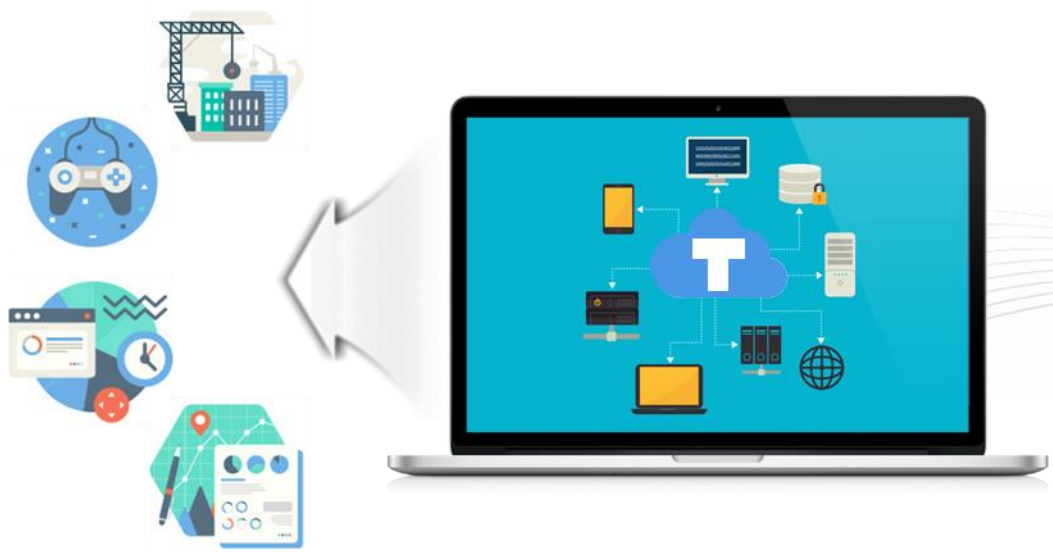
AIoT 데이터 시각화 및 대시보드 개발



Chapter 03.
Matplotlib, Seaborn, Plotly를 이용한 파이썬 데이터 시각화

Ⅰ Matplotlib의 구성 요소 및 특징

Ⅱ Scatterplot

Ⅲ Regplot

Ⅳ Lineplot

Chapter 03.
Matplotlib, Seaborn, Plotly를 이용한 파이썬 데이터 시각화

Ⅴ Boxplot, Stripplot, Swarmplot

Ⅵ Histplot

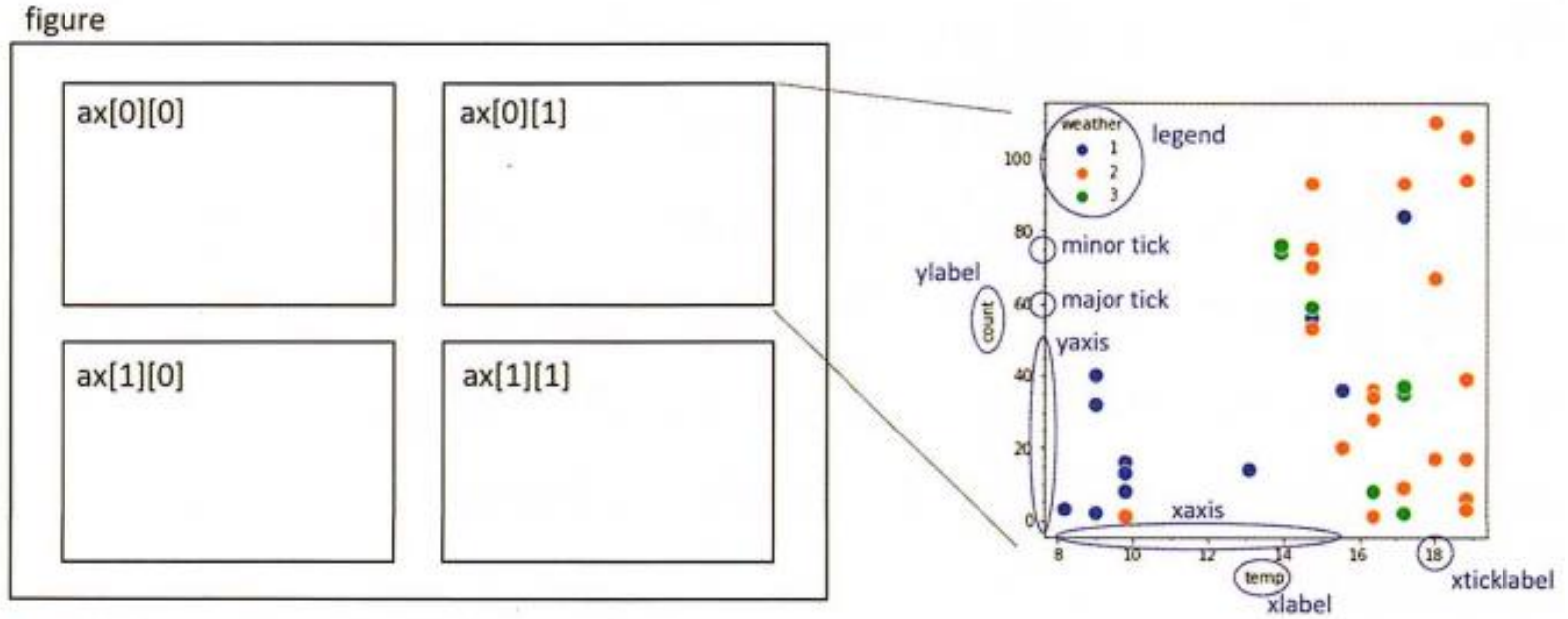
Ⅶ Heatmap

Ⅷ Matplotlib의 axes-level plot과 figure-level plot

Chapter 03.
Matplotlib, Seaborn, Plotly를 이용한 파이썬 데이터 시각화

IX 그래프 세부 요소 튜닝

- **Matplotlib는 하나의 도화지(figure)에 그린 여러 컷의 (axes) 만화에 비유**
- **하나의 figure는 여러 axes를 가질 수 있으며 개수와 각 ax의 크기는 임의로 지정할 수 있음**



▲ 그림 52 Matplotlib 그래프를 구성하는 기본 요소

■ Seaborn 라이브러리

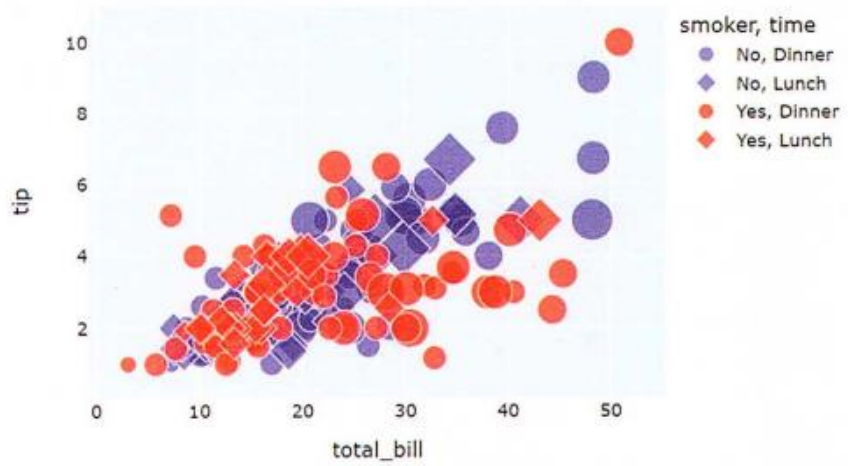
- Matplotlib를 기반으로 보다 깔끔하고 시인성이 뛰어난 그래프를 보다 편하게 그릴 수 있게 해주는 일종의 wrapper
- regression plot, box plot, heatmap 등의 통계적 그래프들을 다양하게 제공
- Seaborn으로 필요한 그래프를 그리고 글자 크기나 legend 위치 등 세부 수정 사항은 Matplotlib 라이브러리를 사용

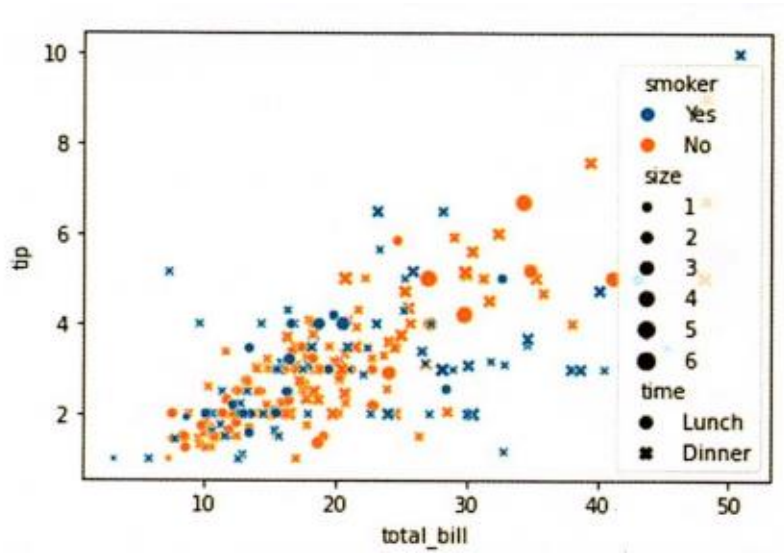
■ <Matplotlib & Seaborn>

- seaborn scatterplot에서는 그룹별 구분을 hue를 이용한 색깔을 통해 표현, scatter의 모양과 크기를 통해 다르게 나타낼 수 있음
- 한 번에 세가지의 변수들을 구분하여 하나의 scatter plot에서 표현하는 방법은 오히려 그래프의 복잡도를 증가시켜 선호되는 방법은 아님

■ <Plotly>

- plotly의 express 패키지를 통해 그리는 방법
- graph object를 이용하는 방법





■ <Matplotlib & Seaborn> Scatterplot : global_internet_users 데이터셋

- 1980년부터 2020년까지 여러 나라의 인터넷 사 용량에 대한 데이터
- "Year" 변수를 x축으로, 인터넷 사용자 수를 나타내는 "No. of Internet Users" 변수를 y축으로 하여 scatter plot 그리기

  → 그래프 를 그리기 위하여 matplotlib 라이브러리를 이용하여 도화지를 준비하고, 1컷 그래프
- 전달 해야 할 인자

  → x: X축으로 지정할 변수

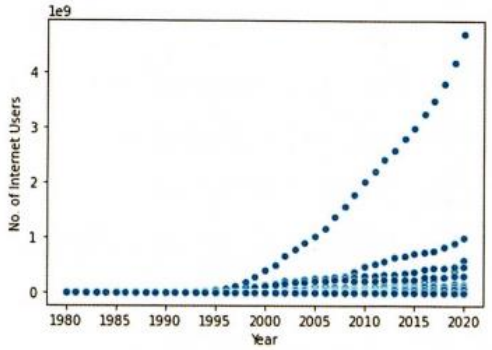
  → y: Y축으로 지정할 변수

  → data: 그래프를 그릴 데이터셋

  → ax: 그래프를 그릴 ax를 지정 (이번 예시에서는 앞서 matplotlib의 subplots을 통해 할당한 ax 변수)

Scatterplot (ch3-1.py)

```
fig, ax = plt.subplots()
sns.scatterplot(x='Year', y='No. of Internet Users', data=df, ax=ax)
```



▲ 그림 53 global_internet_users 데이터셋의 "Year"와 "No. of Internet Users" 변수를 seaborn의 scatterplot을 통해 나타낸 scatter plot

■ <Matplotlib & Seaborn> Scatterplot : global_internet_users 데이터셋

- color, size, marker size를 다르게 하여 나라/지역을 구분하여 그래프 그리기

  → 임의로 3개의 나라만 필터링하

  → loc 메서드를 이용하여 "Entity"가 “China", "India", "Finland"인 행만 필터링

- 필터링한 세 개 나라를 scatter plot에서 색깔별로 구분

  → hue 인자에 구분하는 기준이 되는 변수를 전달

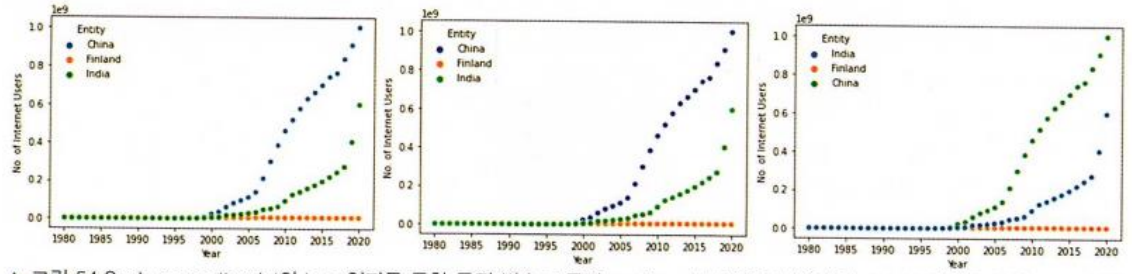
  → hue_order 인자를 전달 "India", "Finland", "China" 순으로 설정

Scatterplot (ch3-1.py)

```
fig, ax = plt.subplots()
sns.scatterplot(
    x='Year', y='No. of Internet Users', data=df, ax=ax, hue='Entity', palette='bright'
)
```

Scatterplot (ch3-1.py)

```
fig, ax = plt.subplots()
sns.scatterplot(
    x='Year', y='No. of Internet Users', data=df, ax=ax,
    hue='Entity', hue_order=['India', 'Finland', 'China']
)
```



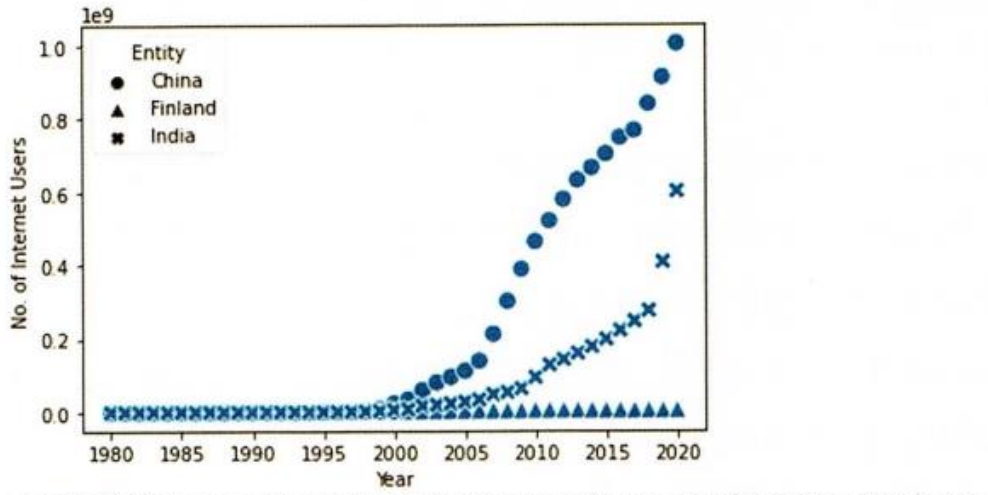
▲ 그림 54 Seaborn scatterplot의 hue 인자를 통한 특정 변수 그룹별 scatter plot 색깔 구분 (좌측). bright 인자를 통한 color 변화 (중앙). hue_order 인자를 통한 color 구분 그룹 순서 변경 (우측)

■ <Matplotlib & Seaborn> Scatterplot : global_internet_users 데이터셋

- hue 대신 style 인자에 "Entity"를 전달하고, markers 인자에 순서대로 “o", "", "x" 를 전달
  → "China", "Finland", "India"가 각각, ▲, X 모양으로 구분
- scatter의 size를 조절하는 인자
  → s 인자에 100을 전달

Scatterplot (ch3–1.py)

```
fig, ax = plt.subplots()
sns.scatterplot(
    x='Year', y='No. of Internet Users', data=df, ax=ax,
    style='Entity', markers=['o','^','X'], s=100
)
```



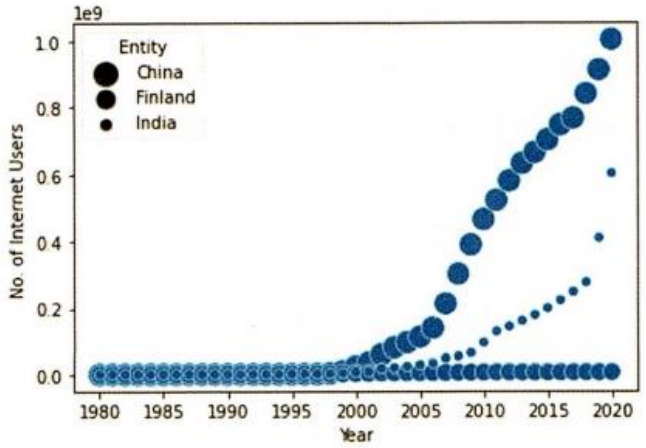
▲ 그림 56 Seaborn scatterplot의 style 인자를 통한 특정 변수 그룹별 scatter plot 모양 구분

