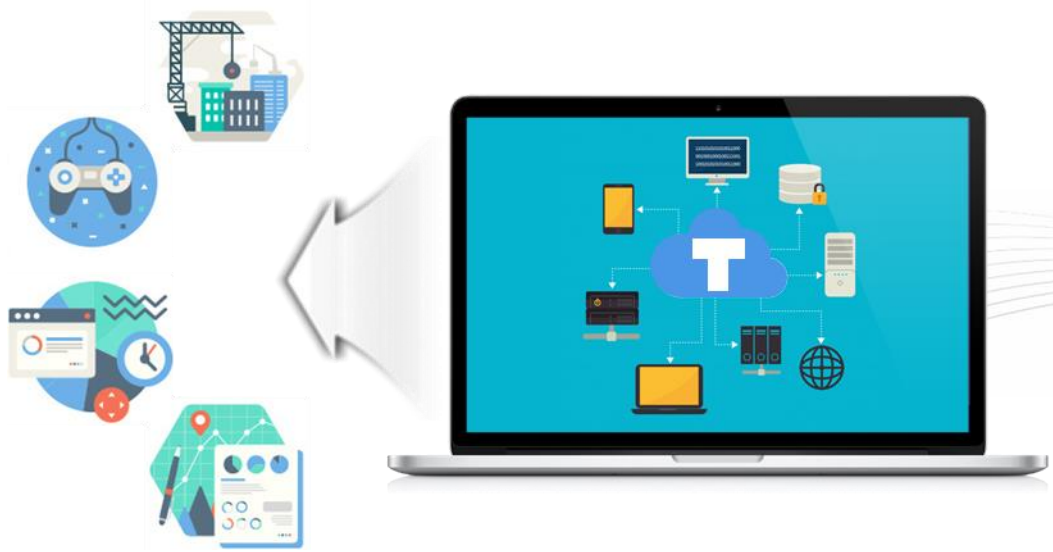
AIoT 데이터 시각화 및 대시보드 개발



Chapter 04.
Matplotlib, Seaborn 및 Plotly 라이브러리 실전 꿀팁 대방출

Ⅰ x축이 날짜인 시계열 그래프 tick label 깔끔하게 표현하기

Ⅱ 다중 축 그래프 그리기

Ⅲ 범례(legend) 위치 조절하기

Ⅳ 그래프의 테두리(spine) 강조하기

Chapter 04.
Matplotlib, Seaborn 및 Plotly 라이브러리 실전 꿀팁 대방출

Ⅴ 그래프 내 텍스트 표현하기

Ⅵ 수평선과 수직선 그리기

Ⅶ FacetGrid로 나눈 각 ax별 mapping으로 그래프 세부 튜닝하기

Ⅷ FacetGrid에서 특정 조건에 해당하는 ax 강조하기

Chapter 04.
Matplotlib, Seaborn 및 Plotly 라이브러리 실전 꿀팁 대방출

IX regplot의 선형회귀선의 식과 상관계수 표시

X 그래프의 축 log 형식으로 변환하기

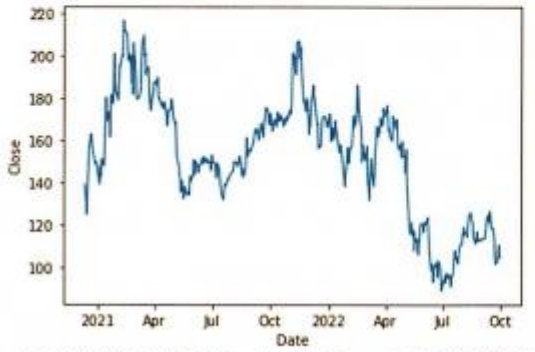
A Seaborn color palette 및 Plotly color의 활용

■ **x축을 시간이나 날짜로 지정하고 아무런 추가 설정을 하지 않으면 xticklabel이 알아보기 어려운 경우가 많음**

- “Date" 열의 데이터 타입을 datetime으로 바꾸기만 해도 어느 정도는 문제를 해결
- xticklabel을 회전시키거나, xticklabel의 표현 방식을 아예 변경하는 것을 고려
- 시계열 그래프 tick label의 표현 방식 변경
- ConciseDateFormatter 사용

x축이 날짜인 시계열 그래프 tick label 깔끔하게 표현하기 (ch4-1.py)

```
import matplotlib as mpl

fig, ax = plt.subplots()
sns.lineplot(x='Date', y='Close', data=df, ax=ax)
ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))
```



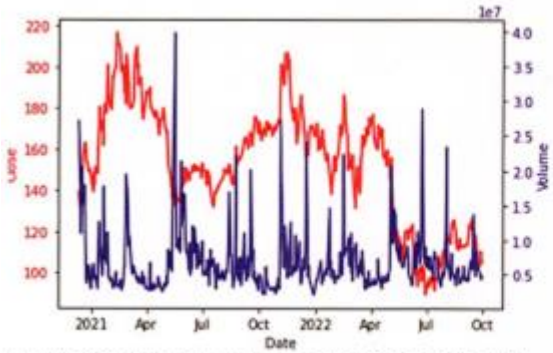
▲ 그림 114 Matplotlib의 ConciseDateFormatter를 이용하여 시계열 그래프의 x축 시인성 향상

