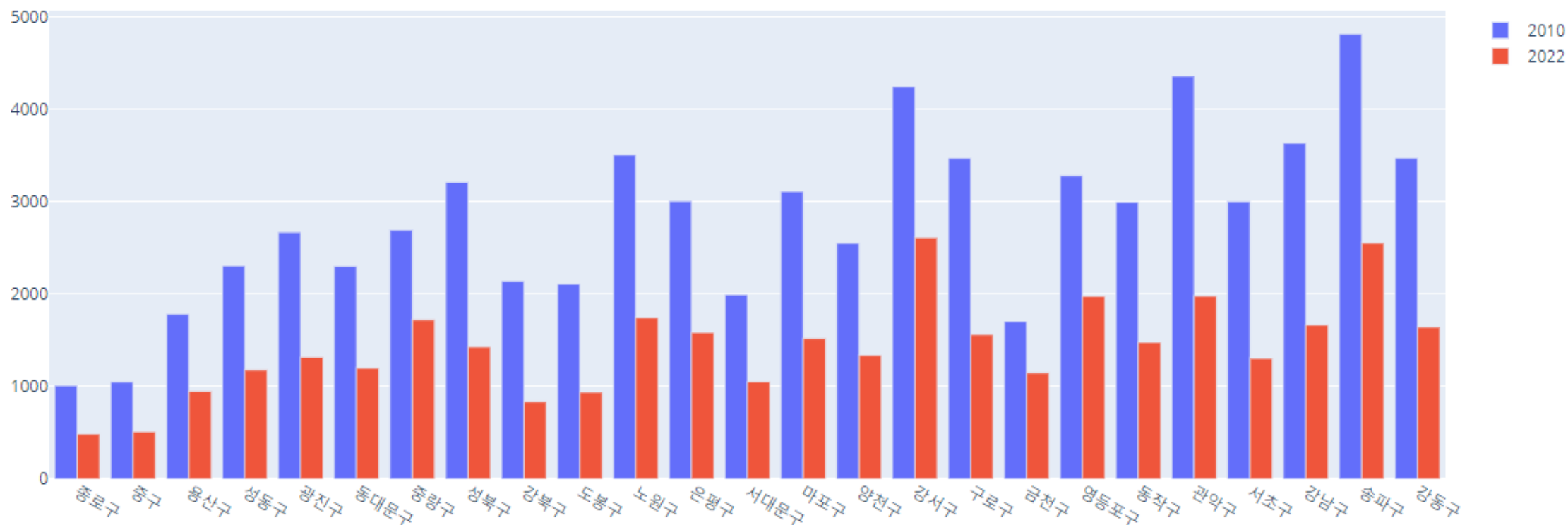
- graph_objects 모듈을 사용하여 여러 개의 차트 생성
- 생성 방법

```
fig = go.Figure()  
fig.add_trace(graph_objects 의 차트 종류)  
....  
fig.add_trace(graph_objects 의 차트 종류)  
fig.show()
```

- graph_objects 의 차트 종류
- ✓ go.Scatter() : scatter 차트
- ✓ go.Bar() : bar 차트
- ✓ go.Scatter(mode='lines') : line 차트