■ <Matplotlib & Seaborn> Scatterplot : global_internet_users 데이터셋

- seaborn scatterplot에서 그룹을 구분할 수 있는 방법으로는 size 인자를 통하여 그룹별로 scatter의 크기 다르게 그리기
  → size 인자에 그룹별로 구분할 변수명을 전달 하고, sizes 인자에 튜플 형태로 scatter 크기의 최소와 최대 범위를 전달

Scatterplot (ch3-1.py)

```
fig, ax = plt.subplots()
sns.scatterplot(
    x='Year', y='No. of Internet Users', data=df, ax=ax,
    size='Entity', sizes=(40, 200)
)
```



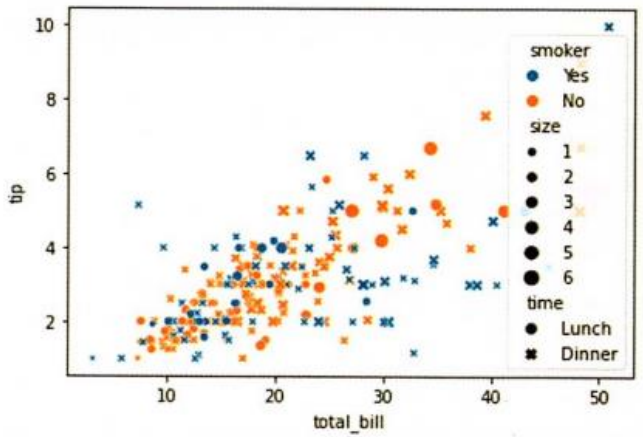
▲ 그림 57 Seaborn scatterplot의 size 인자를 통한 특정 변수 그룹별 scatter plot 크기 구분

■ <Matplotlib & Seaborn> Scatterplot : tips 데이터셋

- 특정 레스토랑에서 지불한 총 가격과 그 때 지불된 팁을 float형 변수로 가 지고 있으며, 성별, 흡연자 여부, 요일, 시간대, 함께 식사를 한 사람 수를 변수로 가지고 있음
- x축을 “total_bill" y축을 “tip"으로 설정하고 흡연자 여부를 색깔로, 시간대를 scatter의 모양으로, 함께 식사한 인 원을 size 변수로 구분해서 그래프 그리기

Scatterplot (ch3-1.py)

```
df = sns.load_dataset('tips')
fig, ax = plt.subplots()
sns.scatterplot(
    x='total_bill', y='tip', data=df, ax=ax,
    hue='smoker', style='time', size='size'
)
```



▲ 그림 58 Seaborn scatterplot의 hue, style, size 인자를 동시에 사용하여 하나 이상의 변수들의 그룹별 구분

■ <Plotly> Scatterplot : internet user 데이터셋도

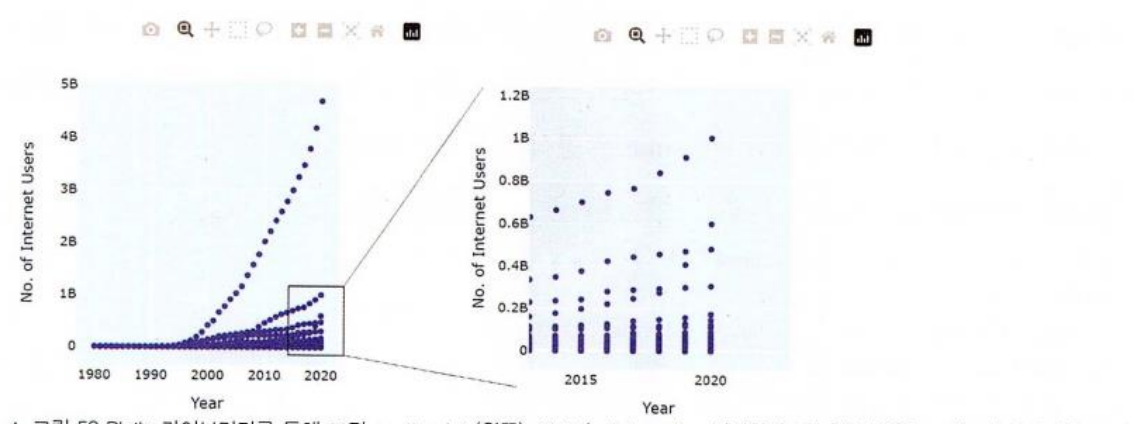
- Plotly를 통해 scatterplot을 그리기

  → Plotly를 통해 그래프를 그 릴 때에는 fig 변수만 할당

- x와 y 인자는 앞서 소개한 Seaborn의 scatterplot과 동일하며, data 인자 대신 data frame 인자를 사용
- width와 height 인자의 단위는 픽셀

Scatterplot (ch3-1.py)

```
fig = px.scatter(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400
)
fig.show()
```



▲ 그림 59 Plotly 라이브러리를 통해 그린 scatterplot (왼쪽). Plotly는 interactive 시각화가 가능하여 왼쪽 그래프의 우하단을 드래그하여 선택하면 선택한 영역을 확대할 수 있다 (오른쪽)

■ <Plotly> Scatterplot : internet user 데이터셋도

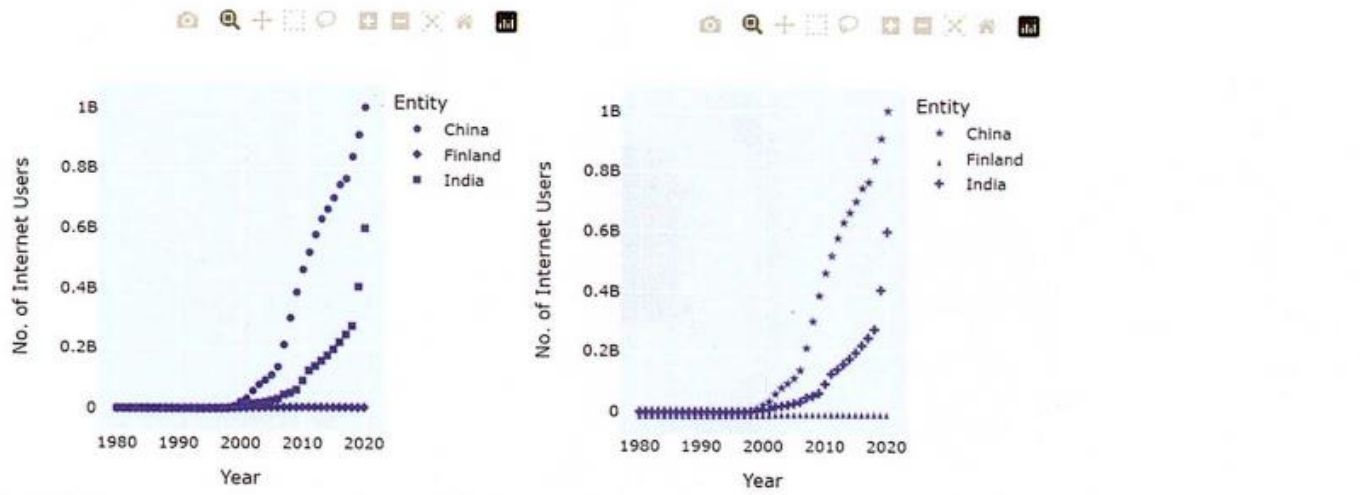
- Plotly에도 symbol 인자를 이용하여 동일한 방식으로 marker의 모양으로 변수를 구분
  → "Chian", "India", "Finland" 나라를 marker의 모양으로 구분
- symbol 인자를 통해서 구분되는 marker의 모양을 변경하려면 아래처럼 symbol_sequence에 원하는 symbol의 이름을 문자열 형태로 묶 어서 리스트로 전달

Scatterplot (ch3-1.py)

```
fig = px.scatter(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400, symbol='Entity'
)
fig.show()
```

Scatterplot (ch3-1.py)

```
fig = px.scatter(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400, symbol='Entity',
    symbol_sequence=['star','arrow','cross']
)
fig.show()
```



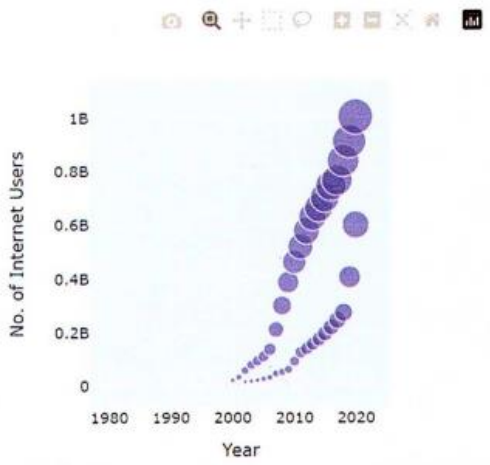
▲ 그림 61 Plotly express scatter 그래프에서 데이터의 변수 별 marker의 모양으로 구분 (왼쪽). symbol_sequence 인자를 통해 marker의 모양을 커스터마이징 (오른쪽)

■ <Plotly> Scatterplot : internet user 데이터셋도

- plotly express도 이와 비슷한 size 인자가 있 지만, 본 예시 데이터처럼 카테고리형 변수에 대해서는 작동하지 않음
  → y 변수인 “No. f Internet Users”는 수치형 데이터이기 때문에 size 인자에 전달할 수 있는데,
  → 해당 변수 크기에 따라서 scatter의 크기가 유동적으로 표현 되는 것을 확인

Scatterplot (ch3-1.py)

```
fig = px.scatter(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400, size='No. of Internet Users'
)
fig.show()
```



▲ 그림 62 Plotly express scatter 그래프에서 size 인자로 수치형 변수인 “No. of Internet Users”를 전달하여 해당 변수의 크기별로 scatter의 크기를 유동적으로 변화하여 표현한 그래프

■ <Plotly> Scatterplot : tips 데이터셋

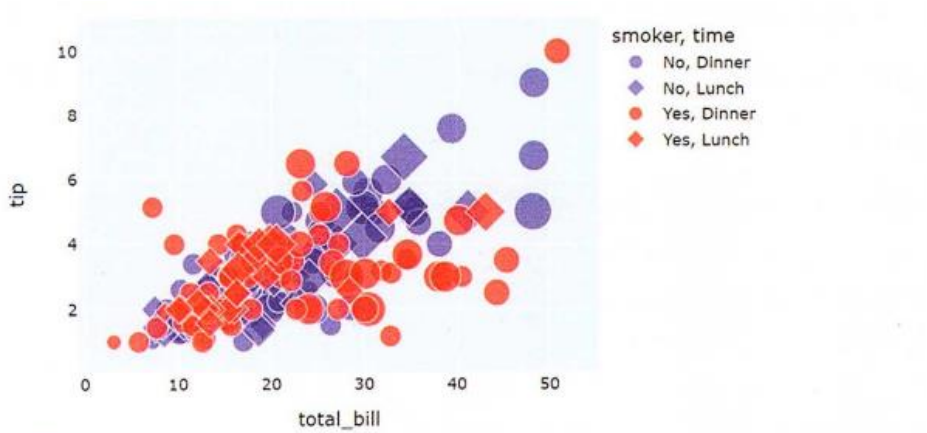
- X축을 “total_bill", y축을 “tip" 변수로 하고 “smoker 변수를 색깔로 구분, “size" 변수를 scatter의 크기로 구분하고 “time” 변수를 scatter의 모양을 통해서 구분하여 그리기

  → y 변수인 “No. f Internet Users”는 수치형 데이터이기 때문에 size 인자에 전달할 수 있는데,

  → 해당 변수 크기에 따라서 scatter의 크기가 유동적으로 표현 되는 것을 확인

Scatterplot (ch3-1.py)

```
df = sns.load_dataset('tips')

fig = px.scatter(
    data_frame=df, x='total_bill', y='tip',
    color='smoker', size='size', symbol='time',
    width=600, height=400,
)
fig.show()
```



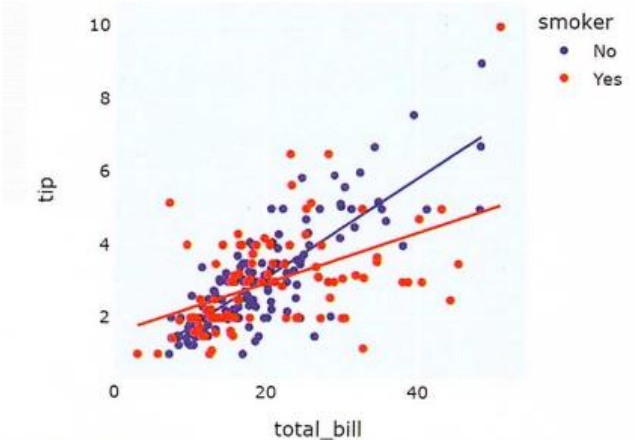
▲ 그림 63 “tips” 데이터셋을 plotly express의 scatterplot으로 나타낸 그래프 (상단 모드바 생략)

■ <Matplotlib & Seaborn>

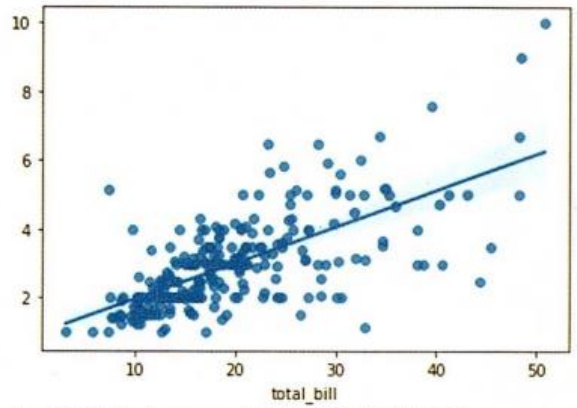
- Seaborn regplot은 regression plot의 약자로, scatter plot 기반 그래프에 회귀선을 추가 가능
- 회귀선은 직선, 포물선, logistic 등 여러 가지 형태가 가능함

■ <Plotly>

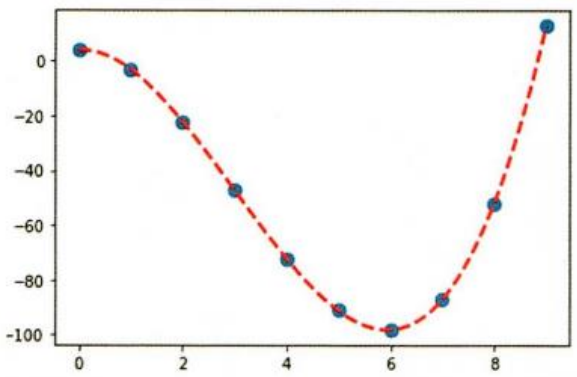
- Seaborn의 regplot을 통해 표현했던 회귀 그래프는 Plotly express scatterplot의 “trendline" 인자를 통해 동일하게 구현함



▲ 그림 68 Plotly express scatterplot의 "color" 인자를 이용하여 구분한 변수별로 회귀선을 추가한 그래프



▲ 그림 64 Seaborn regplot을 이용한 회귀선 표현



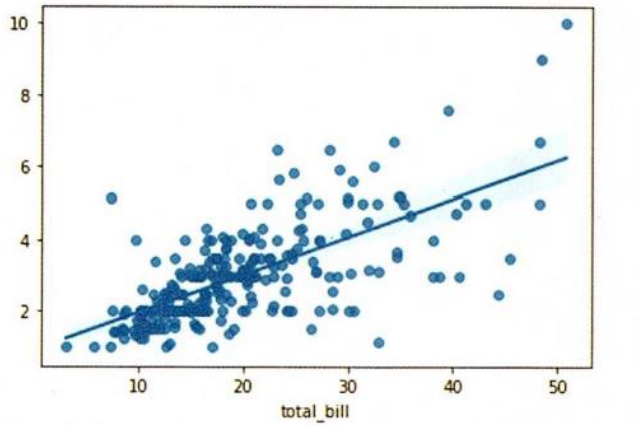
▲ 그림 66 Seaborn regplot의 scatter_kws 및 line_kws 인자를 통한 scatter와 line의 속성 변경

■ <Matplotlib & Seaborn> regplot : tips 데이터셋

- x축을 “total_bill"로, y축을 “tip"으로 하여 regplot을 그려보기
- 회귀선 주변에 보다 옅은 색으로 칠해진 영역을 확인할 수 있는데, 이는 회귀 결과의 95% 신뢰 구간을 나타냄
- 신뢰구간에 해당하는 영역을 나타내고 싶지 않다면 아래와 같이 ci (confidence interval) 인 자에 None을 전달