■ 하나의 ax에 두개 이상의 x축이나 y축을 그려서 여러 그래프를 동시에 나타내는 경우

- 필요에 따라서 하나의 ax에 두개 이상의 x축이나 y축을 그려서 여러 그래프를 동시에 나타내는 경우도 종종 있음
- 특히 y축을 2개 이상 그리는 경우가 많은데, ABNB_stock 데이터셋을 이용하여 예시

다중 축 그래프 그리기 (ch4-2.py)

```
df = pd.read_csv('./datasets/ABNB_stock/ABNB_stock.csv')
df['Date'] = pd.to_datetime(df['Date'])

fig, ax = plt.subplots()
ax2 = ax.twinx()
sns.lineplot(x='Date', y='Close', data=df, ax=ax, color='red')
sns.lineplot(x='Date', y='Volume', data=df, ax=ax2, color='blue')

ax.tick_params(axis='y', labelcolor='red')
ax.yaxis.label.set_color('red')

ax2.tick_params(axis='y', labelcolor='blue')
ax2.yaxis.label.set_color('blue')

ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))
```



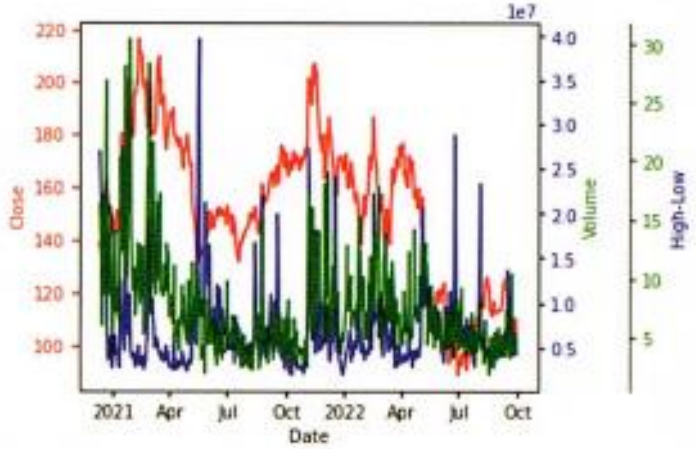
▲ 그림 116 Matplotlib twinx 메서드를 이용한 이중 y축 그래프

■ 하나의 ax에 두개 이상의 x축이나 y축을 그려서 여러 그래프를 동시에 나타내는 경우

- y축을 3개 그리는 경우 예시

다중 축 그래프 그리기 (ch4-2.py)

```
fig, ax = plt.subplots()
fig.subplots_adjust(right=0.75)

ax2 = ax.twinx()
ax3 = ax.twinx()

# 세 번째 y축(ax3)을 그래프의 바깥으로 옮깁니다.
ax3.spines.right.set_position(("axes", 1.2))

sns.lineplot(x='Date', y='Close', data=df, ax=ax, color='red')
sns.lineplot(x='Date', y='Volume', data=df, ax=ax2, color='blue')
sns.lineplot(x='Date', y='High-Low', data=df, ax=ax3, color='green')

ax.yaxis.label.set_color('red')
ax2.yaxis.label.set_color('blue')
ax3.yaxis.label.set_color('green')

tkw = dict(size=4, width=1.5)
ax.tick_params(axis='y', colors='red', **tkw)
ax2.tick_params(axis='y', colors='blue', **tkw)
ax3.tick_params(axis='y', colors='green', **tkw)
ax.tick_params(axis='x', **tkw)

ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))
```



▲ 그림 117 3개의 y축을 가지는 다중 축 그래프

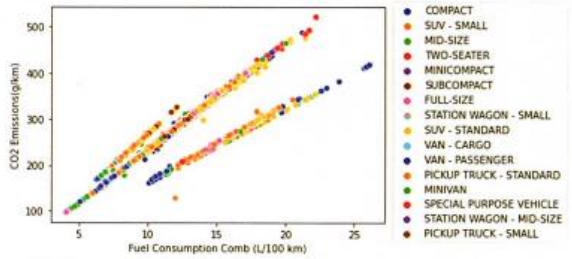
■ 범례(legend) 위치

- legend로 표현해야 할 항목들이 너무 많아 그래프 내부에 legend가 위치했을 때 그래프의 일부를 가리게 되는 문제가 발생 → legend의 위치를 적절하게 조절
- legend 메서드의 bbox_to_anchor인자로 전달된 (1.01, 1.05)

→ 우상단에서 오른쪽 위로 (0.01, 0.05) 만큼 더 떨어진 위치에서 legend를 표시하

범례 (legend) 위치 조절하기 (ch4-3.py)

