

PBL3

Data Visualization



1조
신민경 이정인 전성원

Contents

- ❑ Visualization tools
- ❑ Dataset
- ❑ Data structure
- ❑ Analysis Issues
- ❑ Data Preprocessing
- ❑ Visualization

Visualization tools

1



Matplotlib을 기반한 다양한 색상 테마와
통계용 차트 등의 기능을 추가한 시각화 패키지

2



반응형, 오픈소스 브라우저 기반 시각화 라이브러리
30개 종류의 차트를 가지고 있고 3D그래프도 지원

Dataset

Novel Corona Virus 2019 Dataset

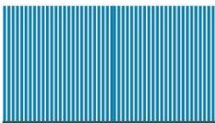
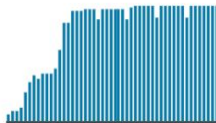



Day level information on covid-19 affected cases

일별 Covid19 누적 확진, 사망, 완치자 수

[https://www.kaggle.com/sudalairaikumar/novel-corona-virus-2019-dataset?
select=covid_19_data.csv](https://www.kaggle.com/sudalairaikumar/novel-corona-virus-2019-dataset?select=covid_19_data.csv)



Data Structure

#	날짜	도시	나라	업데이트	확진자	사망
# SNo Serial Number	📅 ObservationDate Observation date in mm/dd/yyyy	📍 Province/State Province or State	📍 Country/Region Country or region	📅 Last Update Last update date time in UTC	# Confirmed Cumulative number of confirmed cases	# Deaths Cumulative number of deaths cases
 1 306k	 22Jan20 29May21	[null] 25% Unknown 1% Other (224206) 73%	Russia 10% US 9% Other (249438) 81%	 23Jan20 30May21	 -303k 5.86m	 -178 112k
1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0
2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0
3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0
4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0

Analysis Issues

□ 하나의 데이터를 다양한 시각화 툴을 사용하여 분석

- 공통적으로 제공하는 그래프: 차이점 & 장단점
- 각 툴마다 독자적으로 제공하는 그래프: 장점

□ 데이터 구조에 따른 비교 분석

Wide Format

	nation	gold	silver	bronze
0	South Korea	24	13	11
1	China	10	15	8
2	Canada	9	12	12

Long Format

	nation	medal	count
0	South Korea	gold	24
1	China	gold	10
2	Canada	gold	9
3	South Korea	silver	13

Data Preprocessing

Wide Format

```
df.head()
```

Out[67]:

	SNo	Date	State	Country	Last Update	Confirmed	Deaths	Recovered	Active
0	1	2020-01-22	Anhui	Mainland China	1/22/2020 17:00	1	0	0	1
22	23	2020-01-22	Qinghai	Mainland China	1/22/2020 17:00	0	0	0	0
23	24	2020-01-22	Shaanxi	Mainland China	1/22/2020 17:00	0	0	0	0
24	25	2020-01-22	Shandong	Mainland China	1/22/2020 17:00	2	0	0	2

데이터 전처리 (Wide-Form)

```
df = df.rename(columns={'Country/Region': 'Country'})
```

```
df = df.rename(columns={'ObservationDate': 'Date'})
```

```
df['Province/State'] = df['Province/State'].fillna("Unknown")
```

```
df = df.rename(columns = {"Province/State": "State"})
```

```
df = df.rename(columns = {"Country/Region": "Country"})
```

```
df[['Confirmed', 'Deaths', 'Recovered']] = df[['Confirmed', 'Deaths', 'Recovered']].astype(int)
```

```
df["Date"] = pd.to_datetime(df["Date"], format="%m/%d/%Y")
```

```
df = df.sort_values("Date")
```

#Active 칼럼 추가

```
df['Active'] = df['Confirmed']-df['Deaths']-df['Recovered']
```

```
df.head()
```

Data Preprocessing

Long Format

	SNo	Date	State	Country	Last Update	Category	Count
0	1	2020-01-22	Anhui	Mainland China	1/22/2020 17:00	Confirmed	1
1	23	2020-01-22	Qinghai	Mainland China	1/22/2020 17:00	Confirmed	0
2	24	2020-01-22	Shaanxi	Mainland China	1/22/2020 17:00	Confirmed	0
3	25	2020-01-22	Shandong	Mainland China	1/22/2020 17:00	Confirmed	2
4	26	2020-01-22	Shanghai	Mainland China	1/22/2020 17:00	Confirmed	9

데이터 전처리 (Long-Form)

```
long_df = pd.melt(df, id_vars=['SNo', 'Date', 'State',  
                             'Country', 'Last Update'],  
                  value_vars=['Confirmed', 'Deaths',  
                              'Recovered', 'Active'],  
                  var_name='Category', value_name='Count')  
long_df.head()
```