Regplot (ch3-2.py)

```
df = sns.load_dataset('tips')
fig, ax = plt.subplots()
sns.regplot(x='total_bill', y='tip', data=df, ax=ax)
fig.show()
```



▲ 그림 64 Seaborn regplot을 이용한 회귀선 표현

Regplot (ch3-2.py)

```
fig, ax = plt.subplots()
sns.regplot(x='total_bill', y='tip', data=df, ax=ax, ci=None)
```

■ <Matplotlib & Seaborn> regplot : tips 데이터셋

- 임의 3차함수에 대응하는 데이터 (x, y)를 만들어서 regplot을 이용하여 fitting

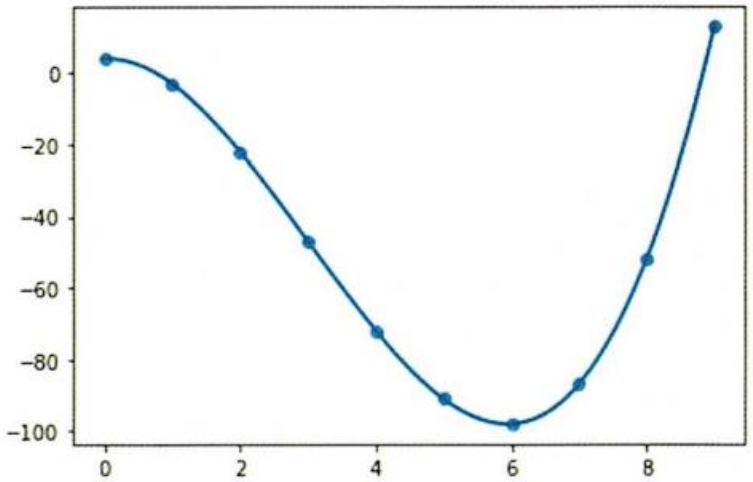
  → regplot의 order 인자에 3을 전달함으로써 거의 완벽한 회귀 선을 도출하는 것을 확인

  → 95%의 신뢰구간을 나타내는 영역이 거의 보이지 않는 것 을 확인할 수 있음

  → 이는 나타난 회귀선이 데이터를 매우 잘 나타낸다는 의미

Regplot (ch3-2.py)

```
import numpy as np

x = np.arange(0, 10, 1)
y = x**3 - 9*x**2 + x + 4

fig, ax = plt.subplots()
sns.regplot(x=x, y=y, ax=ax, order=3)
```



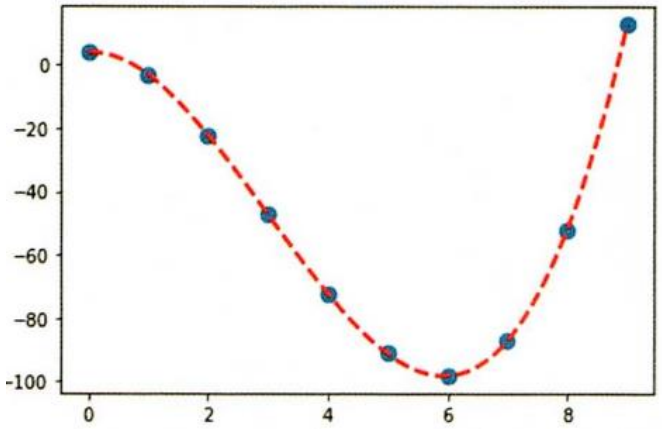
▲ 그림 65 임의의 3차 다항식에 의해 그려진 함수를 seaborn regplot으로 order 인자에 3을 전달하여 거의 완벽한 회귀선을 도출한 그래프

■ <Matplotlib & Seaborn> regplot : tips 데이터셋

- scatter_kws 인자에, 회귀선 관련은 line_kws 인자에 딕셔너리 형태로 값을 전달하여 속성을 변경
  → scatter의 크기를 80으로 설정하고, 회귀선의 색깔을 빨간색 점선으로 변경
- scatter의 크기를 조절하기 위하여 s 인자를 이용
- 회귀선을 점선 형태로 변경하기 위해 linestyle 인자에 '---'를 전달
  → 점선 표현을 위한 문자열은 regplot 외에 matplotlib 및 seaborn 라이브러리에서 통용
- scatter 인자에 False를 전달하게 되면 regplot에서 scatter들을 보이지 않게 설정할 수도 있음
- hue 인자 사용이 불가능

Regplot (ch3-2.py)

```
fig, ax = plt.subplots()
sns.regplot(
    x=x, y=y, ax=ax, order=3,
    scatter_kws={'s':80}, line_kws={'color':'red', 'linestyle':'--'}
)
```



▲ 그림 66 Seaborn regplot의 scatter_kws 및 line_kws 인자를 통한 scatter와 line의 속성 변경

■ <Plotly> regplot : tips 데이터셋

- Plotly express scatterplot의 “trendline" 인자를 통해 동일하게 구현

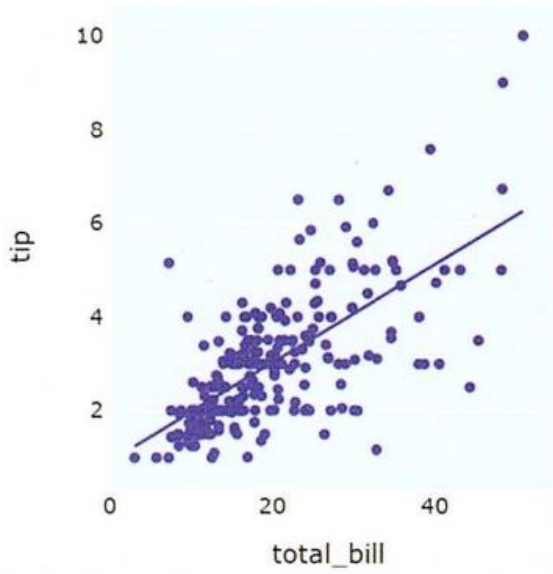
  → 해당 인자에 “ols"를 전달

  → “ols"는 회귀식을 구하는 대표적인 방법들 중 하나인 최소자승법을 뜻 함

  →“ols” 외에도 "lowess", "rolling", "expanding", “ewm" 등 다양한 값들 을 전달할 수 있음

Regplot (ch3-2.py)

```
df = sns.load_dataset('tips')

fig = px.scatter(data_frame=df, x='total_bill', y='tip', width=400, height=400, trendline='ols')
fig.show()
```



▲ 그림 67 Plotly express scatterplot의 “trendline” 인자를 통해 추가한 회귀선

■ <Plotly> regplot : tips 데이터셋

- tips 데이터셋에서 “smoker" 변수별로 회귀선을 따로 구한 경우

  → 단순하게 “color" 인자에 해당 변수명인 “smoker"를 전달

  → plotly 라이브러리의 장점 중 하나임

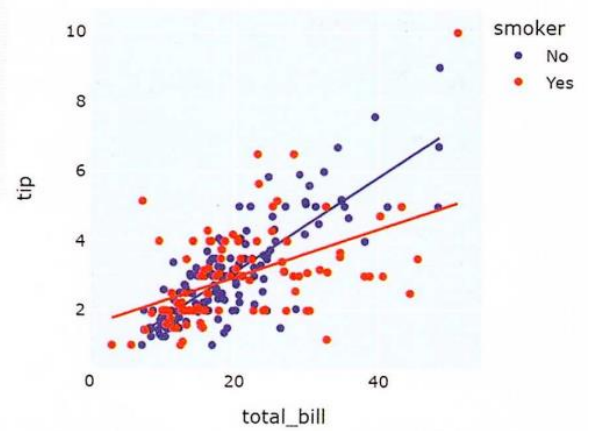
- Scatter들은 “smoker " 변수의 값에 따라 색깔 구분을 하고싶지만, 회귀선은 전체 데이터를 대상으로 그리고 싶다면?

  → trendline scope 인자에 “overall"을 전달

  → 회귀선을 구하는 데 전체 데이터를 사용

Regplot (ch3-2.py)

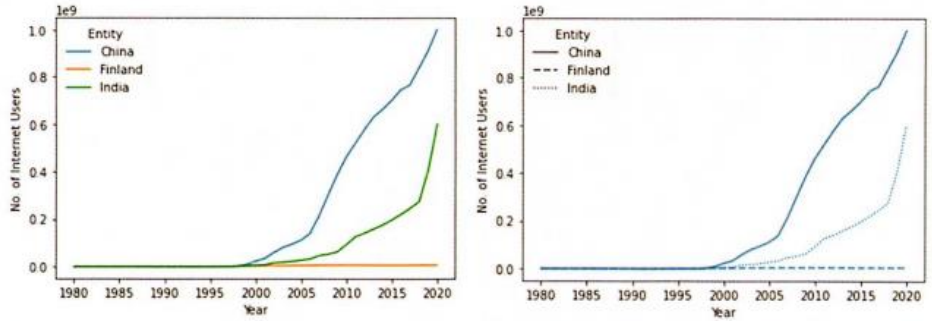
```
fig = px.scatter(
    data_frame=df, x='total_bill', y='tip',
    width=450, height=400,
    color='smoker', trendline='ols'
)
fig.show()
```



▲ 그림 68 Plotly express scatterplot의 "color" 인자를 이용하여 구분한 변수별로 회귀선을 추가한 그래프

■ <Matplotlib & Seaborn>

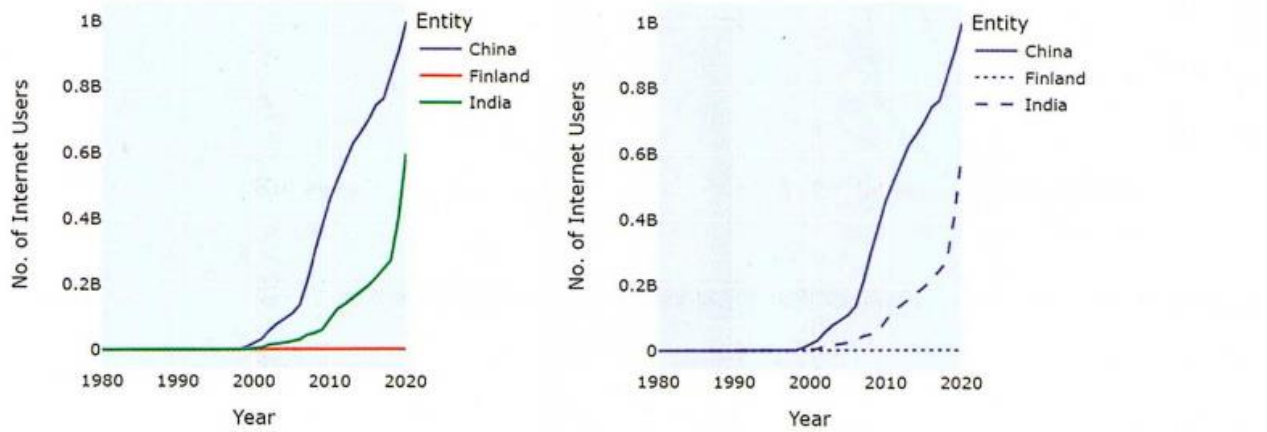
- Seaborn 라이브러리가 제공하는 그래프 중 연속형 변수 간 상관성을 확인할 수 있음
- scatterplot의 scatter 대신 line의 형태로 그래프가 그려짐



▲ 그림 69 Seaborn lineplot을 통한 global_internet_users 데이터셋의 시각화. 연도별 인터넷 사용량을 나라 별로 색깔과 (왼쪽) line의 형태 (오른쪽)로 구분하였습니다.

■ <Plotly>

- Plotly express로도 손쉽게 lineplot을 그릴 수 있음



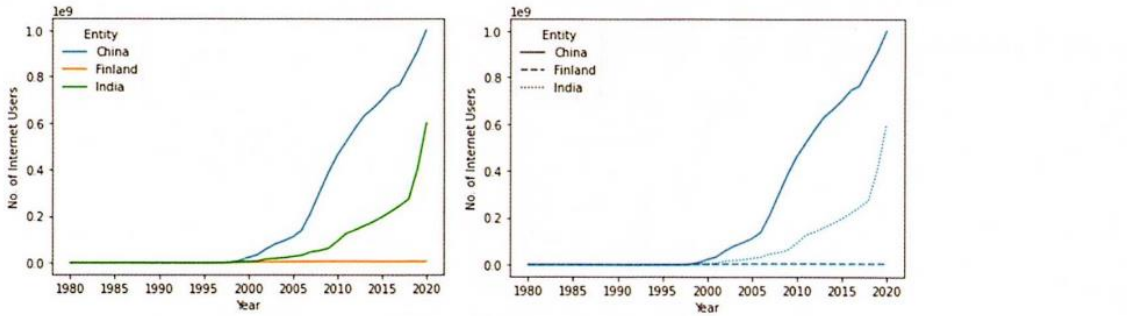
▲ 그림 71 Plotly express lineplot으로 표현한 global_internet_user 데이터셋. "color"인자를 통한 변수별 선 색깔 구분 (왼쪽), "line_dash" 인자를 통한 선 모양 구분 (오른쪽)

■ <Matplotlib & Seaborn> Lineplot : global_internet_users 데이터셋

- Seaborn 라이브러리가 제공하는 그래프 중 연속형 변수 간 상관성을 확인할 수 있는 것
- 각 점들을 이어서 x축의 변수가 변함에 따라 데이터의 추 세를 확인하기 용이하다는 장점을 가지고 있음
- scatterplot과 동일한 방식으 로 특정 변수가 속해 있는 그룹별로 line의 색깔과 모양 (형태)에 따라 구분할 수 있음
  → 필요한 인자는 scatterplot과 동일하게 hue와 style
- 데이터셋에서 "China", "Finland", "India" 세 개의 나라만 남긴 후, 나라 별 로 line의 색깔과 형태로 구분

Lineplot (ch3-3.py)

```
df = pd.read_csv('./datasets/global_internet_users/global_internet_users.csv')
entities = ['China', 'India', 'Finland']
df = df.loc[df['Entity'].isin(entities)]

fig, ax = plt.subplots()
sns.lineplot(
    x='Year', y='No. of Internet Users',
    data=df, ax=ax, hue='Entity'
)
```



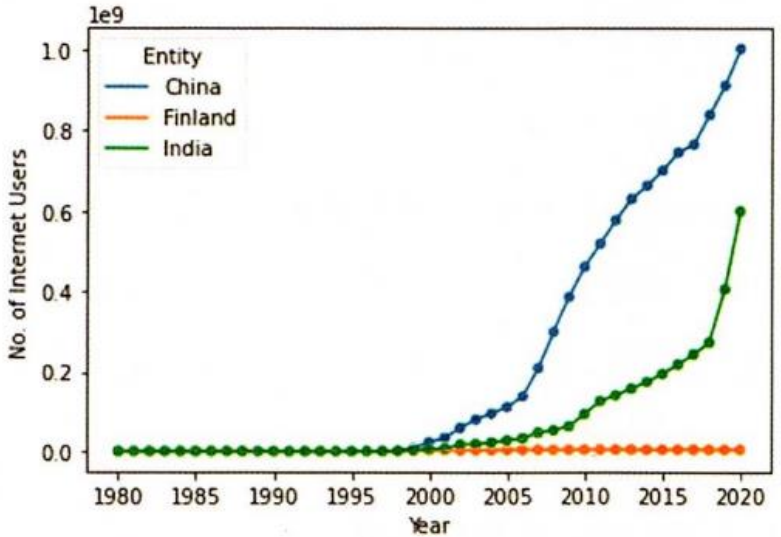
▲ 그림 69 Seaborn lineplot을 통한 global_internet_users 데이터셋의 시각화. 연도별 인터넷 사용량을 나라 별로 색깔과 (왼쪽) line의 형태 (오른쪽)로 구분하였습니다.