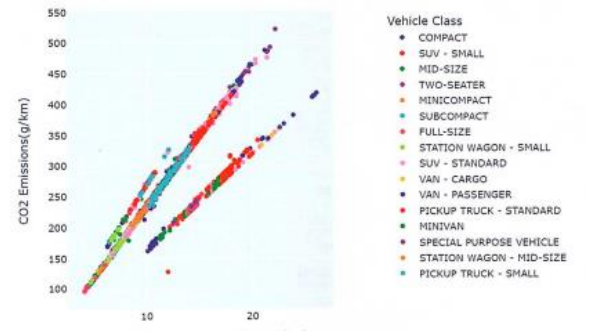
```
fig, ax = plt.subplots()
sns.scatterplot(
    x='Fuel Consumption Comb (L/100 km)',
    y='CO2 Emissions(g/km)',
    data=df, hue='Vehicle Class', palette='bright',
    ax=ax
)
ax.legend(bbox_to_anchor=(1.01, 1.05))
```



▲ 그림 121 matplotlib ax의 legend 위치 변경

범례 (legend) 위치 조절하기 (ch4-3.py)

```
fig = px.scatter(
    df, x='Fuel Consumption Comb (L/100 km)', y='CO2 Emissions(g/km)',
    color='Vehicle Class', width=700, height=500
)
fig.update_layout(legend_x=1.2, legend_y=1)
fig.show()
```



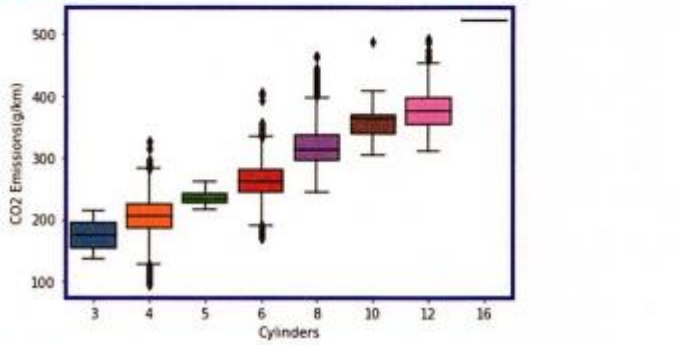
▲ 그림 122 Plotly로 그린 그래프의 범례의 상대위치를 (1.2, 1)로 조절한 결과

■ 그래프의 테두리 색깔과 선 굵기를 조절하여 그래프를 강조하는 법

- 이 방법은 특히 행과 열을 나눠 여러 그래프를 그린 상황에서 특정 조건을 만족하는 data가 있는 그래프만 강조하고자 할 때 효과가 뛰어남
- Seaborn의 FacetGrid와 함께 접목하여 사용

그래프의 테두리(spine) 강조하기 (ch4-4.py)

```
df = pd.read_csv('./datasets/CO2_emissions/CO2_emissions.csv')

fig, ax = plt.subplots()
sns.boxplot(
    x='Cylinders', y='CO2 Emissions(g/km)',
    data=df, ax=ax
)

spines = ['left','right','top','bottom']
for spine in spines:
    ax.spines[spine].set_color('blue')
    ax.spines[spine].set_linewidth(3)
```



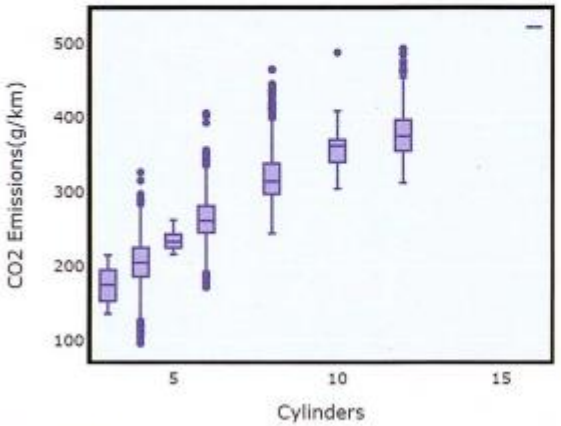
▲ 그림 123 Matplotlib ax의 spines 메서드를 이용한 그래프 테두리(spine) 강조

■ 그래프의 테두리 색깔과 선 굵기를 조절하여 그래프를 강조하는 법

- Plotly 라이브러리를 통해 그린 그래프에서도 테두리를 강조할 수 있음
- update_xaxes 및 update_yaxes 메서드를 이용

그래프의 테두리(spine) 강조하기 (ch4-4.py)

```
fig = px.box(df, x='Cylinders', y='CO2 Emissions(g/km)', width=500, height=400)
fig.update_xaxes(showline=True, linecolor='black', linewidth=3, mirror=True)
fig.update_yaxes(showline=True, linecolor='black', linewidth=3, mirror=True)
fig.show()
```