Data Preprocessing

Wide

						Confirmed			
SNo	Date	State	Country	Last Update		Deaths	Recovered	Active	
0	1	2020-01-22	Anhui	Mainland China	1/22/2020 17:00	1	0	0	1

`pandas.melt(dataframe, id_vars, value_vars, var_name, value_name)`

Long

SNo	Date	State	Country	Last Update	Category	Count	
0	1	2020-01-22	Anhui	Mainland China	1/22/2020 17:00	Confirmed	1

Visualization: table chart

전세계 covid-19 통계 시각화

Total Covid-19

Total Confirmed	Total Deaths	Total Recovered
169951560	3533619	107140669

```
# 전세계 covid-19 통계 시각화
total_c = recent_df.groupby("Country")["Confirmed"].sum()
total_d = recent_df.groupby("Country")["Deaths"].sum()
total_r = recent_df.groupby("Country")["Recovered"].sum()
```

```
fig = go.Figure(data = [go.Table(
    header = dict(
        values = ['<b>Total Confirmed</b>', '<b>Total Deaths</b>', '<b>Total Recovered</b>'],
        line_color='black',
        fill_color= 'DarkRed',
        align='center',
        font=dict(color='white', size=12)
    ),
    cells = dict(
        values = [sum(total_c), sum(total_d), sum(total_r)],
        line_color='darkslategray',
        fill_color= 'white',
        font = dict(color = 'darkslategray', size = 11)
    )
)])

fig.update_layout(title = 'Total Covid-19 ',
    title_x = 0.5,
    title_font = dict(size = 16, color = 'DarkRed'))

fig.show()
```

Visualization: Bar charts

□ 국가별 확진자/사망자/완치자 시각화: Plotly의 bar 활용

Long Format

```
# 나라별 확진자, 사망자, 완치자 누적 통계 Long-Form
bar1_long = pd.DataFrame(recent_long[recent_long["Category"]!="Active"]\
                        .groupby(['Country', 'Category'])["Count"].sum().reset_index())
bar1_long.head()
```

	Country	Category	Count
0	Afghanistan	Confirmed	70111
1	Afghanistan	Deaths	2899
2	Afghanistan	Recovered	57281
3	Albania	Confirmed	132297
4	Albania	Deaths	2449

```
fig = px.bar(bar1_long, x="Country", y="Count", color="Category",\
             title="국가별 확진자, 사망자, 완치자 통계")
fig.show()
```

Wide format: y축 변수를 직접 지정해줘야 한다.
즉, 범주가 많은 경우 Long format이 더 편하다.

Wide Format

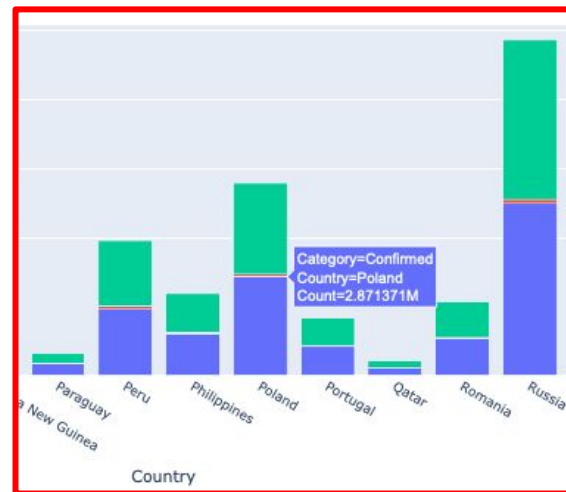
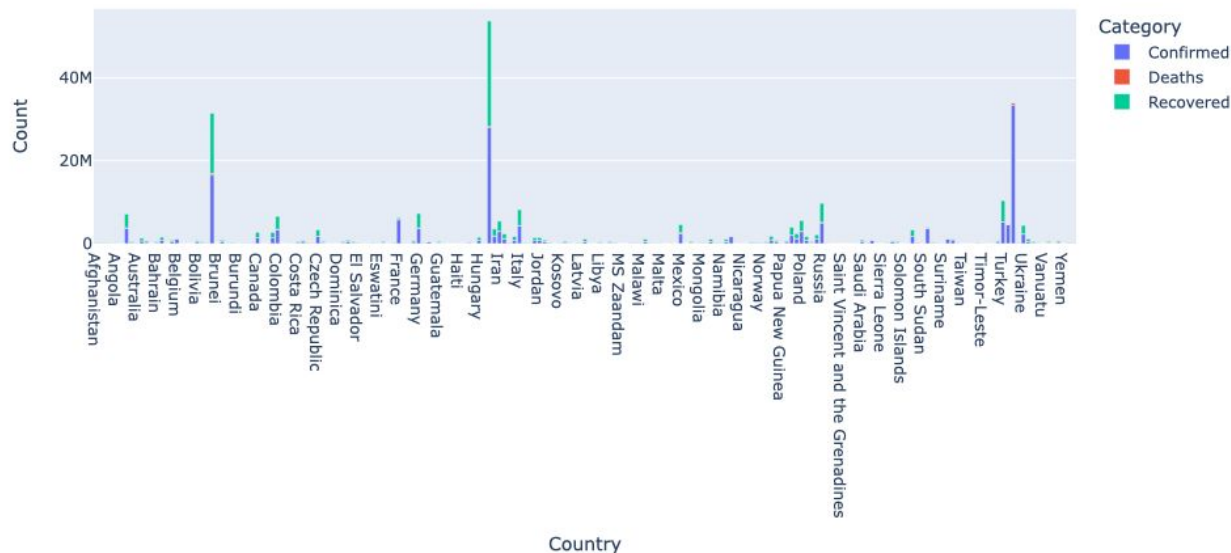
```
# 나라별 확진자, 사망자, 완치자 함께 비율 Wide-Form
bar1_wide = pd.DataFrame(recent_df.groupby("Country")\
                        [["Confirmed", "Deaths", "Recovered"]].sum().reset_index())
bar1_wide.head()
```