■ <Matplotlib & Seaborn> Lineplot : global_internet_users 데이터셋

- scatter plot과 line plot을 동시에 사용하여 line plot에서 데이터의 개수와 분포를 어느 정도 알 수 있음
  → lineplot과 scatterplot을 동일한 ax 위에 그려주면 됨
- scatterplot의 legend 인자에서 False가 전달
  → scatterplot과 관련된 legend를 표시하지 않겠다는 뜻
  → False로 전달되지 않으면 global_internet_users 데이터셋의 각 나라별 색깔 구분이 lineplot과 scatterplot에 의해 중복되어 표시되기 때문에 불필요하게 그래프가 복잡하게 됨

Lineplot (ch3-3.py)

```
fig, ax = plt.subplots()
sns.lineplot(
    x='Year', y='No. of Internet Users',
    data=df, ax=ax, hue='Entity'
)
sns.scatterplot(
    x='Year', y='No. of Internet Users',
    data=df, ax=ax, hue='Entity', legend=False
)
```



▲ 그림 70 lineplot과 scatterplot을 동일 ax 위에 그린 그래프

■ <Plotly> Lineplot : global_internet_users 데이터셋

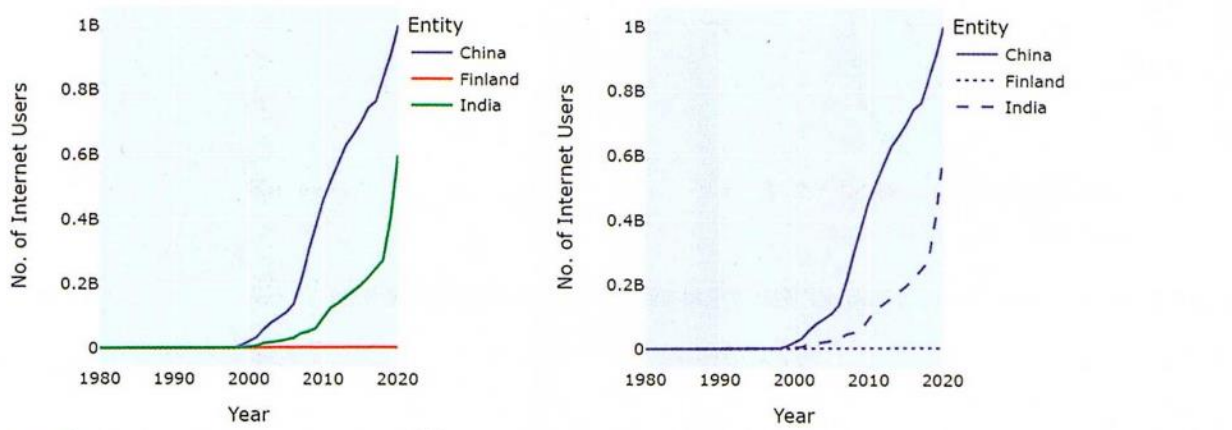
- line 함수를 사용
- "China", "Finland", "India" 세 나라의 연도별 인터넷 사용자 수를 그려보고,
- 선의 색깔을 “color" 인자를 통해서 구분해 보도록 함
- 특정 변수의 값 별로 선의 모양을 구분할 수 있는 데, "color"인자를 사용하는 것과 동일한 방식으로 “line_dash" 인자에 값 별로 구분할 변수명 을 전달하면 됨

Lineplot (ch3-3.py)

```
fig = px.line(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400, color='Entity'
)
fig.show()
```

Lineplot (ch3-3.py)

```
fig = px.line(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400, line_dash='Entity'
)
fig.show()
```



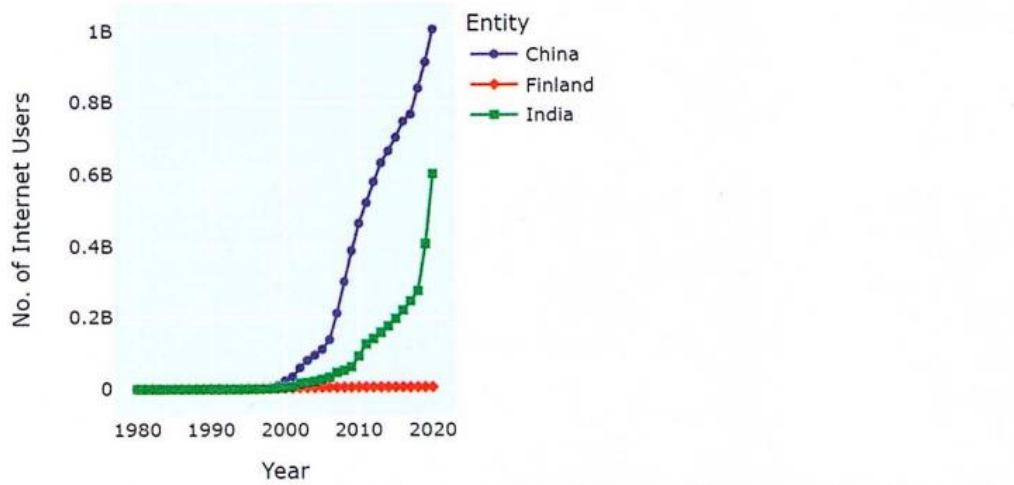
▲ 그림 71 Plotly express lineplot으로 표현한 global_internet_user 데이터셋. “color”인자를 통한 변수별 선 색깔 구분 (왼쪽), “line_dash” 인자를 통한 선 모양 구분 (오른쪽)

■ <Plotly> Lineplot : global_internet_users 데이터셋

- Plotly express lineplot의 “symbol" 인자를 사용하여 위 그림의 왼쪽 그래프에서 scatter를 추가한 그래프 그리기

Lineplot (ch3-3.py)

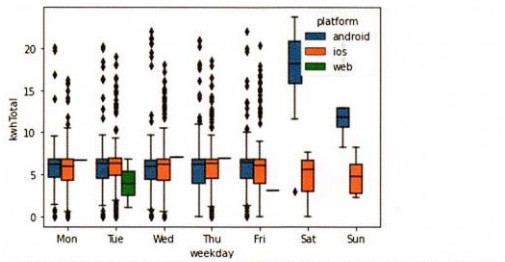
```
fig = px.line(
    data_frame=df, x='Year', y='No. of Internet Users',
    width=400, height=400,
    color='Entity', symbol='Entity'
)
fig.show()
```



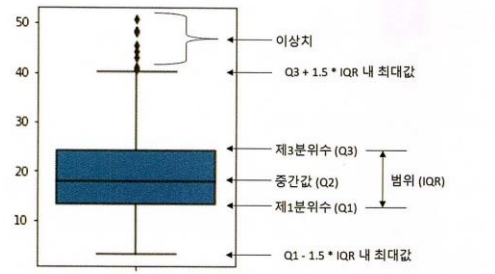
▲ 그림 72 Plotly express lineplot의 “symbol” 인자를 통해 lineplot과 scatterplot을 동시에 표현

■ **<Matplotlib & Seaborn>**

- Box plot은 범주형 하나 이상의 범주형 변수에 대한 데이터의 분포를 사분위수, 중간값, 이상치 등의 통계치를 박스 형태로 표현하는 그래프



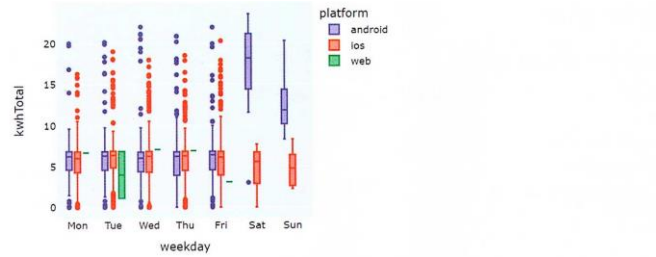
▲ 그림 78 Seaborn boxplot의 hue 인자를 이용한 특정 변수의 그룹별 box 구별 표현



▲ 그림 73 Seaborn boxplot을 이용해 그린 그래프와 boxplot의 각 구성요소의 의미

■ **<Plotly>**

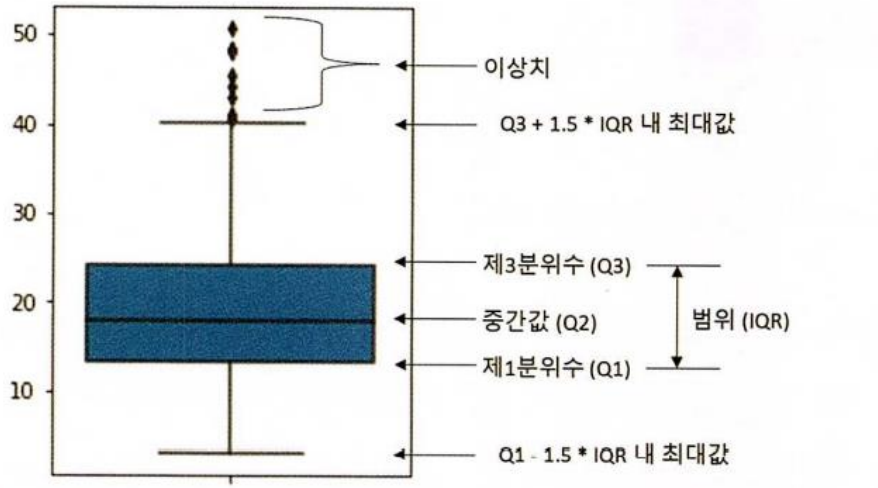
- boxplot 또한 Plotly 라이브러리를 사용하여 손쉽게 그릴 수 있음



▲ 그림 82 Plotly express boxplot에서 "color" 인자를 사용하여 특정 변수의 값 별로 box를 나누어 그린 그래프

■ <Matplotlib & Seaborn> Box plot

- 범주형 하나 이상의 범주형 변수에 대한 데이터의 분포를 사분위수, 중간값, 이상치 등의 통계치를 박스 형태로 표현하는 그래프
- 한 그래프에서 여러 의미 있는 통계 값들을 확인할 수 있어서 특정 범주형 변수의 각 값에 따른 데이터 분포를 비교하기에 용이함
- boxplot은 제1분위수와 제3분위수 사이의 범위를 박스 형태로 표현
- 박스 내에 제2분 위수(중간값)를 선으로 표시하고 있으며,
- 제3분위수와 제1분위수 사이의 범위를 interquartile range (IQR)이라고 칭함
- 박스 밖 위, 아래로는 수염 (whisker)가 있는데,
  위쪽 수염의 끝 은 Q3 + 1.5. IQR 보다 작거나 같은 데이터 중 최대값을 나타내며,
- 아래쪽 수염의 끝은 Q1- 1.5. IQR 보다 크거나 같은 데이터 중 최소값을 나타냄
- box plot에서는 이렇게 양쪽 수염 바깥에 위치 한 데이터들을 ***이상치***로 간주함



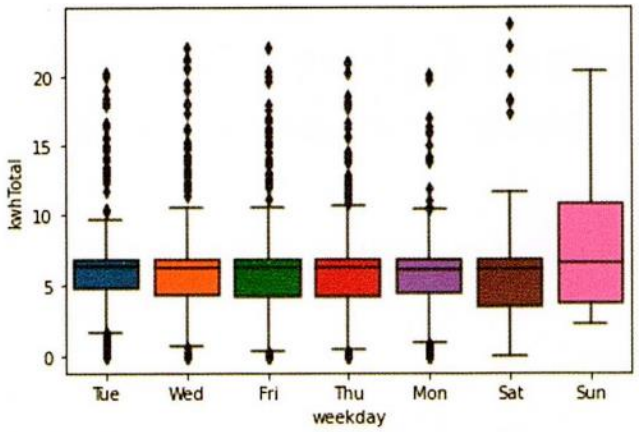
▲ 그림 73 Seaborn boxplot을 이용해 그린 그래프와 boxplot의 각 구성요소의 의미

■ <Matplotlib & Seaborn> Box plot : EV_charge 데이터셋

- EV charge 데이터셋은 각 전기자동차의 충전 이력을 기록한 로그 데이터
  → 충전이 시작되고 끝난 날짜와 시간, 충전량, 충전요금 등이 건 별로 기록되어 있음
- 이 데이터셋을 이용하여 요일별로 충전량을 box plot으로 그려서 요일별 전기차 충전량을 비교

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
df = pd.read_csv('./datasets/EV_charge/EV_charge.csv')

fig, ax = plt.subplots()
sns.boxplot(x='weekday', y='kwhTotal', data=df, ax=ax)
```



▲ 그림 74 Seaborn boxplot을 이용한 EV_charge 데이터셋의 요일 별 전기차 충전량

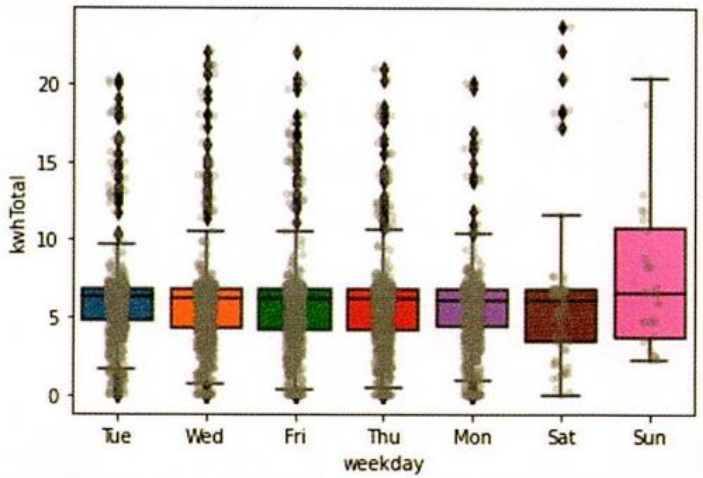
- Box plot은 데이터의 여러 통계치들을 한 눈에 알기 쉽다는 장점이 있지만,
- 통계치를 나타내는 것이기 때문에 원본 데이터의 분포를 알기에는 어렵다는 단점도 존재

■ <Matplotlib & Seaborn> Box plot : EV_charge 데이터셋

- Seaborn에서는 boxplot과 stripplot 혹은 swarmplot을 이용하여 원본 데이터를 box plot과 함께 표현할 수 있음
  → boxplot과 stripplot 혹은 swarmplot을 동일 ax에 그림
- stripplot을 그려줄 때 color를 grey로 설정하고, alpha 인자에 0.4를 전달
  → color 인자에 아무것도 전달하지 않으면, box와 동일한 색으로 자동설정됨
  → box와 색깔이 겹치기 때문에 stripplot에 의해 그려진 scatter를 확인하기 어렵기 때문에
  → alpha는 투명도를 설정하는 인자

Boxplot, Stripplot, Swarmplot (ch3–4.py)

```
fig, ax = plt.subplots()
sns.stripplot(x='weekday', y='kwhTotal', data=df, ax=ax, color='grey', alpha=0.4)
sns.boxplot(x='weekday', y='kwhTotal', data=df, ax=ax)
```



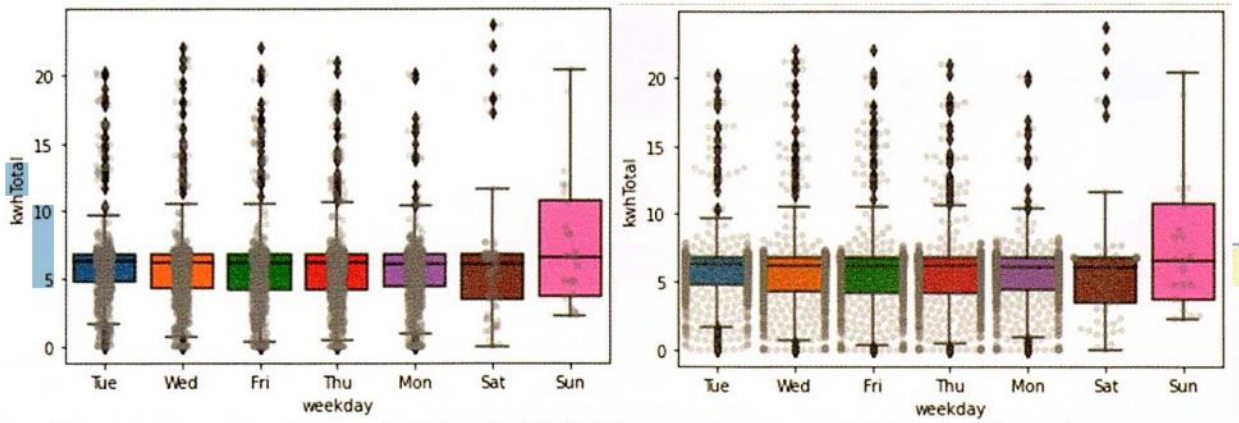
▲ 그림 75 Seaborn boxplot과 stripplot을 동시에 사용하여 boxplot을 이루는 원본 데이터를 scatter 형식으로 함께 표현

■ stripplot과 swarmplot의 차이점은 무엇일까?

- 이 둘은 모두 하나 이상의 범 주형 변수의 데이터 분포를 scatter 형식으로 표현하지만, 표현 방식에 약간의 차이가 있음
- swarmplot은

  → 각 scatter들의 겹침을 최소화하도록 그래프가 그려지기 때문에 각각의 scatter들을 확인하기에 보다 편리하다는 장점

  → 데이터의 크기가 커질수록 swarmplot은 stripplot에 비해 코드 실행 속도가 현저히 떨어지기 때문에, 각 상황에 따라 적절한 plot을 선택하여 사용
- boxplot과 stripplot을 함께 그린 경우(왼쪽)와 boxplot과 swarmplot을 함께 그린 경우(오른쪽)



▲ 그림 76 Seaborn boxplot과 stripplot을 함께 그린 경우(왼쪽)와 boxplot과 swarmplot을 함께 그린 경우(오른쪽)

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
fig, ax = plt.subplots()
sns.swarmplot(x='weekday', y='kwhTotal', data=df, ax=ax, color='grey', alpha=0.4)
sns.boxplot(x='weekday', y='kwhTotal', data=df, ax=ax)
```