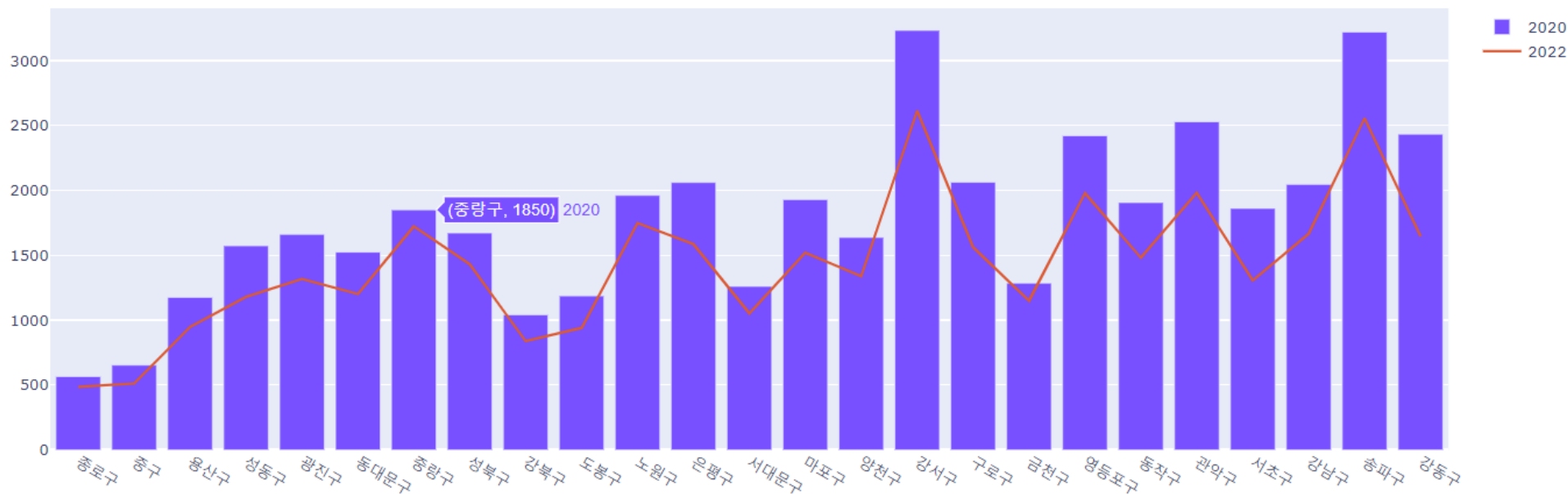
bar차트 여러 개 그리기

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(x=혼인건수['자치구별'], y=혼인건수['2010'], name='2010'))
fig.add_trace(go.Bar(x=혼인건수['자치구별'], y=혼인건수['2022'], name='2022'))
fig.show()
```



bar차트 여러 개 그리기

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(x=혼인건수['자치구별'], y=혼인건수['2020'], name='2020'))
fig.add_trace(go.Scatter(x=혼인건수['자치구별'], y=혼인건수['2022'], mode='lines', name='2022'))
fig.show()
```