	Country	Confirmed	Deaths	Recovered
0	Afghanistan	70111	2899	57281
1	Albania	132297	2449	129215
2	Algeria	128456	3460	89419
3	Andorra	13693	127	13416
4	Angola	34180	757	27646

```
fig = px.bar(bar1_wide, x="Country", y=["Confirmed", "Deaths", "Recovered"]\
             title="국가별 확진자, 사망자, 완치자 통계",\
             labels={"value": "Count", "variable": "Category"})
fig.show()
```

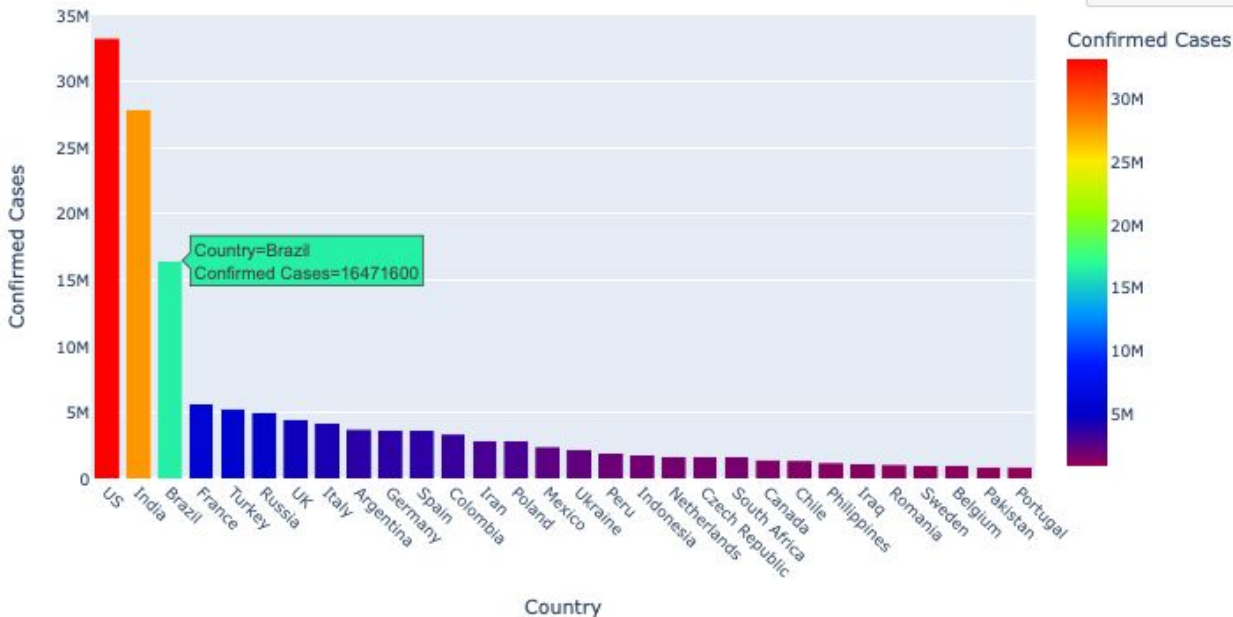
Visualization: Bar charts

□ 국가별 확진자/사망자/완치자 시각화: Plotly의 bar 활용



Visualization: Bar charts

□ 확진자 수 상위 30개 국가: Plotly의 bar 활용



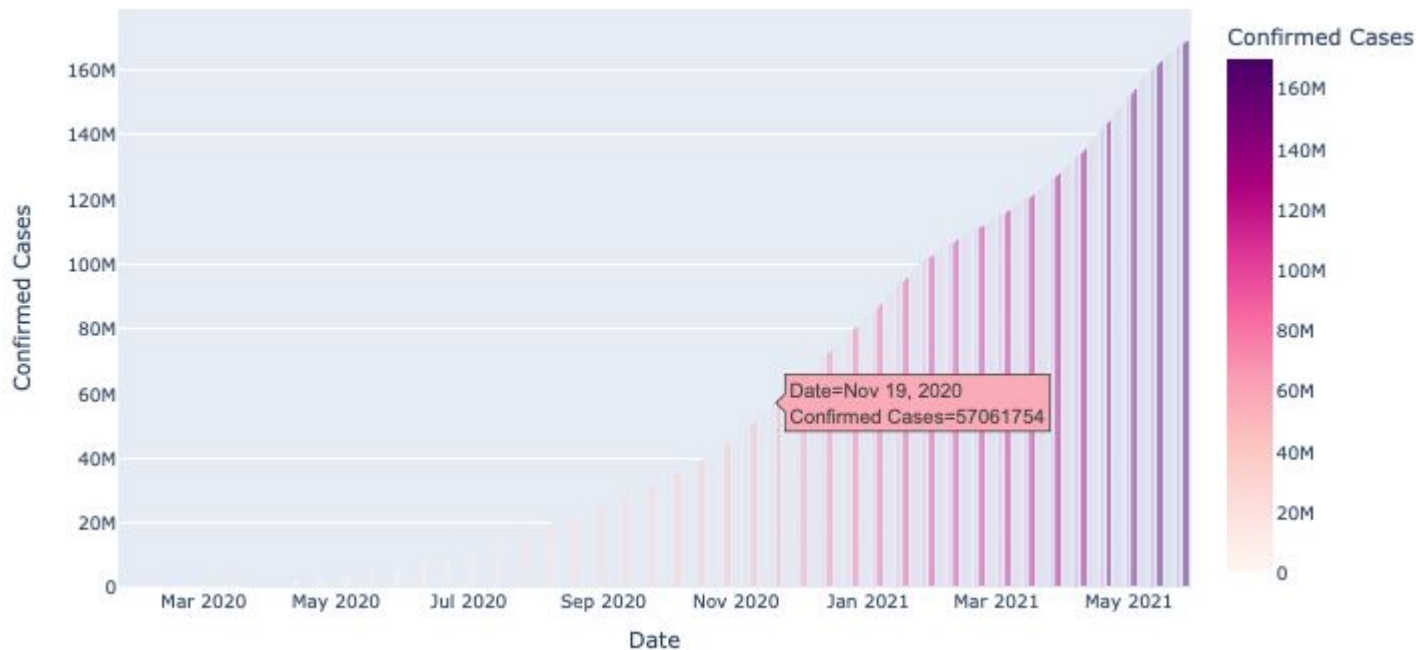
```
#Plotly bar- 상위30개 확진된 나라 분포 시각화
df1 = recent_df.groupby("Country")["Confirmed"].sum()
        .sort_values(ascending = False).reset_index().head(30)
fig = px.bar(df1,
            x = 'Country',
            y = 'Confirmed',
            color = 'Confirmed',
            color_continuous_scale = 'rainbow',
            labels = {"Confirmed": "Confirmed Cases"})

fig.update_layout(title = 'Top 30 Confirmed Cases of Country',
                  title_x = 0.5,
                  title_font = dict(size = 18, color = 'black'),
                  yaxis = dict(title = 'Confirmed Cases'),
                  xaxis = dict(tickangle = 45))

fig.show()
```

Visualization: Bar charts

□ 전세계의 일일 확진자 수 변화: Plotly의 bar 활용



```
#Plotly bar - 전세계 일일 확진자 변화
dftime = pd.DataFrame(df.groupby("Date")["Confirmed"].sum().reset_index())

fig = px.bar(dftime,
             x = 'Date',
             y = 'Confirmed',
             color = 'Confirmed',
             color_continuous_scale = 'rdpu',
             labels = {"Confirmed": "Confirmed Cases"})

fig.update_layout(title = 'World daily confirmed cases',
                  title_x = 0.5,
                  title_font = dict(size = 18, color = 'DarkRed'),
                  xaxis = dict(title = 'Date'),
                  yaxis = dict(title = 'Confirmed Cases'))

fig.show()
```