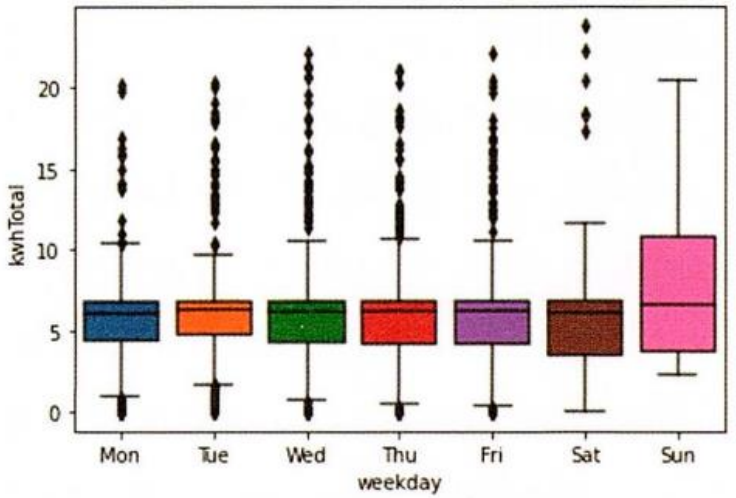
■ <Matplotlib & Seaborn> Box plot : EV_charge 데이터셋

- 그래프에서 x축을 월요일부터 일요일 순서대로 설정하여 그래프 그리기

  → x축의 순서를 의도대로 변경하기 위해서는 order 인자에 리스트 형식으로 x축의 순서를 전달하면 됨

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
weekday_order = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']

fig, ax = plt.subplots()
sns.boxplot(x='weekday', y='kwhTotal', data=df, ax=ax, order=weekday_order)
```



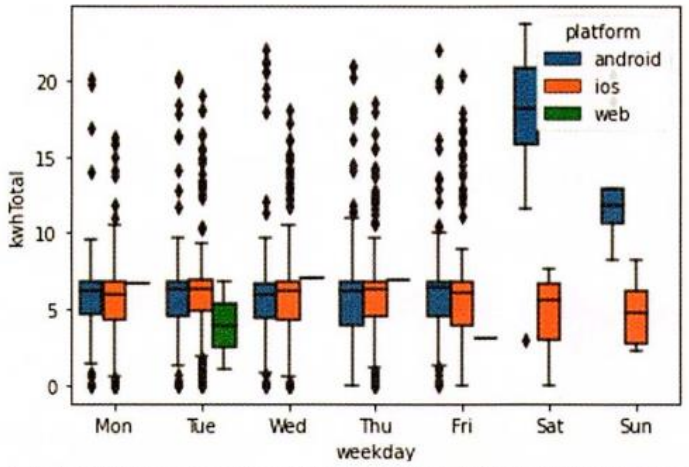
▲ 그림 77 Seaborn boxplot의 x축 순서 임의 지정

■ <Matplotlib & Seaborn> Box plot : EV_charge 데이터셋

- seaborn boxplot은 x축 각 변수 내에서 또 다른 변수의 특정 그룹별로 box를 나눠 서 그릴 수 있음
  → scatterplot에서와 동일하게 hue 인자를 통해 그룹별로 나눌 변수 명을 전달하면 됨
- EV_charge 데이터셋에서 “platform" 변수를 hue 인자에 전달하여 요일별 전기차 충전량을 platform 변수 그룹별로 나타냄
  → platform 변수는 각 충전 건에서 사용자가 충전을 위해 이용했던 기기를 “ios", "android", "web" 세 가지로 나눠서 나타냄

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
fig, ax = plt.subplots()
sns.boxplot(
    x='weekday', y='kwhTotal',
    data=df, ax=ax, order=weekday_order, hue='platform'
)
```



▲ 그림 78 Seaborn boxplot의 hue 인자를 이용한 특정 변수의 그룹별 box 구별 표현

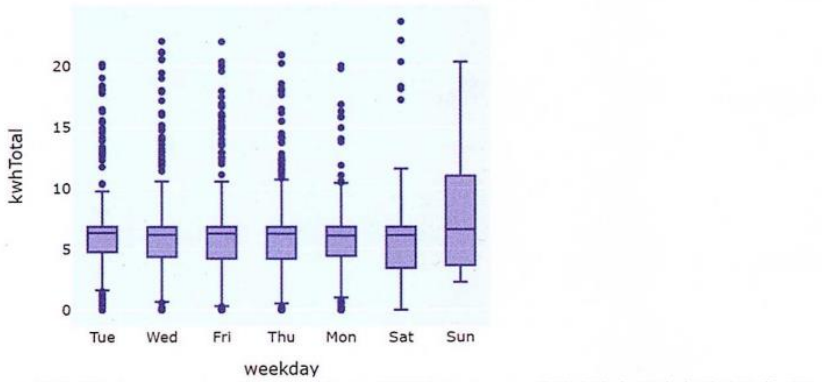
■ <Plotly> Box plot : EV_charge 데이터셋

- EV charge 데이터넷 을 이용하여 요일별 전기차 충전량을 비교

  → Plotly express의 box 함수를 사용

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
fig = px.box(
    data_frame=df, x='weekday', y='kwhTotal',
    width=500, height=400
)
fig.show()
```

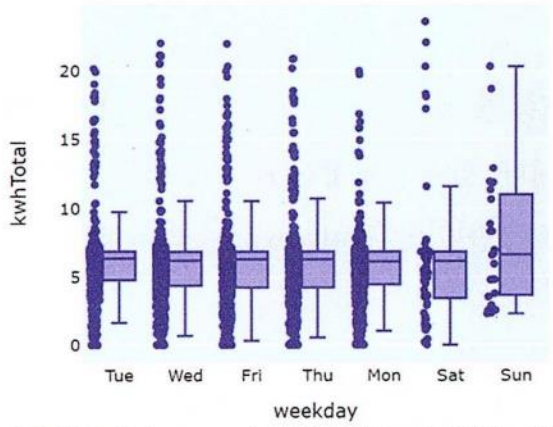


▲ 그림 79 Plotly express의 boxplot으로 표현한 EV charge 데이터셋의 요일별 전기차 충전량

- Plotly 라이브러리에서도 유사한 형태로 boxplot에서 각 데이터 포인트들을 나타냄

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
fig = px.box(
    data_frame=df, x='weekday', y='kwhTotal',
    width=500, height=400, points='all'
)
fig.show()
```



▲ 그림 80 Plotly express를 통해 그린 boxplot에서 box와 각 데이터 포인터들을 scatter로 함께 나타낸 그래프

■ <Plotly> Box plot : EV_charge 데이터셋

- x축을 보면 월요일부터 일요일까지 요일 순서가 뒤죽박죽인 것을 확인

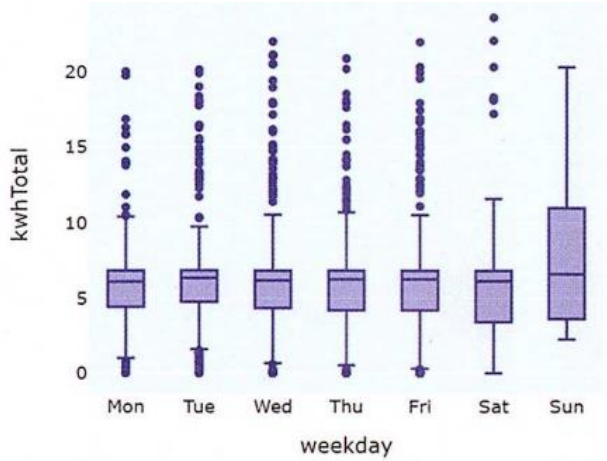
  → 이는 x축으로 사용한 category형 column의 데이터가 원 데이터셋에서 나오는 순서에 영향을 받기 때문

  → Plotly boxplot에서는 “category_orders" 변수에 딕셔너리 형식으로 데이터 순서를 지정해 주면 됨

  → x축 변수명인 “weekday”를 key로 하고, 월요일부터 일요일까지의 순서를 나타내는 리스트를 value로 하는 딕셔너리를 전달하여 boxplot의 x축 순서를 지정

Boxplot, Stripplot, Swarmplot (ch3-4.py)

```
weekday_order = {'weekday': ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']}
fig = px.box(
    data_frame=df, x='weekday', y='kwhTotal',
    width=500, height=400,
    category_orders=weekday_order
)
fig.show()
```



▲ 그림 81 Plotly express boxplot의 x축 데이터 순서를 “category_orders” 인자를 통해 지정하여 그린 그래프

■ <Plotly> Box plot : EV_charge 데이터셋

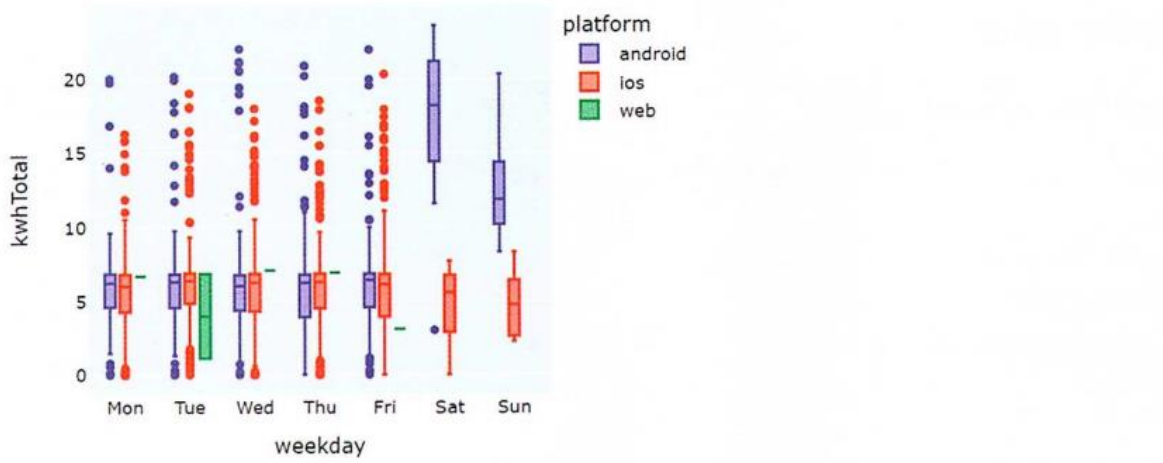
- Plotly를 통해 그린 boxplot 또한 특정 변수의 값 별로 색깔을 구분하여 각 x 축 변수에 대한 box를 나누어서 그리기

  → 사용하는 인자는 Plotly scatterplot. lineplot과 동일하게 “color"를 사용함

  → “color" 인자에 “platform"을 전달하여 해당 변수의 값 별로 box를 색깔별로 나누어 그림

Boxplot, Stripplot, Swarmplot (ch3-4.py)

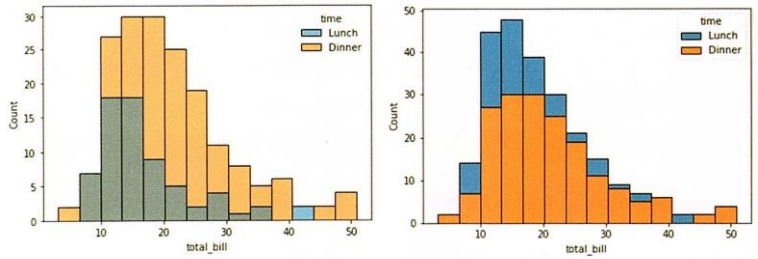
```
weekday_order = {'weekday': ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']}
fig = px.box(
    data_frame=df, x='weekday', y='kwhTotal',
    width=500, height=400,
    category_orders=weekday_order, color='platform'
)
fig.show()
```



▲ 그림 82 Plotly express boxplot에서 “color” 인자를 사용하여 특정 변수의 값 별로 box를 나누어 그린 그래프

■ **<Matplotlib & Seaborn>**

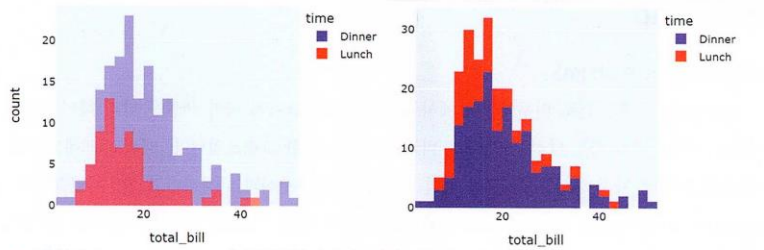
- Seaborn histplot은 하나 혹은 두 변수에 대한 데이터의 분포를 보여주는 그래프



▲ 그림 85 Seaborn histplot에서 특정 변수 그룹별 색깔 구분. multiple 인자를 전달하지 않았을 때(왼쪽)와 multiple 인자로 'stack'을 전달했을 때(오른쪽)

■ **<Plotly>**

- Plotly 라이브러리의 express 패키지의 histogram 함수를 이용하여 Plotly에서도 간단하게 히스토그램을 그릴 수 있음



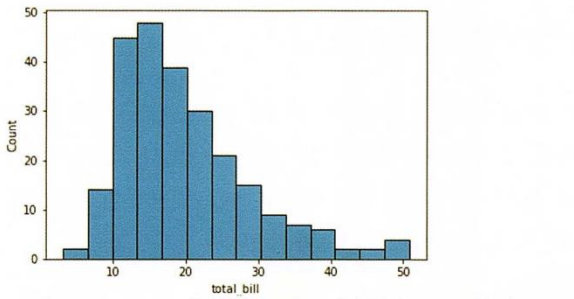
▲ 그림 88 Plotly express histogram에서 특정 변수의 값 별로 막대를 나누어 그린 그래프. "barmode" 인자를 "overlay"로 지정했을 때와 (왼쪽) "relative"로 지정했을 때 (오른쪽)

■ <Matplotlib & Seaborn> Histplot : tips 데이터셋

- total_bill을 x축으로 하는 히스토그램을 그리기

Histplot (ch3-5.py)

```
df = sns.load_dataset('tips')

fig, ax = plt.subplots()
sns.histplot(df, x='total_bill', ax=ax)
```



▲ 그림 83 Seaborn histplot을 이용하여 그린 tips 데이터셋의 total_bill 변수 히스토그램

- histplot에서는 bins와 binwidth 두개 인자 중 하나를 이용하여 bin의 크기를 조정

→ bins 인자는 총 몇 개의 bin을 (막대를) 그릴 것인지를 입력받고, binwidth는 하나의 bin 크기(길이)를 전달받음

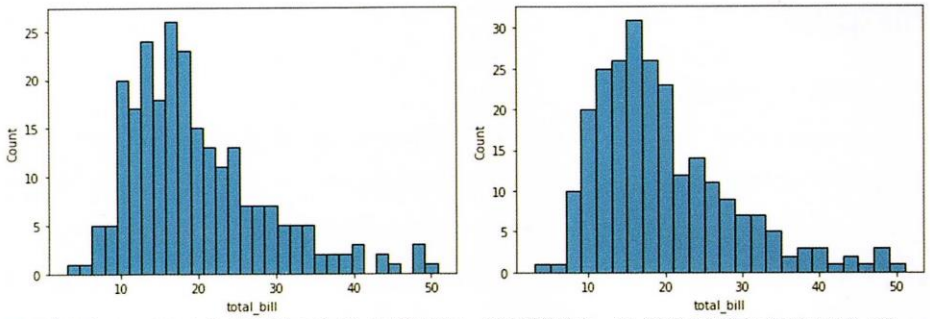
→ binwidth 인자에 2를 전달하여 하나의 bin이 가지는 길이를 2로 하여 그림

Histplot (ch3-5.py)

```
fig, ax = plt.subplots()
sns.histplot(df, x='total_bill', ax=ax, bins=30)
```

Histplot (ch3-5.py)

```
fig, ax = plt.subplots()
sns.histplot(df, x='total_bill', ax=ax, binwidth=2)
```



▲ 그림 84 Seaborn histplot을 이용하여 그린 히스토그램에서 bins 인자와(왼쪽) binwidth 인자(오른쪽)를 통한 막대 개수 조정