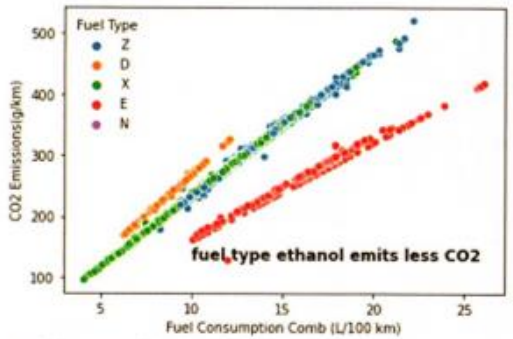
▲ 그림 124 Plotly 라이브러리로 그린 그래프의 테두리 선 굵기 변경

■ 그래프 안에 텍스트를 추가로 입력하여 나타내고자 하는 바를 보다 명확히 전달

- ax 내에 텍스트를 입력하는 방법은 text와 annotate 메서드 두 가지

그래프 내 텍스트 표현하기 (ch4-5.py)

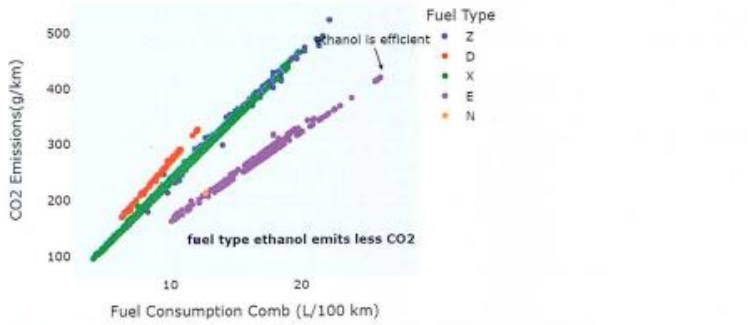
```
df = pd.read_csv('./datasets/CO2_emissions/CO2_emissions.csv')

fig, ax = plt.subplots()
sns.scatterplot(
    x='Fuel Consumption Comb (L/100 km)',
    y='CO2 Emissions(g/km)', data=df, ax=ax, hue='Fuel Type'
)
ax.text(
    x=10, y=130,
    s='fuel type ethanol emits less CO2',
    fontdict={'fontsize':12, 'weight':'bold'}
)
```



▲ 그림 125 text 메서드를 이용한 그래프 내 텍스트 삽입

그래프 내 텍스트 표현하기 (ch4-5.py)

```
fig = px.scatter(
    df, x='Fuel Consumption Comb (L/100 km)',
    y='CO2 Emissions(g/km)', width=500, height=400,
    color='Fuel Type'
)
fig.add_annotation(
    x=20, y=130, text='<b>fuel type ethanol emits less CO2</b>',
    showarrow=False
)
fig.add_annotation(
    x=0.9, xref='x domain', y=0.75, yref='y domain', text='ethanol is efficient',
    showarrow=True, arrowhead=2
)
fig.show()
```



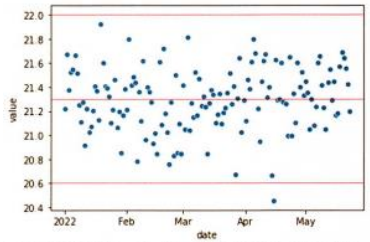
▲ 그림 128 Plotly 라이브러리에서 add_annotation 메서드를 통해 그래프 내 텍스트 추가하기

■ 그래프 내 수평선과 수직선을 그리는 방법

- 수평선 및 수직선은 그래프로 나타낸 데이터를 특정 상수와 상대적인 크기를 비교할 때 많이 사용
- 수평선은 axhline (ax horizontal line의 약자입니다) 메서드를 통해 그릴 수 있음
- 수직선을 그리는 메서드는 수평선을 그리는 axhline 메서드와 비슷하게 axvline을 사용함 (ax vertical line의 약자)
- Plotly 라이브러리에서 수평선 및 수직선을 그리고자 한다면 각각 add_hline과 add_vline 메서드를 이용할 수 있음

수평선과 수직선 그리기 (ch4-6.py)

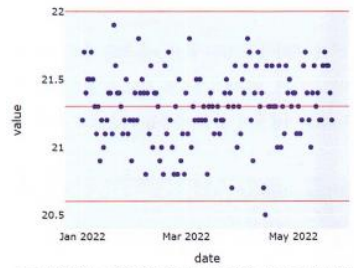
```
fig, ax = plt.subplots()
sns.scatterplot(x='date', y='value', data=df, ax=ax)
ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))

ax.axhline(df['lower_spec'].iloc[-1], color='red', linewidth=0.5)
ax.axhline(df['target'].iloc[-1], color='red', linewidth=0.5)
ax.axhline(df['upper_spec'].iloc[-1], color='red', linewidth=0.5)
```



▲ 그림 130 Matplotlib ax의 axhline 메서드를 이용한 수평선 그리기

수평선과 수직선 그리기 (ch4-6.py)

```
fig = px.scatter(df, x='date', y='value', width=500, height=400)
fig.add_hline(df['lower_spec'].iloc[-1], line_color='red', line_width=0.5)
fig.add_hline(df['target'].iloc[-1], line_color='red', line_width=0.5)
fig.add_hline(df['upper_spec'].iloc[-1], line_color='red', line_width=0.5)
fig.show()
```