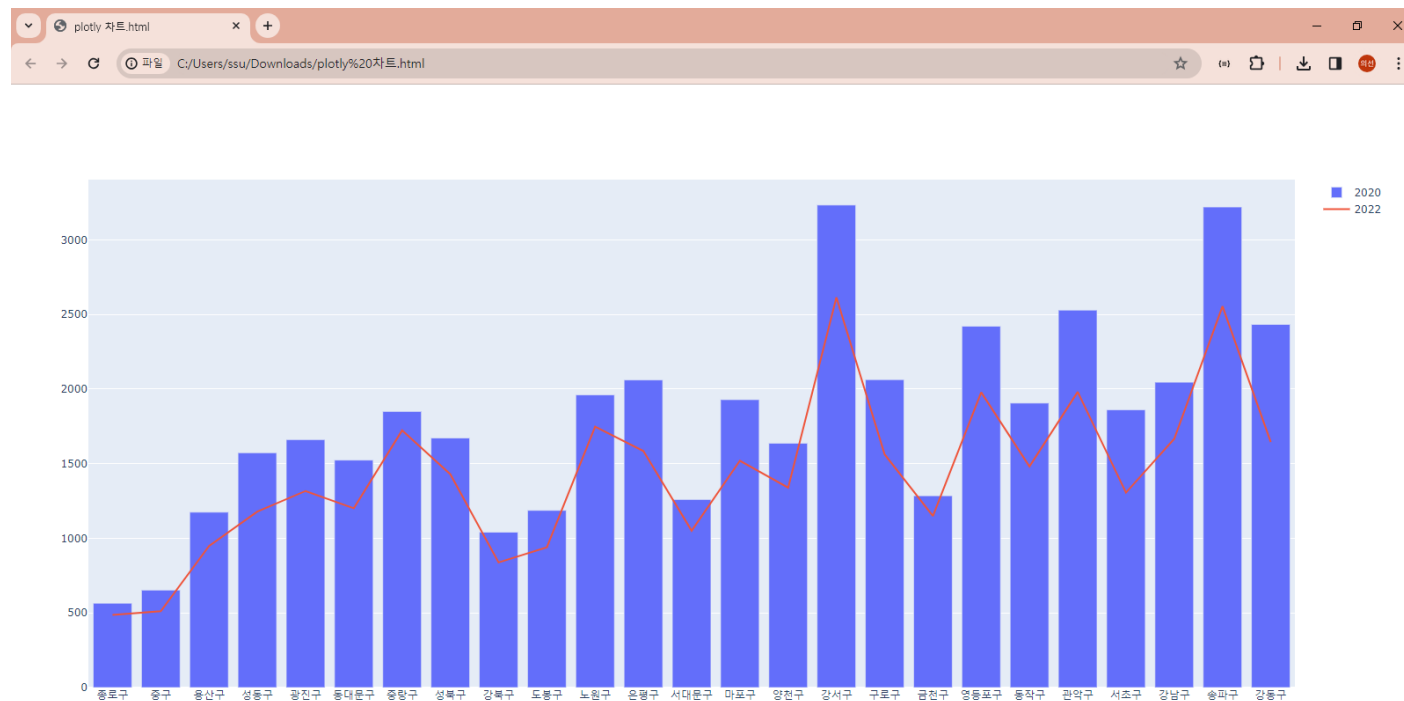
HTML 로 저장하기

- `fig.write_html('html파일명')` : 차트를 HTML로 저장

✓ 'html파일명' : *.html

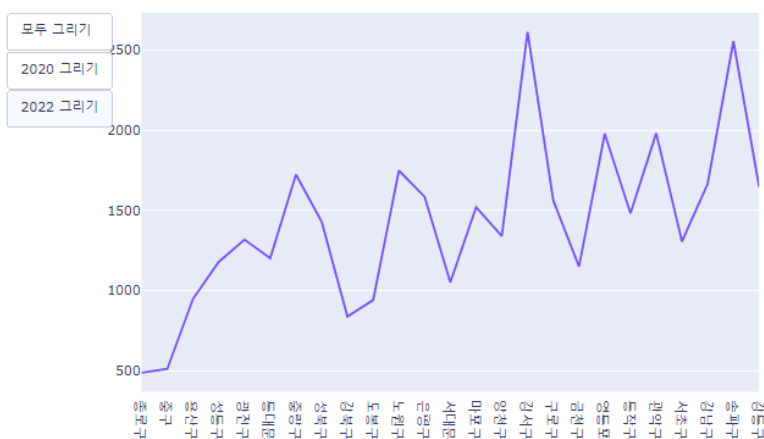
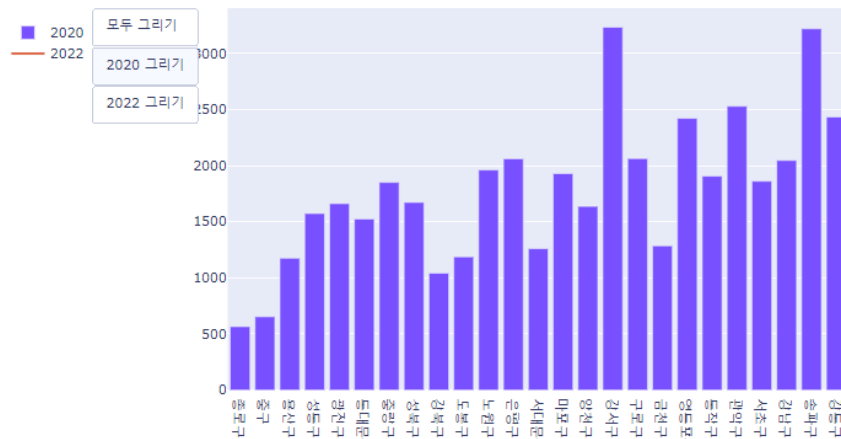
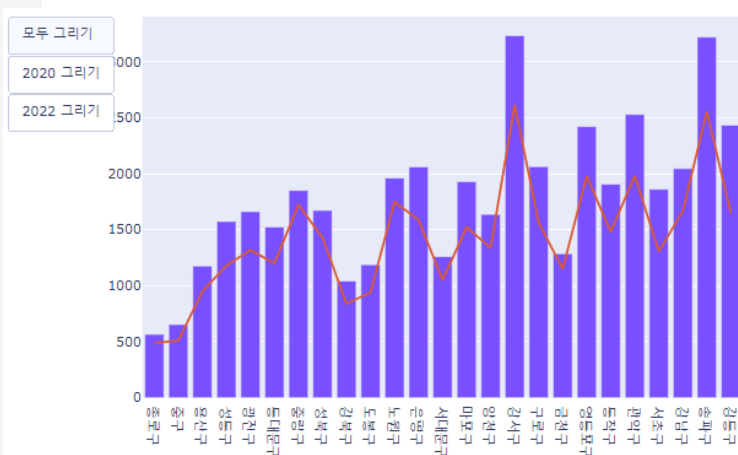
```
fig.write_html('plotly 차트.html')
```