■ <Matplotlib & Seaborn> Histplot : tips 데이터셋

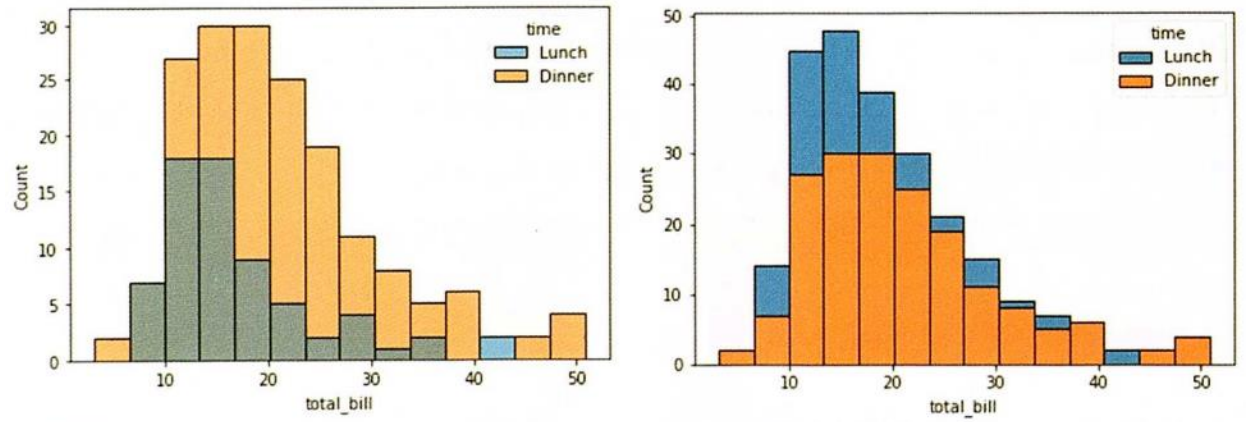
- tips 데이터셋 에서 hue 인자에 “time” 변수를 전달하여 시간대에 따라 막대의 색을 구분하여 히스토그램 그리기
- histplot 역시 특정 변수의 그룹을 색상별로 구분하여 표시
  → hue 인자를 통해 변수명을 전달
- multiple 인자에 아 무것도 전달하지 않으면 위 그림의 왼쪽 그래프와 같이 두 색깔에 해당하는 그룹이 겹쳐서 표현
- 그룹별 색깔 구분을 하지만 각 bin에 해당하는 데이터들을 위로 쌓아서 count (y축의 길이)가 누적되게끔 히스토그램을 그려야 할 때가 있는데, 이 때에는 multiple 인 자에 'stack'을 전달

Histplot (ch3–5.py)

```
fig, ax = plt.subplots()
sns.histplot(df, x='total_bill', ax=ax, hue='time')
```

Histplot (ch3–5.py)

```
fig, ax = plt.subplots()
sns.histplot(df, x='total_bill', ax=ax, hue='time', multiple='stack')
```



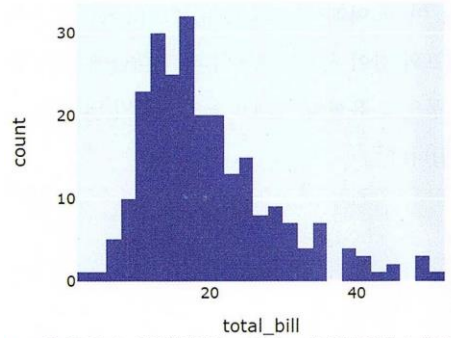
▲ 그림 85 Seaborn histplot에서 특정 변수 그룹별 색깔 구분. multiple 인자를 전달하지 않았을 때(왼쪽)와 multiple 인자로 'stack'을 전달했을 때(오른쪽)

■ <Plotly> Histplot : tips 데이터셋

- tips 데이터셋을 이용하여 “total_bill" 변수 의 분포를 Plotly histogram 그리기

Histplot (ch3-5.py)

```
df = sns.load_dataset('tips')

fig = px.histogram(data_frame=df, x='total_bill', width=450)
fig.show()
```



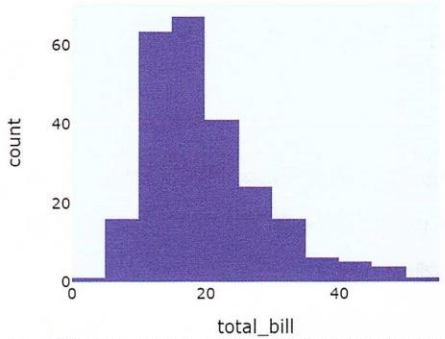
▲ 그림 86 Plotly 라이브러리로 그린 tips 데이터셋의 “total_bill” 변수의 분포 히스토그램

- Plotly로 그린 히스토그램의 bin 크기를 조절하기 위해서는 “nbins" 인자에 bin의 개수를 전 달하는 방법
- “nbins" 인자에 20을 전달

  → “nbins" 인자에 전달되는 값이 클수록 bin 개수가 증가하고, 작을수록 감소

Histplot (ch3-5.py)

```
fig = px.histogram(
    data_frame=df, x='total_bill', width=450, nbins=20
)
fig.show()
```



▲ 그림 87 Plotly로 그린 tips 데이터셋의 “total_bill” 히스토그램에서 bin의 크기를 증가시킨 그래프

■ <Plotly> Histplot : tips 데이터셋

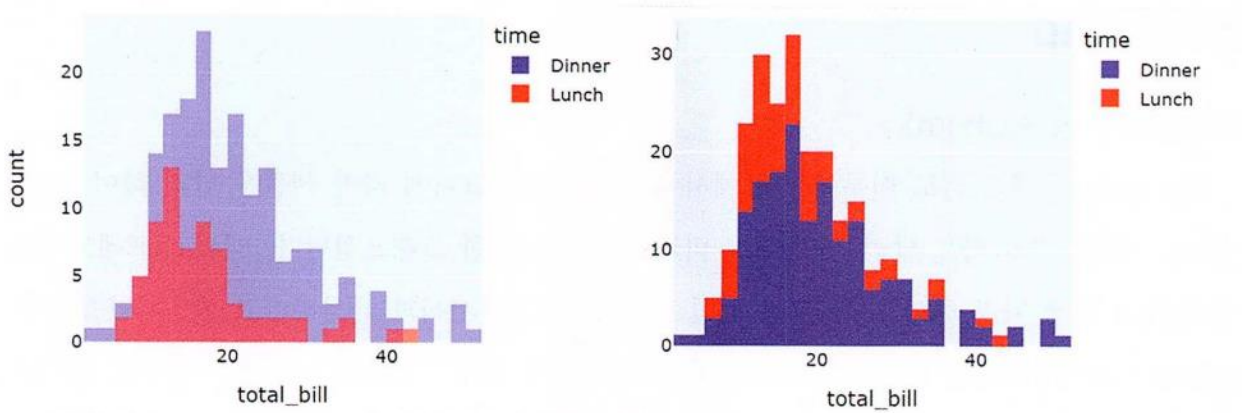
- “color"인자를 사용하여 특정 변수의 값 별로 히스토그 램 막대를 나누어서 그리기
  → 해당 인자에 “time”을 전달하여 시간대 별로 “total bill" 변수의 값 분포를 확인
  →“barmode" 인자에 “overlay"를 지정하게 되면 아래 그림의 좌측 그래프와 같이 히스토그램의 각 막대들에 대한 값 이 0부터 시작되면서 중첩되어 그려짐
  → 동일한 유형의 그래프를 중첩하지 않게끔 그 리고 싶다면 “overlay" 대신 "group"을 전달

Histplot (ch3-5.py)

```
fig = px.histogram(
    data_frame=df, x='total_bill', width=450,
    color='time', barmode='overlay'
)
fig.show()
```

Histplot (ch3-5.py)

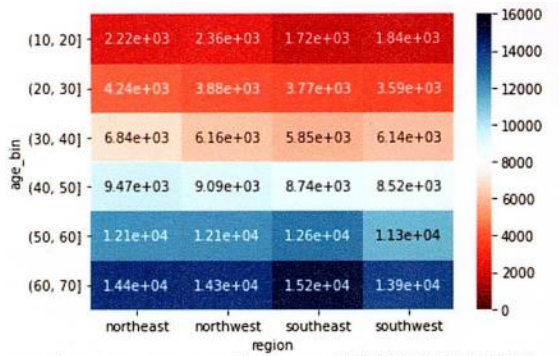
```
fig = px.histogram(
    data_frame=df, x='total_bill', width=450, color='time'
)
fig.show()
```



▲ 그림 88 Plotly express histogram에서 특정 변수의 값 별로 막대를 나누어 그린 그래프. “barmode” 인자를 “overlay”로 지정했을 때와 (왼쪽) “relative”로 지정했을 때 (오른쪽)

■ <Matplotlib & Seaborn>

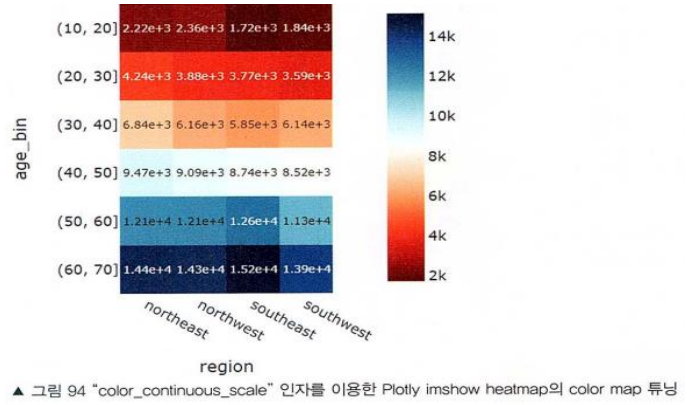
- Heatmap은 행과 열로 이루어진 표에서 각 셀의 값의 크기에 따라 색깔을 달리 하여, 단순 숫자로 이루어진 표와는 달리 값의 대소 비교를 용이하게 한 그래프



▲ 그림 91 Seaborn heatmap의 color bar 범위 설정 및 셀 색깔 변경

■ <Plotly>

- Matplotlib 및 Seaborn을 통해 구현하든 Plotly를 통해 구현하는 x, y축을 지정하는 방식은 크게 다르지 않음



▲ 그림 94 "color_continuous_scale" 인자를 이용한 Plotly imshow heatmap의 color map 튜닝

■ <Matplotlib & Seaborn> Heatmap : medical_cost 데이터셋

- Medical cost 데이터셋은 의료보험 회사가 지불한 비용을 의료보험 혜택을 받은 수혜자의 나이, 성별, BMI, 의료 보험 혜택을 받을 수 있는 아이 수, 흡연 여부, 지역 정보와 함께 나타냄

Heatmap (ch3-6.py)

```
df = pd.read_csv('./datasets/medical_cost/medical_cost.csv')
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

▲ 그림 89 medical_cost 데이터셋에서 첫 5개 행을 살펴본 결과

■ <Matplotlib & Seaborn> Heatmap : medical_cost 데이터셋

- Seaborn 라이브러리의 heatmap을 그리려면 2차원 Numpy array 형태의 데이 터나, pivot 함수로 정제된 Pandas dataframe이 필요함
- 가로 지 역으로, 세로축을 구간화된 나이로 하고, 각 셀에는 해당 나이 구간과 지역에 해당하는 의료보험료의 중간값을 갖는 pivot table 만들기
- Numpy의 arange를 이용하여 10부터 70까지, 10 단위로 하는 구간을 만들었고, Pandas의 cut 함수를 이용하여 데이터셋의 나이(age' 열) 데이터를 이용하여 각 나이 구간에 해당하는 열 age bin을 새로 추가

Heatmap (ch3-6.py)

```
age_bin_list = np.arange(10, 80, 10)
df['age_bin'] = pd.cut(df['age'], bins=age_bin_list)

pivot_df = df.pivot_table(
    index='age_bin', columns='region', values='charges', aggfunc='median'
)
```

■ <Matplotlib & Seaborn> Heatmap : medical_cost 데이터셋

- Seaborn 라이브러리로 heatmap 그리기

  → annot 인자에 True를 전달하면 위 그림처럼 각 셀에 해당하는 값이 숫자로 함께 표현

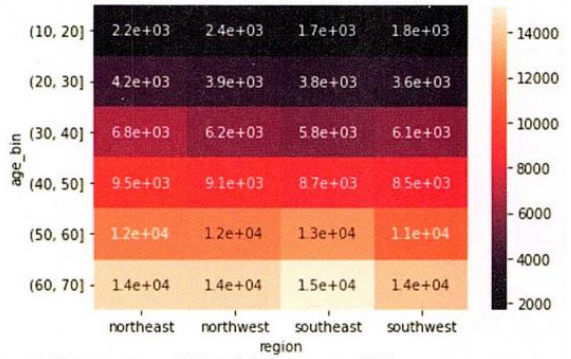
  → fmt 인자에 적절한 formatting 문자열을 전달하면 표현되는 annotation 숫자의 형식을 변경할 수 있음

  → 각 셀의 값이 소수점 아래 1자리까지 가지는 exponential 형태로 표현

- 이를 소수점 2자리까지 표현되도록 변경 → fmt 인자에 '. 2e'를 전달

Heatmap (ch3-6.py)

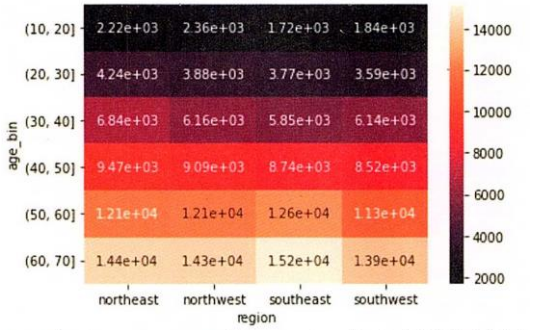
```
fig, ax = plt.subplots()
sns.heatmap(pivot_df, ax=ax, annot=True)
```



▲ 그림 90 Seaborn 라이브러리의 heatmap 예시

Heatmap (ch3-6.py)

```
fig, ax = plt.subplots()
sns.heatmap(pivot_df, ax=ax, annot=True, fmt='.2e')
```



▲ 그림 91 Seaborn heatmap에서 annotation 되는 숫자의 형식 지정하기

■ <Matplotlib & Seaborn> Heatmap : medical_cost 데이터셋

- color bar의 최대, 최소를 각각 16000, 0으로 설정하고, cmap 인자를 변경하 여 셀 값이 작을수록 빨간색, 클수록 파란색으로 표현

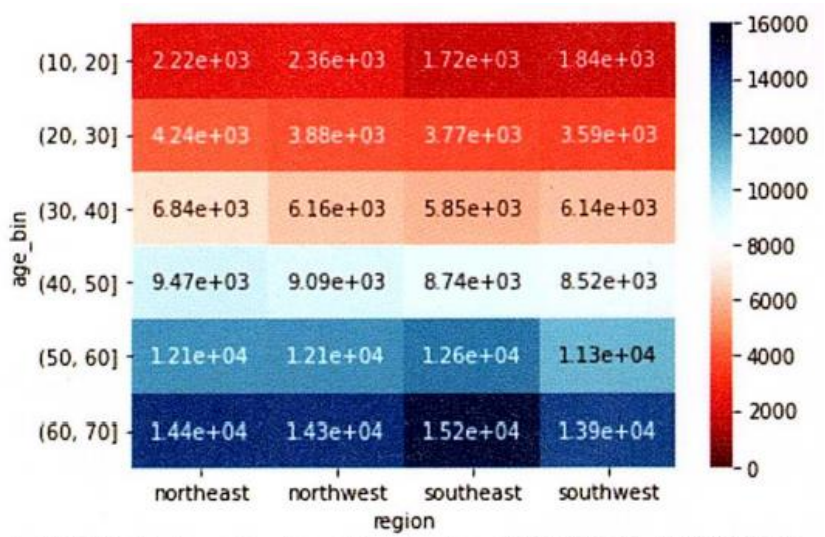
  → heatmap의 오른편에는 값의 크기에 따라 적용되는 색깔을 표시해 주는 color bar가 있음

  → color bar의 최대 및 최소값의 범위를 vmax, vmin 인자를 설정해 줌으로써 변경할 수 있음

  → 셀 값의 크기에 따라 표현되는 색깔을 cmap 인자를 전달함으로써 변경할 수 있음

Heatmap (ch3-6.py)