Visualization: Bar charts

□ 국가별 확진자/사망자/완치자 시각화: Seaborn의 bar 활용

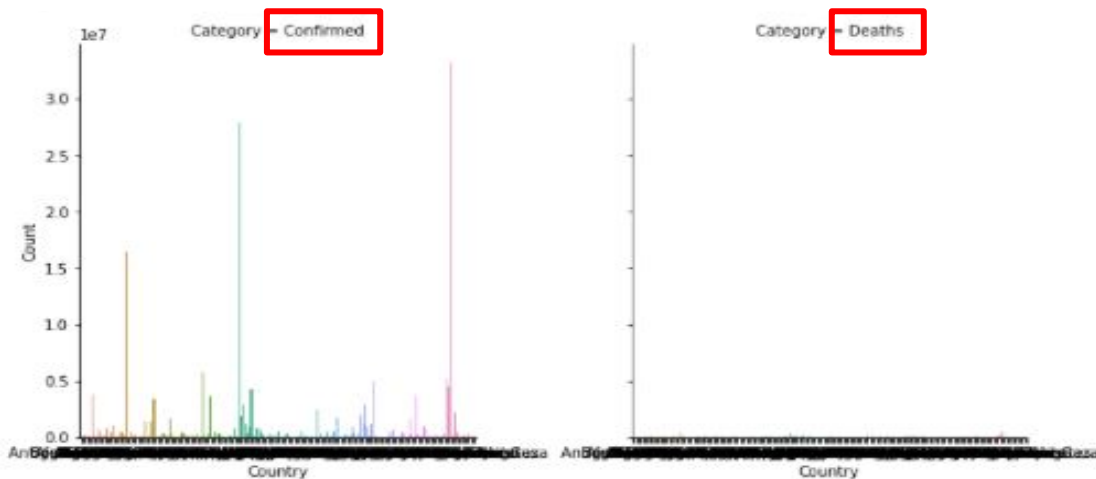


Wide format: y변수가 여러 개인 경우는 시각화 할 수 없다.

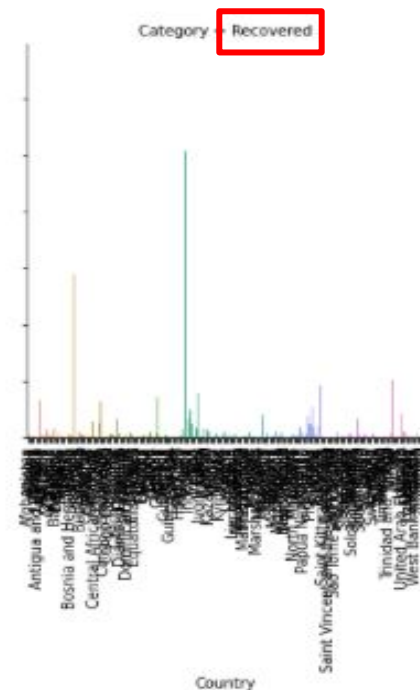
Long format: hue를 이용하여 범주별 그래프를 그리기 편하다.

Visualization: Bar charts

전세계의 일일 확진자 수 변화: Seaborn의 bar 활용

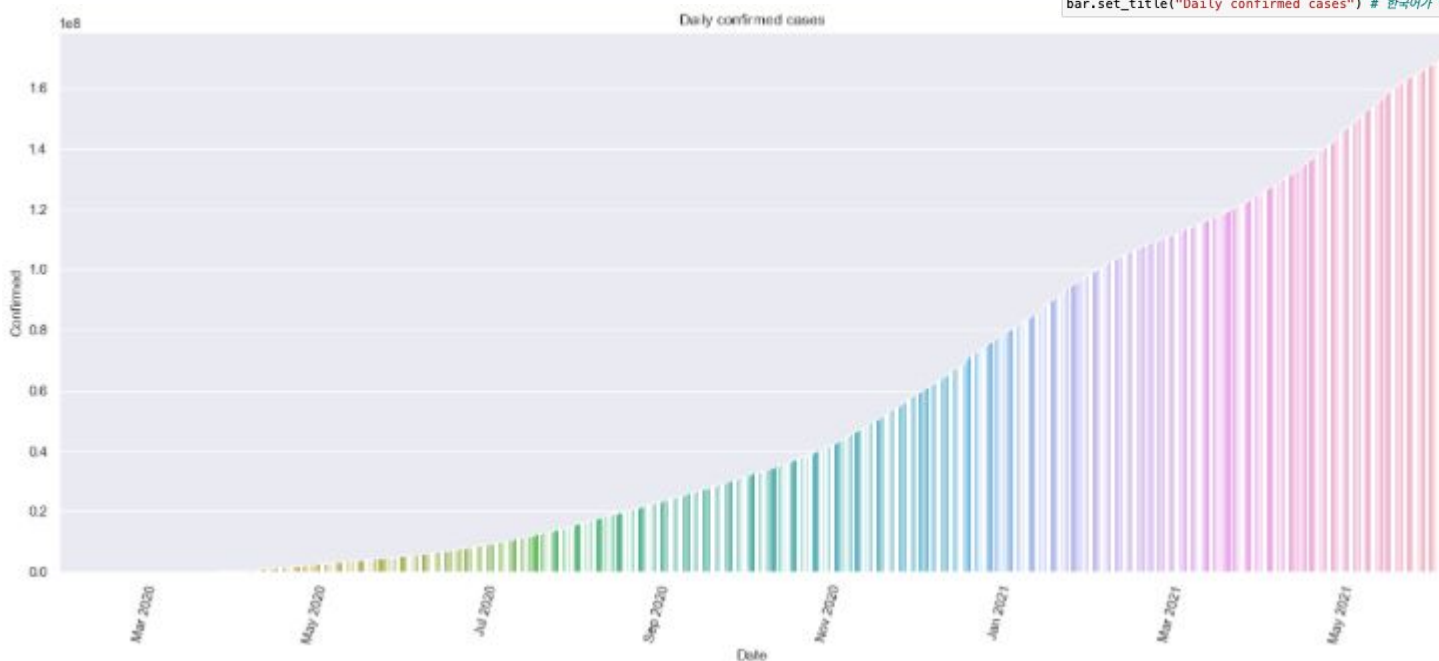


```
bar2 = sns.catplot(x="Country", y="Count", data=bar1_long, col="Category", kind="bar")
plt.xticks(rotation=90)
plt.title("Confirmed, Deaths, Recovered by Country")
plt.figure(figsize=(1000,500))
```