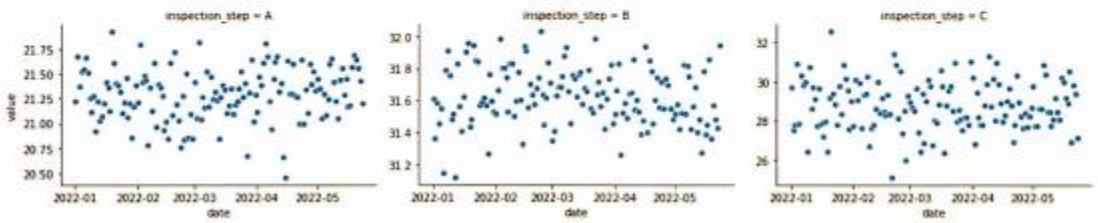
▲ 그림 131 Plotly 라이브러리로 그린 그래프에 add_hline 메서드를 통해 수평선 추가하기

■ **Seaborn FacetGrid를 이용하여 "A", "B", "C"각 검수 공정에 대한 계측 값 시계열 그래프를 열로 나눠서 그려보고, 각 ax별로 matplotlib의 여러 메서드를 mapping하는 법**

- 3가지의 검사 공정에 대해 총 3개의 그래프가 필요하므로, seaborn의 FacetGrid를 이용하여 검수 공정을 기준으로 열을 나눔

FacetGrid로 나눈 각 ax별 mapping으로 그래프 세부 튜닝하기 (ch4-7.py)

```
df = pd.read_csv('./datasets/product_inspection/product_inspection.csv')
df['date'] = pd.to_datetime(df['date'])

g = sns.FacetGrid(df, sharex=False, sharey=False, col='inspection_step', aspect=1.6)
g.map_dataframe(sns.scatterplot, x='date', y='value')
```



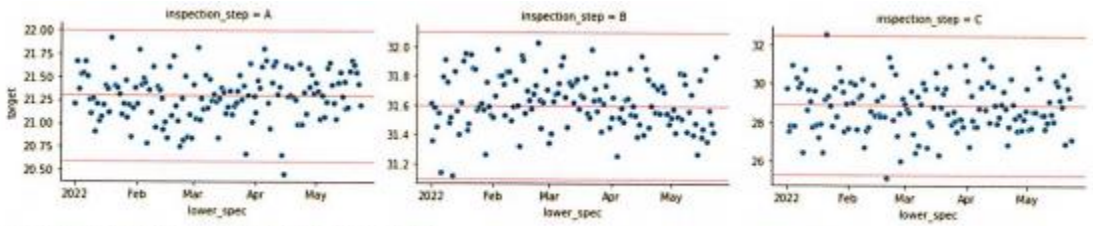
▲ 그림 132 seaborn FacetGrid로 그린 product_inspection 데이터셋 계측값의 시계열 그래프

■ **Seaborn FacetGrid를 이용하여 "A", "B", "C"각 검수 공정에 대한 계측 값 시계열 그래프를 열로 나눠서 그려보고, 각 ax별로 matplotlib의 여러 메서드를 mapping하는 법**

- 검수 공정 “A", "B”, “C"는 각각 관리하는 spec의 값이 다음
- 따라서 각 ax(검수 공정)마다 그에 적절한 spec 선을 그어야 함
- FacetGrid의 map 메서드를 이용하여 각 그래프에 spec을 나타내는 수평선을 그림

FacetGrid로 나눈 각 ax별 mapping으로 그래프 세부 튜닝하기 (ch4-7.py)

```
import matplotlib as mpl
def custom(lower_spec, target, upper_spec, **kws):
    ax = plt.gca()

    ax.axhline(lower_spec.iloc[-1], color='red', linewidth=0.5)
    ax.axhline(target.iloc[-1], color='red', linewidth=0.5)
    ax.axhline(upper_spec.iloc[-1], color='red', linewidth=0.5)

    ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))

g.map(custom, 'lower_spec', 'target', 'upper_spec')
```