```
fig, ax = plt.subplots()
sns.heatmap(
    pivot_df, ax=ax, annot=True, fmt='.2e',
    vmax=16000, vmin=0, cmap='RdBu'
)
```



▲ 그림 91 Seaborn heatmap의 color bar 범위 설정 및 셀 색깔 변경

■ <Plotly> Heatmap : medical_cost 데이터셋

- Plotly express에서 heatmap을 그리기 위해서는 imshow 함수를 사용

  → imshow 함수에 데이터뿐만 아니라 x축 및 y축을 함께 전달

  → x축으로는 pivot_df 데이터프레임의 column을 전달

  → 축으로 는 index를 전달하

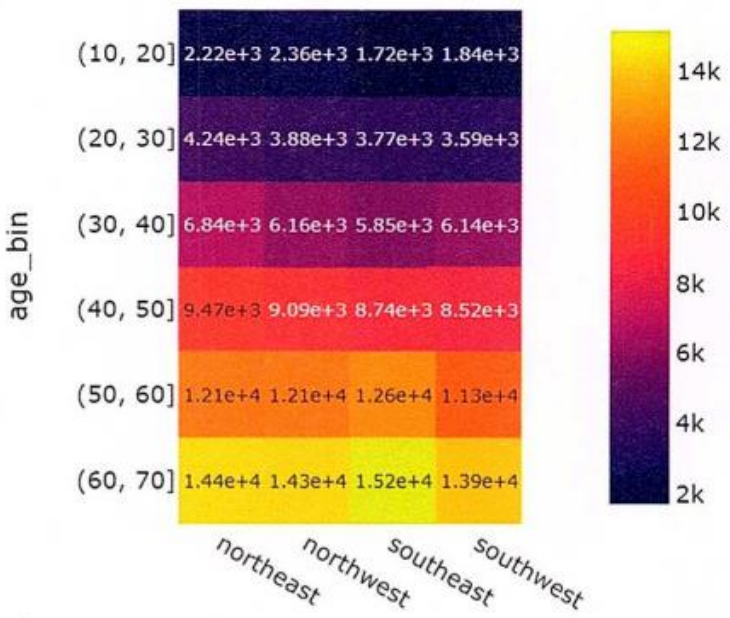
  → Index를 전달할 때 문자열로 데이터 타입을 변경

  (pivot_df 데이터프레임의 index가 [10, 20)과 같은 interval 형식이기 때문)

  → “text_auto" 인자에 숫자 표현식을 “.2e”로 전달

Heatmap (ch3-6.py)

```
fig = px.imshow(
    pivot_df, x=pivot_df.columns, y=pivot_df.index.astype('str'),
    text_auto='.2e', width=400, height=400
)
fig.show()
```



▲ 그림 93 Plotly express의 imshow를 통해 그린 heatmap

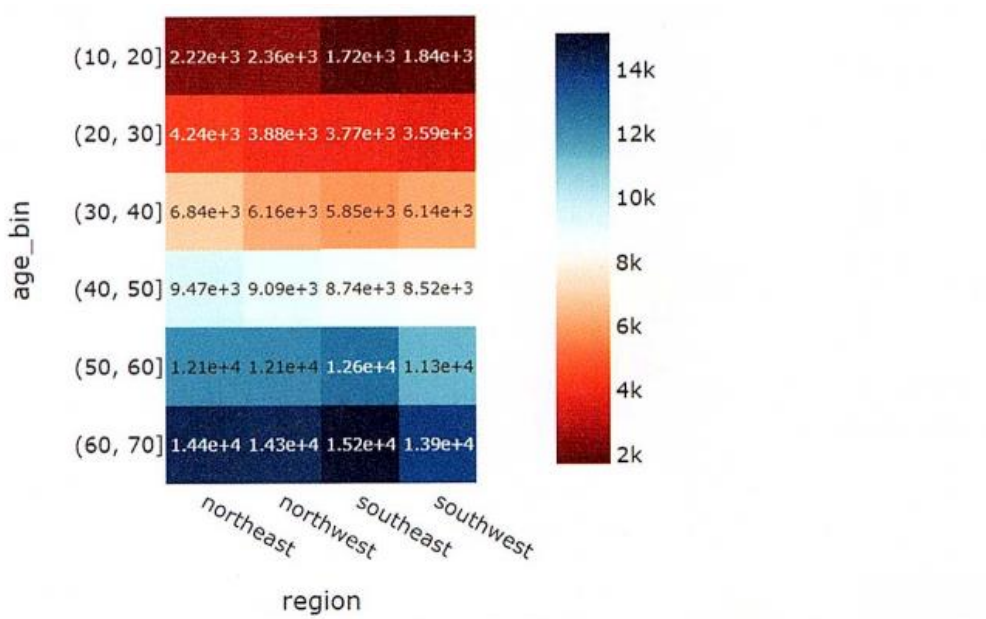
■ <Plotly> Heatmap : medical_cost 데이터셋

- Plotly 라이브러리도 Seaborn과 비슷하게 다양한 color palette를 제공

  → “color_continuous scale" 인자를 통해 color map을 변경할 수 있음

Heatmap (ch3-6.py)

```
fig = px.imshow(
    pivot_df, x=pivot_df.columns, y=pivot_df.index.astype('str'),
    text_auto='.2e',width=400, height=400,
    color_continuous_scale='RdBu'
)
fig.show()
```



▲ 그림 94 “color_continuous_scale” 인자를 이용한 Plotly imshow heatmap의 color map 튜닝

■ Axes-level plot

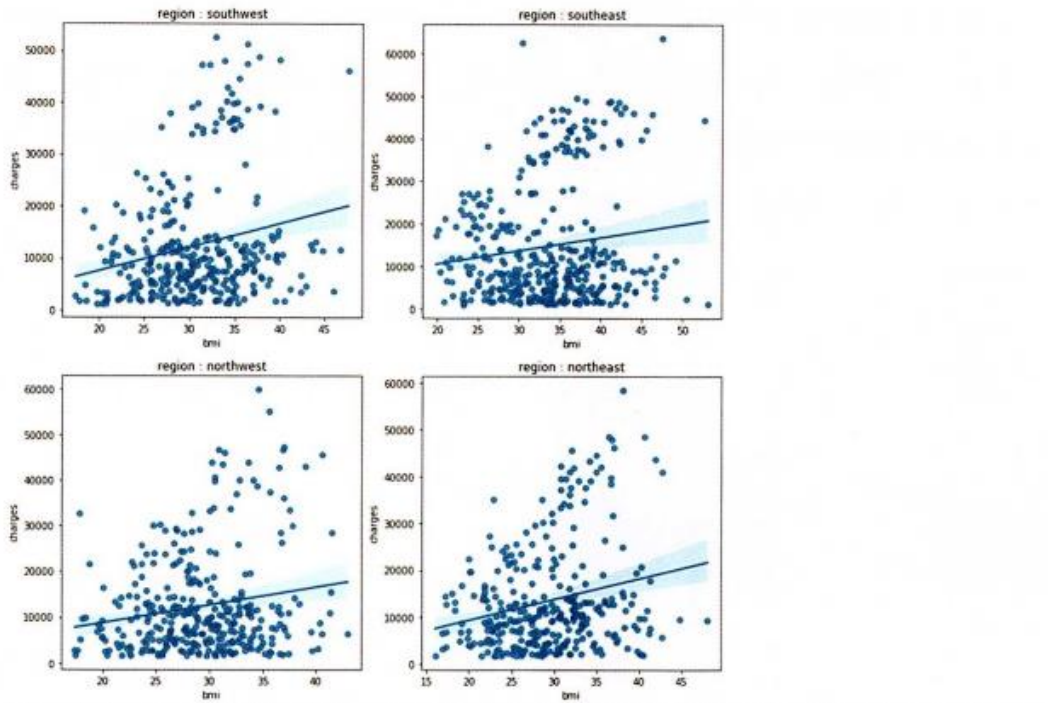
- Figure 안에 사용자가 임의로 설정한 하나 이상의 axes에 각각 그래프를 그리고 커스터마이징 할 수 있음
- 변수 내 그룹의 개수에 해당하는 ax들을 생 성하고, 반복문을 통하여 특정 그룹별 서브 데이터셋을 생성하여 생성한 각 ax들에 하나하나 그래프를 생성하는 작업을 해야 함

■ Figure-level plot

- axes 위에 그래프를 그리는 것이 아니기 때문에 인자로 ax를 받지 않으며, 하나의 figure 위에 그려짐
- 특정 변수 내 데이터 그룹별로 그래프를 각각 그려야 하는 경우 구분할 변수명과 관련된 관련 인자 몇개만 전달하면 손쉽게 그래프를 나눠서 그릴 수 있음

■ Axes-level plot : medical_cost 데이터셋

- 4개의 axes에 medical_cost 데이터셋을 이용하여 그래프 그리기
- ax는 ax [행 번호][열 번호]처럼 2차원 리스트 형태로 호출
- axes-level plot을 통해 각 ax 하나하나 설정하여 그래프를 그릴 수 있음



▲ 그림 96 medical_cost 데이터셋의 4가지 region 변수의 그룹에 대해 각각을 seaborn regplot으로 나타낸 그림

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

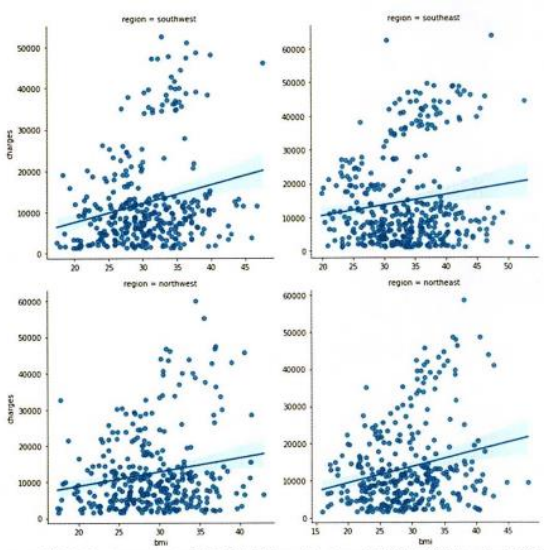
```
df = pd.read_csv('./datasets/medical_cost/medical_cost.csv')

fig, ax = plt.subplots(2, 2, figsize=(12, 12))
sns.regplot(
    x='bmi', y='charges', data=df.query('region == "southwest"'),
    ax=ax[0][0]
)
ax[0][0].set_title('region : southwest')

sns.regplot(
    x='bmi', y='charges', data=df.query('region == "southeast"'),
    ax=ax[0][1]
)
ax[0][1].set_title('region : southeast')

sns.regplot(
    x='bmi', y='charges', data=df.query('region == "northwest"'),
    ax=ax[1][0]
    )
ax[1][0].set_title('region : northwest')

sns.regplot(
    x='bmi', y='charges', data=df.query('region == "northeast"'),
    ax=ax[1][1]
    )
ax[1][1].set_title('region : northeast')
```

■ Figure-level plot : medical_cost 데이터셋

- medical_cost 데이터셋을 이용하여 4개의 그래프 그리기
- Seaborn 라이브러리의 lmplot을 이용
- col_wrap 인자는 하나의 figure에서 몇 개 의 열을 그린 다음 행으로 넘어갈지를 입력하는 인자
- col_wrap에 2를 전달하여 2열까지만 그리 고 다음 행에 이어서 그래프를 그리게끔 설정
- sharex와 sharey 인자는 x축 과 y축의 범위를 공유(동일하게)하도록 설정하는 인자
  → 인자들을 False로 설정하여 각 각의 그래프마다 다른 x와 y 범위를 가지게끔 설정

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

```
sns.lmplot(
    x='bmi', y='charges', data=df,
    col='region', col_wrap=2,
    sharex=False, sharey=False
)
```



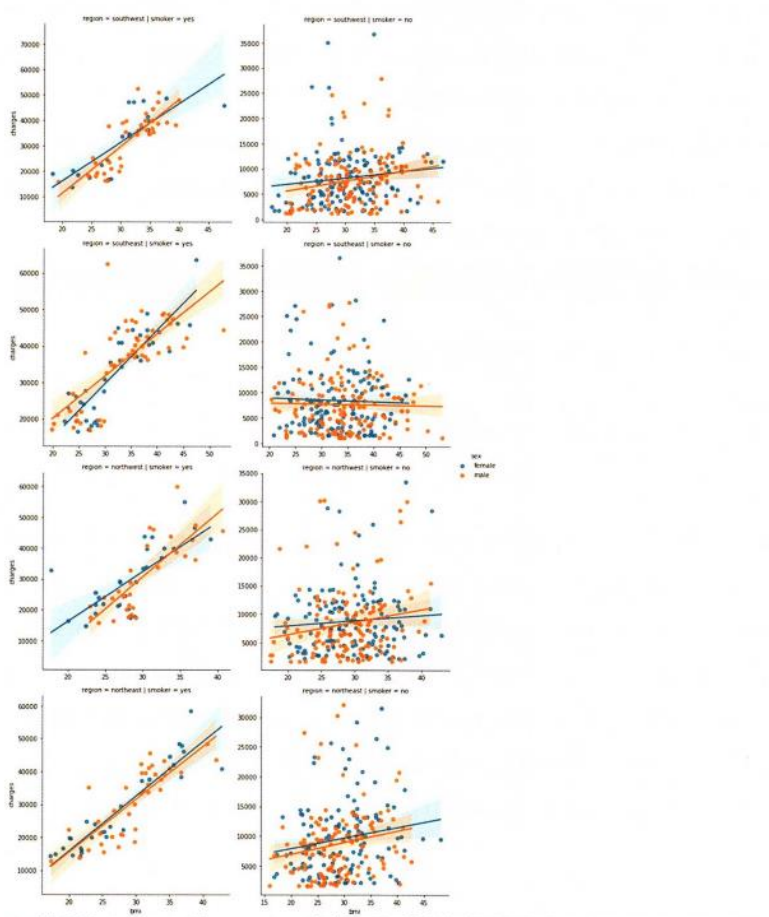
▲ 그림 97 Seaborn lmplot을 이용하여 medical_cost 데이터셋의 "region" 별로 regression plot을 나타낸 그림

■ Figure-level plot : medical_cost 데이터셋

- medical_cost 데이터셋의 “region"을 구분하여 그래프를 그리고, col는 “smoker"를 구분하여 그래프 그리기
- lmplot은 regplot에서는 사용할 수 없었던 hue 인자를 사용할 수 있음
  → hue 인자를 sex로 지정하여 추가

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

```
sns.lmplot(
    x='bmi', y='charges', data=df,
    col='smoker', row='region', hue='sex',
    sharex=False, sharey=False
)
```



▲ 그림 98 Seaborn lmplot의 col, row, hue 인자를 모두 사용하여 여러 변수의 각 그룹 별 그래프를 나눠 그리기

■ Axes-level plot : medical_cost 데이터셋

- Seaborn의 FacetGrid 객체를 이용

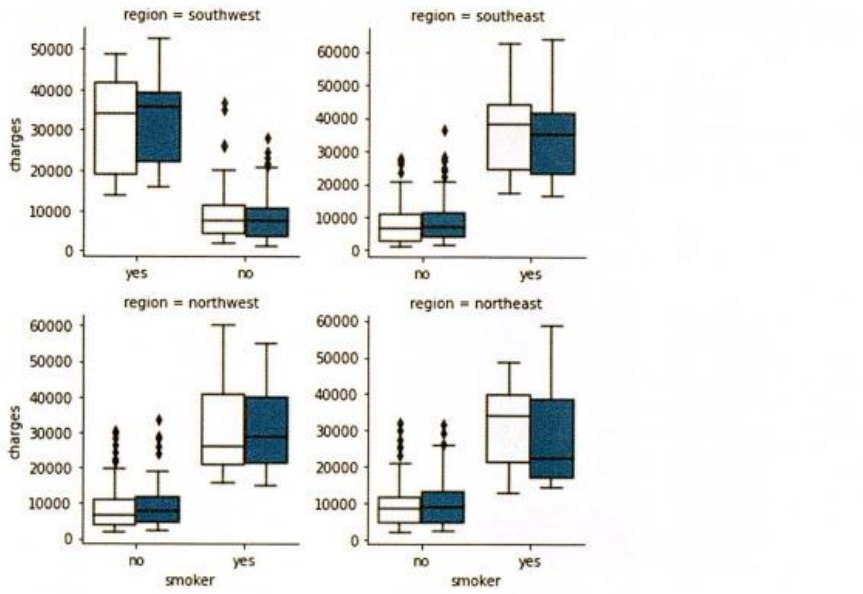
  → FacetGrid는 특정 데이터셋을 인자로 받아서 지정한 변수의 그룹별로 데이터를 열 및 행 으로 나누어 놓을 수 있음

- 데이터를 특정 변수의 그룹 별로 나눈 후에 boxplot과 같이 원하 는 ax-level plot을 각 열 및 행에 mapping하여 뿌려줄 수 있음
- FacetGrid를 이용하여 medical_cost 데이터셋을 region 변수별로 나누어서 “smoker"에 대 한 "charges"를 보여주는 boxplot 그리기

  → hue 인자에 “sex"를 전달 하여 성별로 box를 나누어 그리기

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