 plotly 차트.html

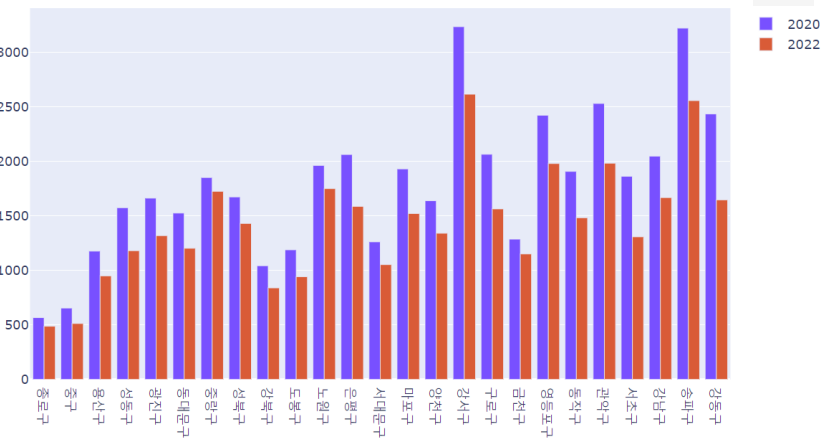
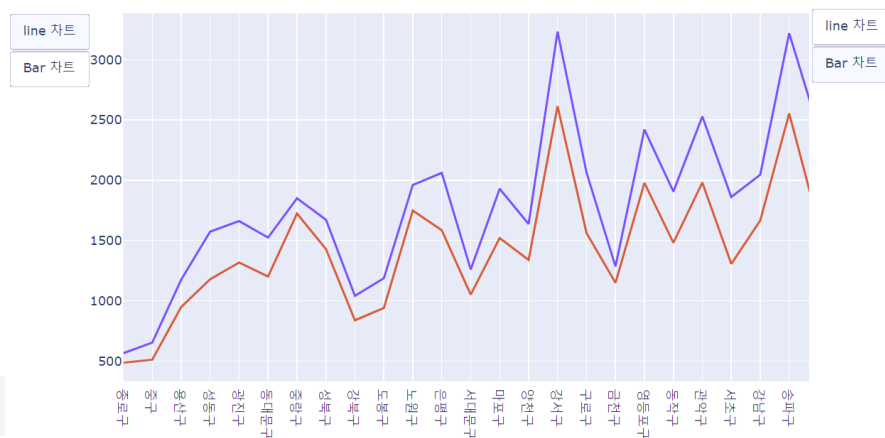


버튼을 활용하여 인터랙티브 차트 생성하기

• 차트 데이터 선택하기



• 차트 종류 변경하기



버튼을 활용하여 인터랙티브 차트 생성하기

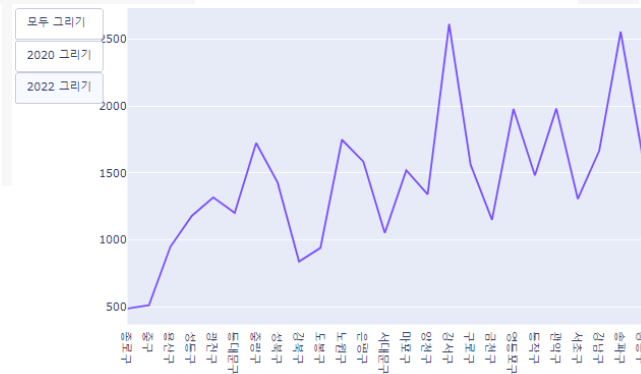
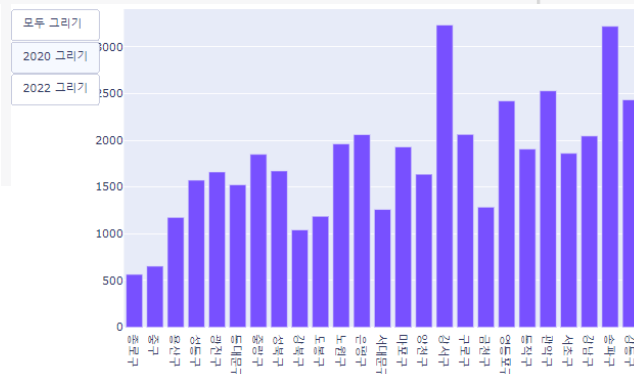
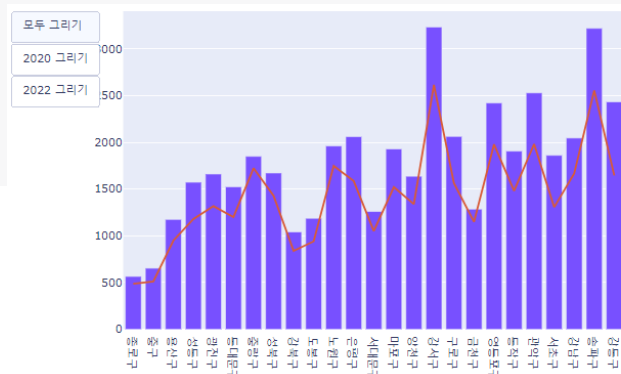
- 생성방법

```
fig.update_layout(  
    updatemenus=[  
        dict(  
            type="buttons",  
            buttons = list([  
                dict(label="문자열", method="수행할기능", args=[수행동작들]),  
                dict(label="문자열", method="수행할기능", args=[수행동작들]),  
                dict(label="문자열", method="수행할기능", args=[수행동작들])  
            ])  
        )  
    ]  
)
```