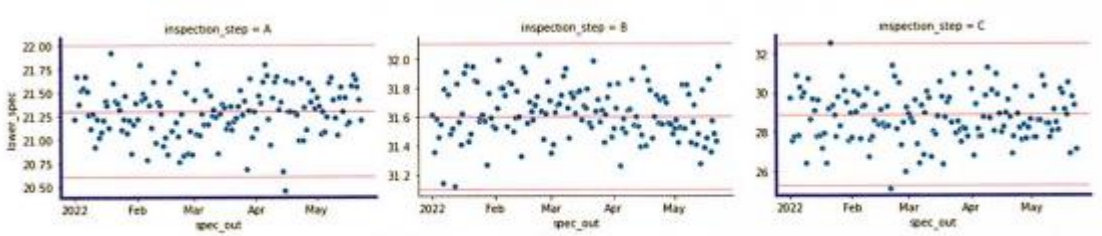


▲ 그림 133 axhline 메서드를 이용한 수평선 그리기

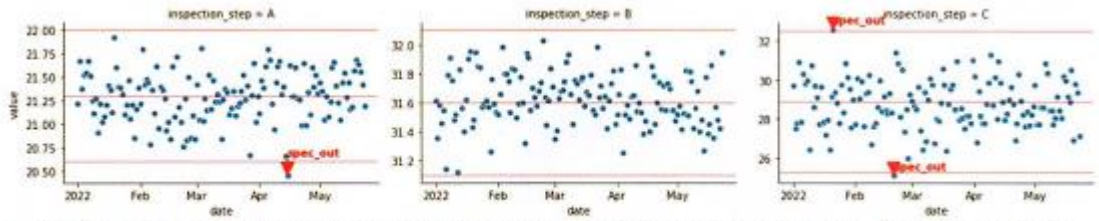
Ⅷ FacetGrid에서 특정 조건에 해당하는 ax 강조하기

■ **특정 조건에 상응하는 그래프의 테두리를 강조하여 표시하면 수많은 그래프에서 원하는 조건에 해당하는 그래프만 눈에 띄게 표현**

- 특정 조건에 해당하는 data가 있을 경우 해당 ax 그래프의 테두리를 강조
- 특정 조건에 해당하는 data에 annotate 메서드를 이용하여 텍스트 표시를 하는 방법



▲ 그림 137 Seaborn FacetGrid의 특정 조건을 만족하는 ax만 테두리 강조

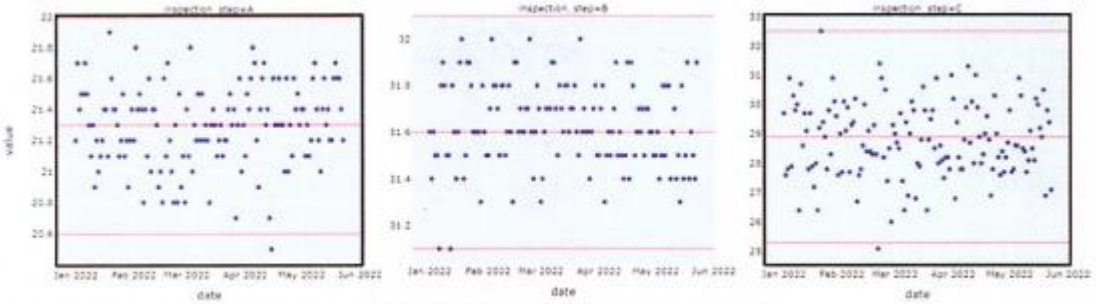


▲ 그림 138 Seaborn FacetGrid의 특정 조건을 만족하는 data의 scatter 옆에 annotate 메서드를 이용한 텍스트와 화살표 표시

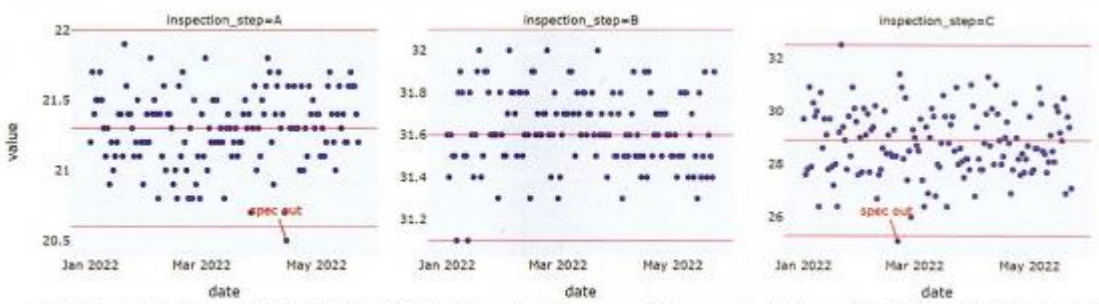
Ⅷ FacetGrid에서 특정 조건에 해당하는 ax 강조하기

■ **특정 조건에 상응하는 그래프의 테두리를 강조하여 표시하면 수많은 그래프에서 원하는 조건에 해당하는 그래프만 눈에 띄게 표현**

- Plotly 라이브러리를 사용하여 이상점이 있는 검수 공정에 대한 그래프에 테두리 강조효과
- add_annotation 메서드를 활용하여 Plotly에서 spec out된 scatter에 annotation을 추가