Visualization: Bar charts

□ 전세계의 일일 확진자 수 변화: Seaborn의 bar 활용



```
# Seaborn - 일일 확진자 변화
l = df.time["Date"].count()
plt.xticks(rotation=75)
bar = sns.barplot(data=df.time, x = 'Date', y = 'Confirmed')

sns.set(rc = {'figure.figsize':(15,8)})
bar.set_xticks(np.arange(l//16,l//8)) # x축 라벨 및 간격도 직접 조절해야 함 (안하면 완전 배극해서 못알아봄)
bar.set_xticklabels(['Mar 2020', 'May 2020', 'Jul 2020', 'Sep 2020', 'Nov 2020', 'Jan 2021', 'Mar 2021', 'May 2021'])
bar.set_title("Daily confirmed cases") # 한국어가 안됨
```

Visualization: Bar charts

❑ Plotly

x축의 길이가 너무 길거나 데이터가 촘촘한 경우 확대해서 보거나 일부분만 잘라서 볼 수 있기 때문에 편하다.

❑ Seaborn

장점)

Long format의 경우 문법도 간단하고 sub-plots 기능을 제공 -> 시각적으로 명료한 그래프 작성가능

단점)

그래프에 한글을 사용하기 복잡하다

x축의 간격이 너무 촘촘한 경우 글씨가 겹쳐서 안보인다

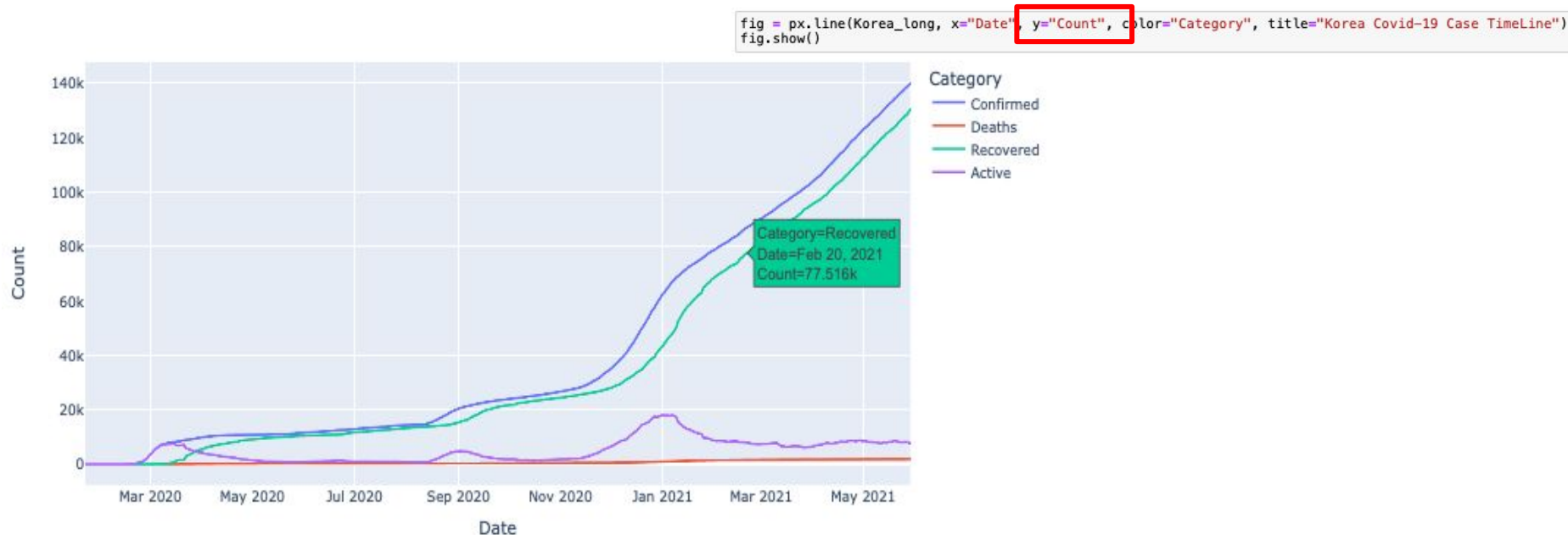
정확한 y 값을 보기 어렵다

Wide format은 시각화하기 어렵다

Visualization: Line charts

한국의 확진자/사망자/완치자/비완치자 타임라인: Plotly의 line 활용

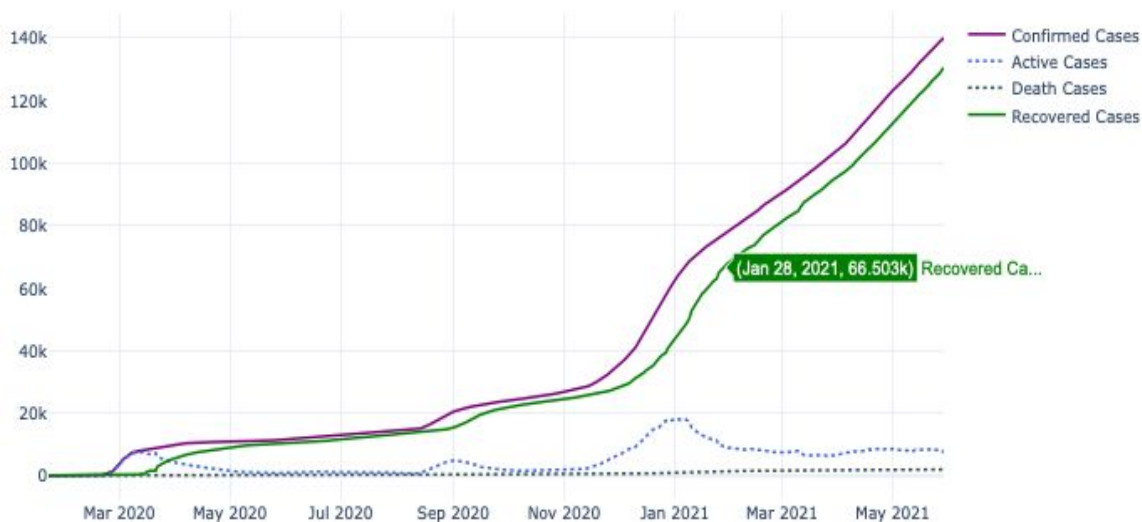
Long Format



Visualization: Line charts

한국의 확진자/사망자/완치자/비완치자 타임라인: Plotly의 line 활용

Wide Format



```
#Plotly- Scatter 사용
Data_Korea = df_Korea.groupby(["Date"])["Confirmed", "Deaths", "Recovered", "Active"]
               .sum().reset_index().sort_values("Date").reset_index()

fig = go.Figure()