```
g = sns.FacetGrid(
    data=df, col='region', col_wrap=2, sharex=False, sharey=False
)
g.map_dataframe(
    sns.boxplot, x='smoker', y='charges', hue='sex'
)
```



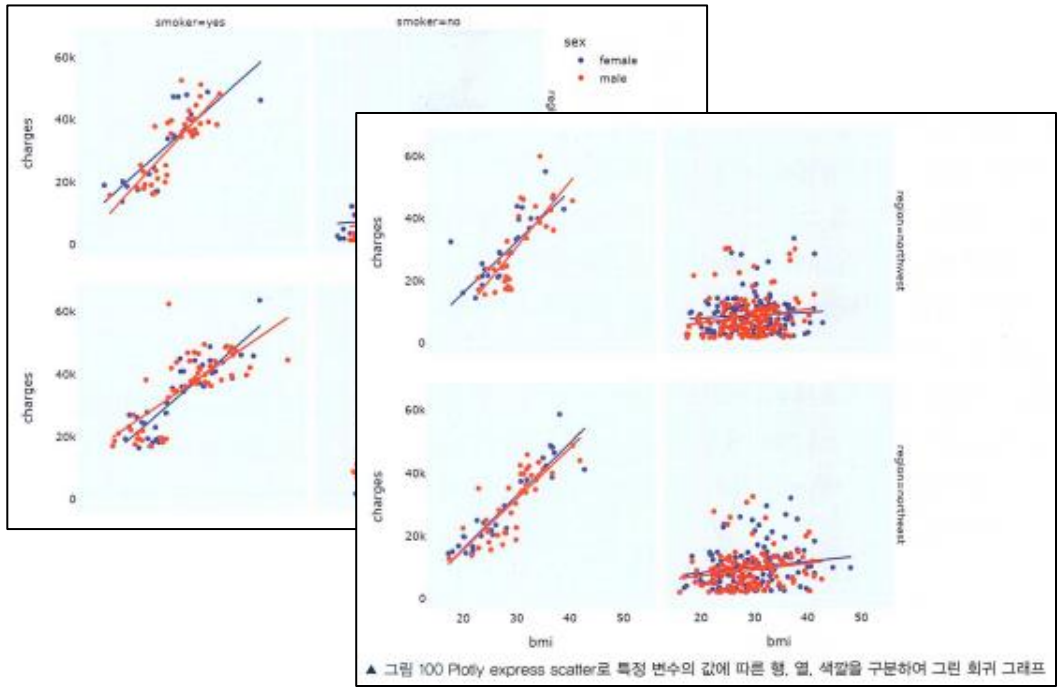
▲ 그림 99 Seaborn FacetGrid를 이용한 특정 변수 그룹별 행과 열을 구분하여 axes-level 그래프 그리기

■ <Plotly> Plotly 라이브러리에서는 각 그래프 함수에서 facet 인자 사용 : medical_cost 데이터셋

- lotly 라이브러리의 scatter 함수는 Seaborn의 Implot과 유사하게 “facet_row" 및 “facet_col"에 각각 행과 열로 구분할 변 수명을 전달하여 손쉽게 그래프를 행과 열로 나누는 기능을 제공
- Plotly scatter를 이용하여 medical_cost 데이터셋의 “bmi"에 대한 “charges" 그래프를 “region” 변 수와 “smoker" 변수 값에 따라 행과 열로 나누어 그리고, 이 때 각 그래프에서 성별에 따른 색 깔 구분 그리기

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

```
fig = px.scatter(
    data_frame=df, x='bmi', y='charges',
    color='sex', facet_row='region', facet_col='smoker',
    width=700, height=1200, trendline='ols'
)
fig.show()
```



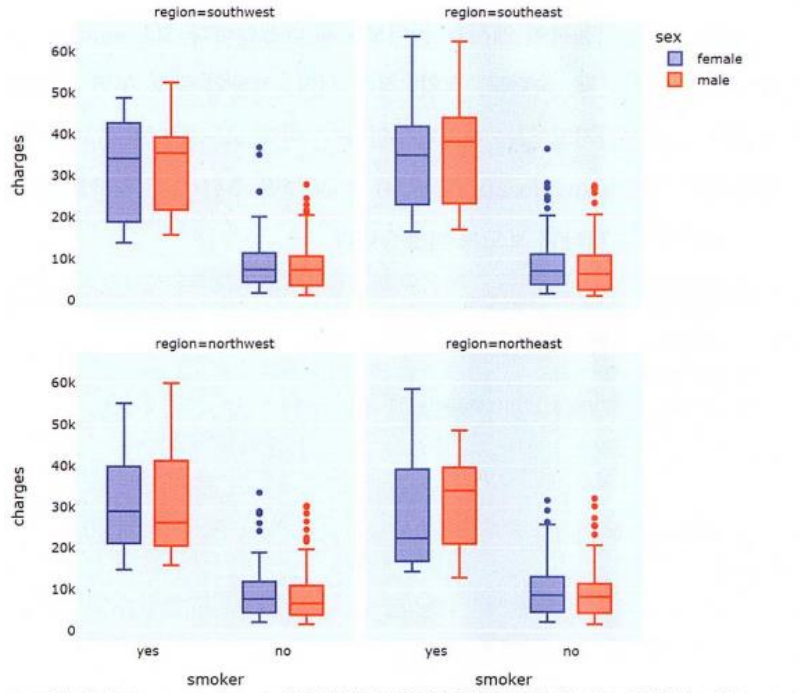
▲ 그림 100 Plotly express scatter로 특정 변수의 값에 따른 행, 열, 색깔을 구분하여 그린 회귀 그래프

■ <Plotly> Plotly 라이브러리에서는 각 그래프 함수에서 facet 인자 사용 : medical_cost 데이터셋

- lotly 라이브러리의 scatter 함수는 Seaborn의 Implot과 유사하게 “facet_row" 및 “facet_col"에 각각 행과 열로 구분할 변 수명을 전달하여 손쉽게 그래프를 행과 열로 나누는 기능을 제공
- Plotly scatter를 이용하여 medical_cost 데이터셋의 “bmi"에 대한 “charges" 그래프를 “region” 변 수와 “smoker" 변수 값에 따라 행과 열로 나누어 그리고, 이 때 각 그래프에서 성별에 따른 색 깔 구분 그리기

Matplotlib의 axes-level plot과 figure-level plot (ch3-7.py)

```
fig = px.box(
    data_frame=df, x='smoker', y='charges',
    facet_col='region', facet_col_wrap=2, color='sex',
    width=700, height=700
)
fig.show()
```



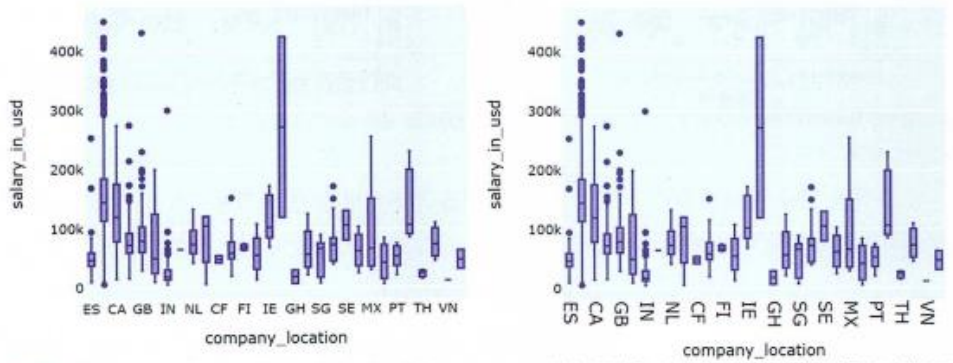
▲ 그림 101 Plotly express boxplot을 이용하여 특정 변수의 값에 따라 행, 열을 구분하여 그린 boxplot

■ <축 라벨 튜닝하기>

- ticklabel의 회전은 회전시키고자 하는 tick이 포함된 그래프가 그려진 ax에서 tick_params 메서드를 통해 할수 있음
- ticklabel을 회전시킬 때와 동일한 tick_params 메서드를 사용하여 x축과 y축 모두 글씨 크기를 조절하기 위해 axis 인자에 both를 전달
- major ticklabel에 대한 글씨 크기 조절이므로 which 인자에 major를 지정하고, labelsize를 지정하여 원하는 글씨 크기를 입력

그래프 세부 요소 튜닝 (ch3-8.py)

```
fig = px.box(
    data_frame=df_10companies, x='company_location', y='salary_in_usd',
    width=500, height=400)
fig.update_xaxes(tickfont={'size':16}, tickangle=90)
fig.show()
```



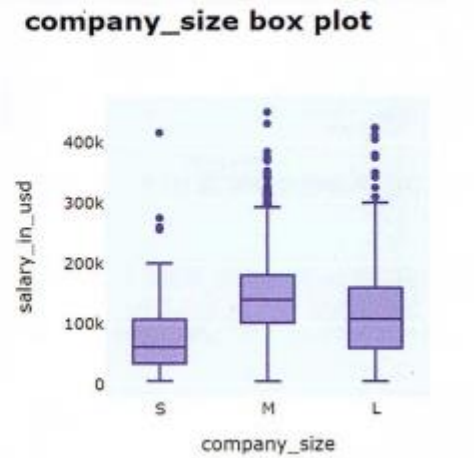
▲ 그림 104 Plotly로 그린 boxplot (왼쪽). 해당 그래프에서 update_xaxes 메서드를 사용하여 xticklabel의 글씨 크기를 16pt와 90도 회전한 그래프 (오른쪽)

■ <그래프 제목 입력〉

- ax 별로제목을 지정할 때에는 아래 코드처럼 set_title 메서드를 사용함
- Plotly express의 여러 그래프들을 그리는 함수들에는 "title" 인자가 내장되어 있는 경우가 대부분
- 이때 그래프 제목의 글씨 크기나 글씨체 등을 바꾸고자 한다면 “update_layout" 메서드를 사용

그래프 세부 요소 튜닝 (ch3-8.py)

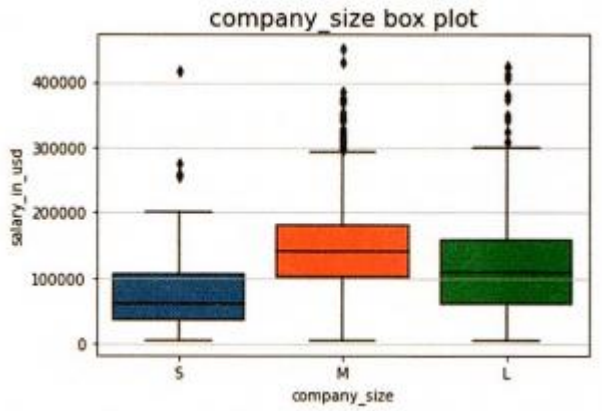
```
fig = px.box(
    data_frame=df, x='company_size', y='salary_in_usd',
    width=400, height=400,
    title='<b>company_size box plot</b>',
    category_orders={'company_size':['S','M','L']}
)
fig.update_layout({'title_font_size':20})
fig.show()
```



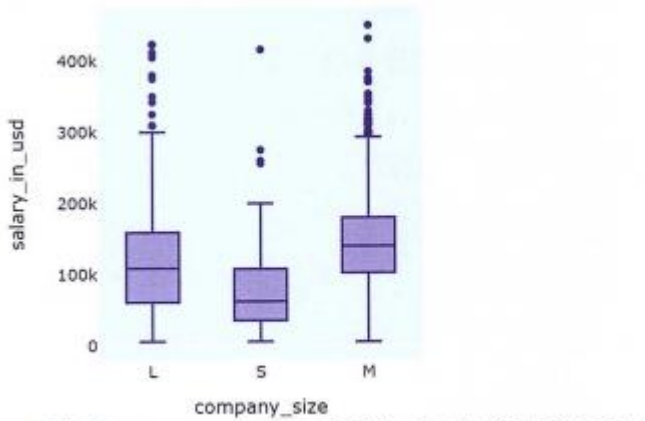
▲ 그림 106 Plotly express로 그린 그래프에 제목 추가하기

■ <Grid 표시>

- grid는 각 ax마다 표시해 줄 수 있는데, grid 메서드를 호출
- 이 때 x축과 y축 모두 혹은 둘 중 하나에 대한 눈금선만 그리도록 지정해 줄 수 있는데,
  axis 인자에 보조선을 나타내고자 하는 축 이름을 전달
- x, y 두축 모두 보조선을 표시하기 위해서는 'both'를 전달하거나, axis 인자를 따로 명시하여 전달하지 않아도 됨
  (기본값이 axis="both')



▲ 그림 107 메서드를 이용하여 y축에 대한 보조선 그리기



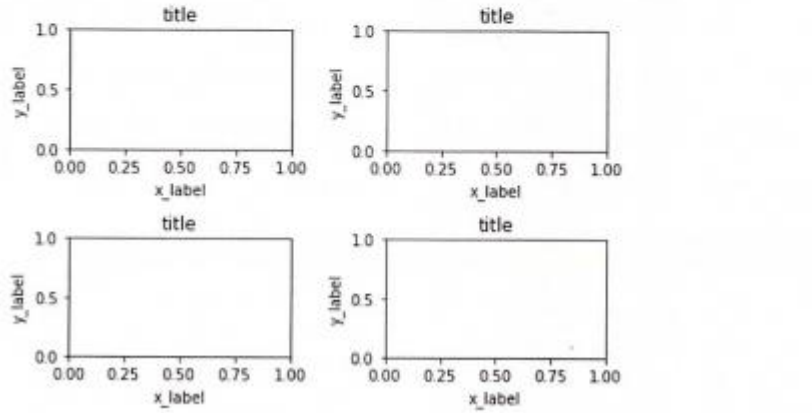
▲ 그림 108 Plotly express로 그린 그래프에서 y축 gridline을 삭제한 그래프

■ <Figure 내 각 ax 세부 위치 조절>

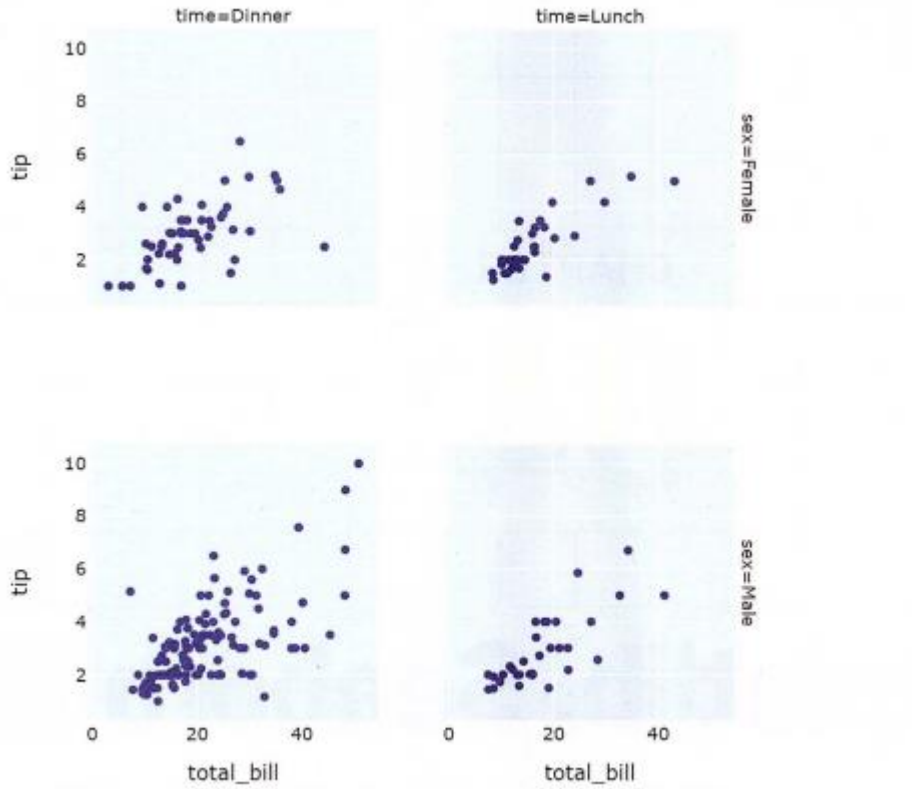
- matplotlib pyplot의 tight_layout 메서드 : ax의 그래프의 크기 조정
- W_pad, h_pad, pad 인자를 따로 명시하여 전달 : 각 ax간의 여백을 보다 세밀하게 조절

그래프 세부 요소 튜닝 (ch3-8.py)

```
fig, ax = plt.subplots(2, 2)
for idx in range(2):
    for jdx in range(2):
        ax[idx][jdx].set_xlabel('x_label')
        ax[idx][jdx].set_ylabel('y_label')
        ax[idx][jdx].set_title('title')

plt.tight_layout()
```



▲ 그림 110 tight_layout 메서드를 이용하여 각 ax 그래프가 최대한 겹치지 않도록 크기를 최적화



▲ 그림 111 Plotly express scatterplot에서 facet을 나누어 그렸을 때 각 subplot 간의 공백을 조절한 그래프