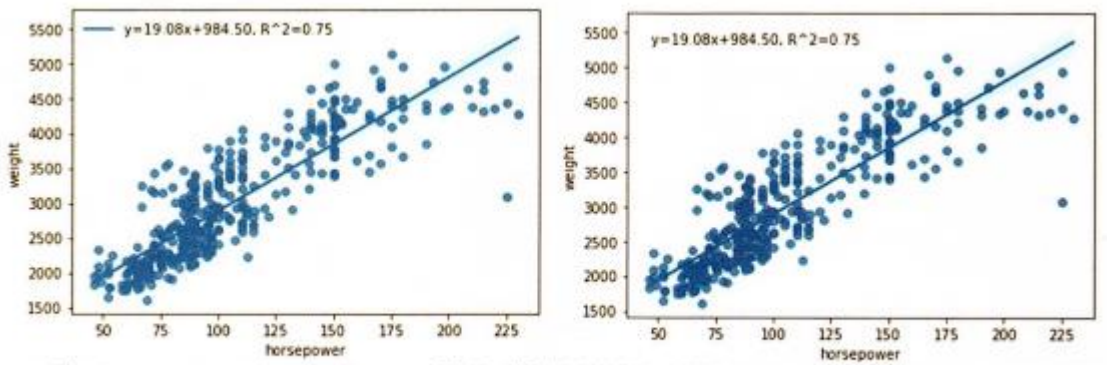
▲ 그림 139 Plotly facet을 통해 그린 subplot에서 특정 조건을 만족하는 그래프만 테두리 강조



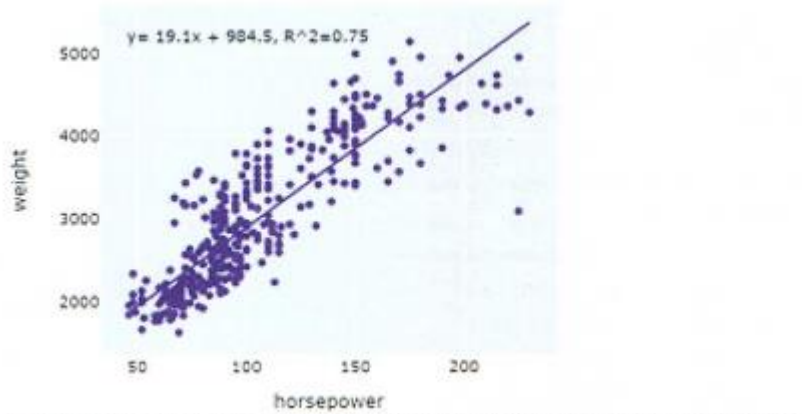
▲ 그림 140 Plotly로 그린 그래프에서 특정 조건을 만족하는 data의 scatter 옆에 annotate 메서드를 이용한 텍스트와 화살표 표시

■ scipy 라이브러리를 함께 이용

- scipy 라이브러리를 이용하여 해당 변수들에 대한 회귀선의 식과 결정계수를 구하고, 그래프 위에 표시
- scipy 라이브러리의 stats에서 linregress 함수를 임포트함
- linregress 함수에 선형회귀를 구할 x, y 두 가지 변수에 대한 데이터를 각각 인자로 전달하면 순서대로 회귀식의 기울기, y절편, 피어슨 상관계수 (제곱하면 결정계수) p-값, 기울기의 표준편차를 반환함



▲ 그림 142 seaborn regplot과 scipy의 linregress 함수를 이용하여 회귀식과 결정계수를 그래프 위에 표시, legend를 이용한 표현법 (왼쪽), text 메서드를 이용한 표현법 (오른쪽)



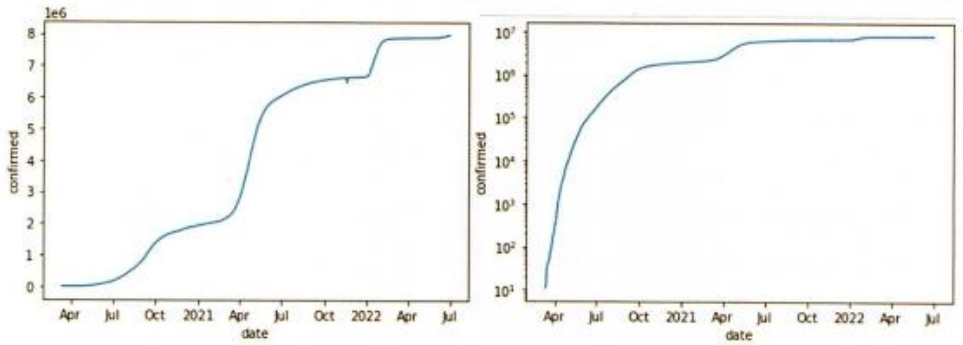
▲ 그림 143 Plotly express의 get_trendline_results 함수를 통해 구한 회귀식의 그래프 위 표현

■ 그래프의 축 log 형식으로 변환

- 데이터의 값이 특정 변수에 따라 기하급수적으로 변하는 데이터셋은 그래프로 표현했을 때 데이터의 변동 폭이 크기 때문에 값이 변하는 것을 세부적으로 확인하기 어려울 때가 있음
- 그래프의 축을 로그 스케일로 변경하면 그래프의 시인성을 향상시킬 수 음

그래프의 축 log 형식으로 변환하기 (ch4-10.py)

```
df = pd.read_csv('./datasets/Covid19-India/Covid19-India.csv')
df['date'] = pd.to_datetime(df['date'])
df = df.loc[df.region == 'Maharashtra']

fig, ax = plt.subplots()
sns.lineplot(x='date', y='confirmed', data=df, ax=ax)
ax.xaxis.set_major_formatter(mpl.dates.ConciseDateFormatter(ax.xaxis.get_major_locator()))
```