fig.add_trace(go.Scatter(x=Data_Korea['Date'],
                        y=Data_Korea['Confirmed'],
                        mode='lines',
                        name='Confirmed Cases',
                        marker_color='purple'))

fig.add_trace(go.Scatter(x=Data_Korea['Date'],
                        y=Data_Korea['Active'],
                        mode='lines',
                        name='Active Cases',
                        marker_color='RoyalBlue',
                        line=dict(dash='dot'))))

fig.add_trace(go.Scatter(x=Data_Korea['Date'],
                        y=Data_Korea['Deaths'],
                        mode='lines',
                        name='Death Cases',
                        marker_color='DarkSlateGray',
                        line=dict(dash='dot'))))

fig.add_trace(go.Scatter(x=Data_Korea['Date'],
                        y=Data_Korea['Recovered'],
                        mode='lines',
                        name='Recovered Cases',
                        marker_color='green'))

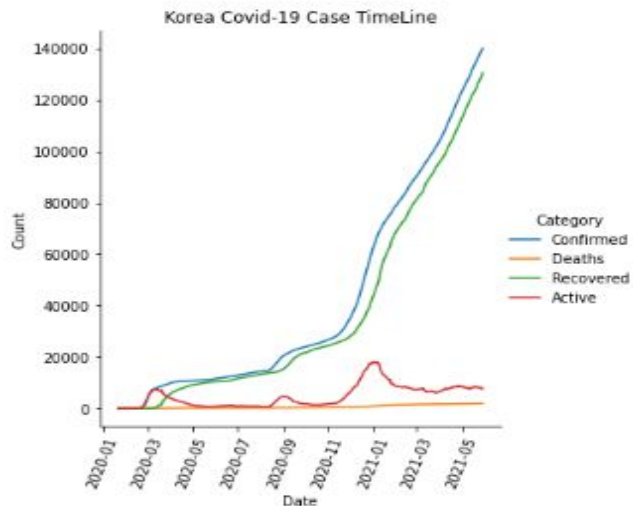
fig.update_layout(title='Korea Covid-19 Case TimeLine',
                  title_x=0.5,
                  title_font=dict(size=18, color='DarkRed'),
                  template='plotly_white')

fig.show()
```