- ✓ **label="문자열"**: 버튼에 보여질 텍스트
- ✓ **method="수행할기능"**: 버튼을 클릭했을때 진행할 기능(restyle, relayout, update, animate)
- ✓ **args=[...]**: method에 따라 구체적으로 수행할 동작들

버튼을 활용하여 차트 선택하기

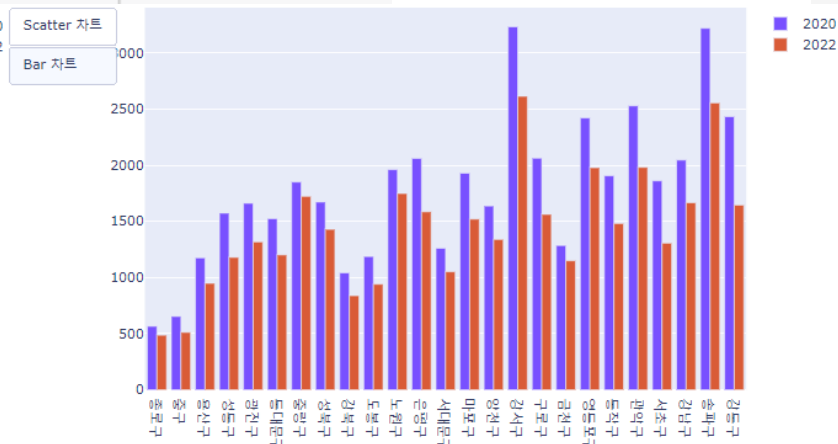
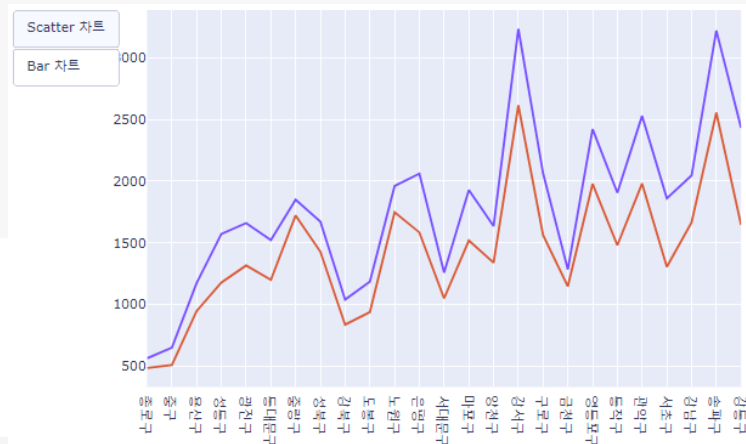
```
fig = go.Figure()
fig.add_trace(go.Bar(x=혼인건수['자치구별'], y=혼인건수['2020'], name='2020'))
fig.add_trace(go.Scatter(x=혼인건수['자치구별'], y=혼인건수['2022'], mode='lines', name='2022'))
fig.update_layout(
    updatemenus=[
        dict(
            type="buttons",
            buttons = list([
                dict(label="모두 그리기", method='update', args=[{'visible': [True, True]}]),
                dict(label="2020 그리기", method='update', args=[{'visible': [True, False]}]),
                dict(label="2022 그리기", method='update', args=[{'visible': [False, True]}])
            ])
        )
    ])
fig.show()
```



버튼을 활용하여 차트 종류 변경하기

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=혼인건수['자치구별'], y=혼인건수['2020'], mode='lines', name=2020))
fig.add_trace(go.Scatter(x=혼인건수['자치구별'], y=혼인건수['2022'], mode='lines', name=2022))

fig.update_layout(
    updatemenus=[
        dict(
            type = "buttons",
            buttons = list([
                dict(label="Scatter 차트", method="restyle", args=["type", "scatter"]),
                dict(label="Bar 차트", method="restyle", args=["type", "bar"])
            ])
        )
    ]
)
fig.show()
```



수고하셨습니다.