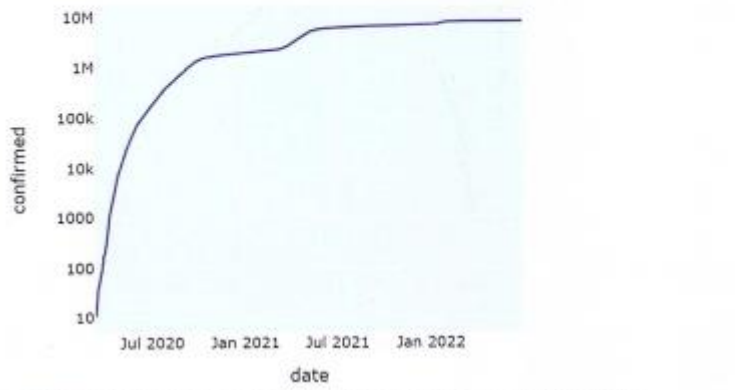
▲ 그림 144 seaborn lineplot을 이용해 그린 Covid19-India 데이터셋의 "Maharashtra" 주에 대한 코로나19 바이러스 확진 수 시계열 그래프. y축 선형 스케일(왼쪽)과 로그 스케일(오른쪽)

■ 그래프의 축 log 형식으로 변환

- Plotly express로 그린 그래프에서 특정 축을 로그 스케일로 변환하는 방법은 Matplotlib 라이브러리보다 훨씬 간단함
- Plotly express의 여러 그래프를 그리는 함수들은 대부분 "log_x"나 “log_y"처럼 특정 축을 로그 스케일로 변환시켜주는 인자를 기본으로 제공

그래프의 축 log 형식으로 변환하기 (ch4-10.py)

```
fig = px.line(
    df, x='date', y='confirmed', width=500, height=400, log_y=True
)
fig.show()
```



▲ 그림 145 Plotly express lineplot으로 y축을 로그 스케일로 나타낸 그래프

A Seaborn color palette 및 Plotly color의 활용

■ seaborn 라이브러리에서 color palette를 활용하는 법

- scatter의 색깔은 palette 인자를 이용하여 변경할 수 있음
- Seaborn 라이브러리는 많이 사용될 법한 여러 가지 형태의 color palette를 기본적으로 제공
- seaborn 라이브러리의 diverging_palette 인자를 이용하여 위 그림의 오른쪽과 유사한 color palette를 생성할 수 있음
- matplotlib 공식 홈페이지에서 가져온 color string들의 예시

X11/CSS4	xkcd		X11/CSS4	xkcd		X11/CSS4	xkcd	
#00FFFF	#13EAC9	aqua	#008000	#15B01A	green	#DDA0DD	#580F41	plum
#7FFFD4	#04D8B2	aquamarine	#808080	#929591	grey	#800080	#7E1E9C	purple
#F0FFFF	#069AF3	azure	#4B0082	#380282	indigo	#FF0000	#E50000	red
#F5F5DC	#E6DAA6	beige	#FFFFF0	#FFFFCB	ivory	#FA8072	#FF796C	salmon
#000000	#000000	black	#F0E68C	#AAA662	khaki	#A0522D	#A9561E	sienna
#0000FF	#0343DF	blue	#E6E6FA	#C79FEF	lavender	#C0C0C0	#C5C9C7	silver
#A52A2A	#653700	brown	#ADD8E6	#7BC8F6	lightblue	#D2B48C	#D1B26F	tan
#7FFF00	#C1F80A	chartreuse	#90EE90	#76FF7B	lightgreen	#008080	#029386	teal
#D2691E	#3D1C02	chocolate	#00FF00	#AAFF32	lime	#FF6347	#EF4026	tomato
#FF7F50	#FC5A50	coral	#FF00FF	#C20078	magenta	#40E0D0	#06C2AC	turquoise
#DC143C	#8C000F	crimson	#800000	#650021	maroon	#EE82EE	#9A0EEA	violet
#00FFFF	#00FFFF	cyan	#000080	#01153E	navy	#F5DEB3	#FBDD7E	wheat
#00008B	#030764	darkblue	#808000	#6E750E	olive	#FFFFFF	#FFFFFF	white
#006400	#054907	darkgreen	#FFA500	#F97306	orange	#FFFF00	#FFFF14	yellow
#FF00FF	#ED0DD9	fuchsia	#FF4500	#FE420F	orangered	#9ACD32	#BBF90F	yellowgreen
#FFD700	#DBB40C	gold	#DA70D6	#C875C4	orchid			
#DAA520	#FAC205	goldenrod	#FFC0CB	#FF81C0	pink			

▲ 그림 150 matplotlib 라이브러리에서 사용할 수 있는 다양한 문자열 색상 코드[8]

▲ 그림 156 Plotly에서 제공하는 연속형 데이터용 built-in color map에 대한 예시