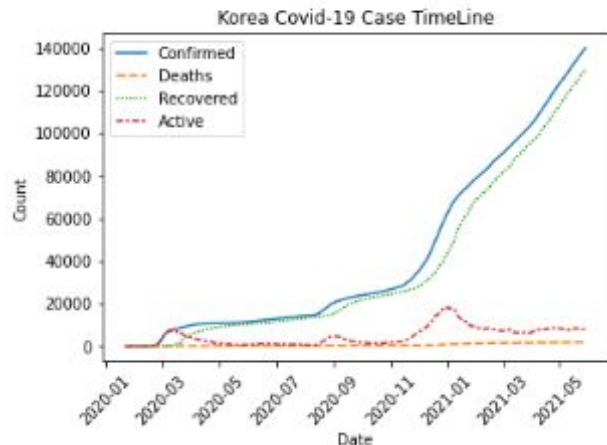
Wide format: 여러 그래프를 합치기 위해 y변수 개수만큼 함수를 적용해야 한다.
Long format: 범주를 자동으로 분류해줘서 편하다.

Visualization: Line charts

❑ 한국의 확진자/사망자/완치자/비완치자 타임라인: Seaborn의 line 활용



```
line1 = sns.relplot(x="Date", y="Count", hue="Category", kind="line", data=Korea_long)
plt.xticks(rotation=70)
line1.set(title='Korea Covid-19 Case TimeLine')
```



```
line_data = pd.DataFrame(df_Korea[["Confirmed", "Deaths", "Recovered", "Active", "Date"]]).set_index("Date")
line_data.head()
```

```
line2 = sns.lineplot(data=line_data)
plt.xticks(rotation=45)
plt.ylabel("Count")
line2.set_title("Korea Covid-19 Case TimeLine")
```

Wide format: 인덱스에 x축이 오도록 하고 남은 컬럼은 y축 범주가 된다.
따라서 데이터 구조를 바꿔야 하므로 조금 번거롭지만 그래프는 깔끔하게 그려졌다.

Visualization: Line charts

❑ Plotly

Long format 데이터: 그래프 그리기가 편하다.

Wide format 데이터: y변수 개수만큼 그래프를 일일이 그려야 해 불편하다.

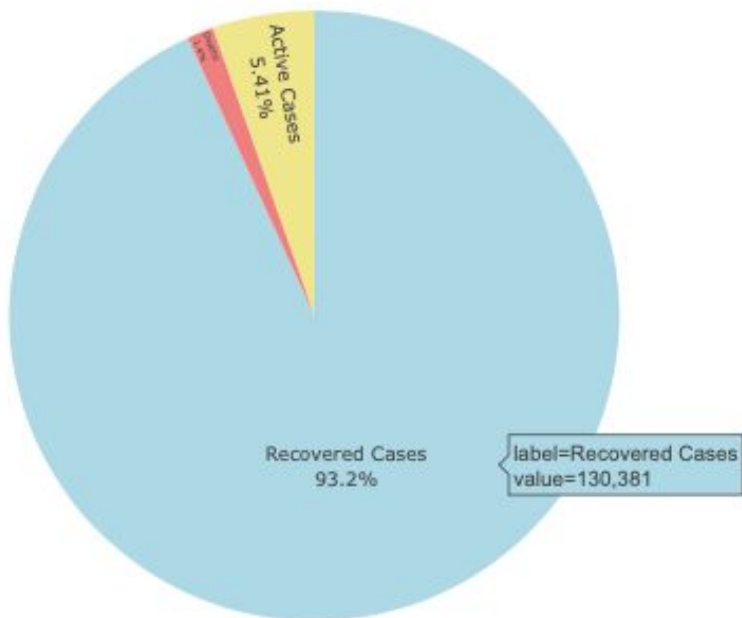
❑ Seaborn

장점) Plotly에 비해 Wide format 데이터를 시각화하는게 편했다.

단점) 정확한 x, y값을 볼 수 없어서 Plotly에 비해 아쉽다.

Visualization: Pie charts

❑ 한국의 누적 사망자/완치자/Active 비율: Plotly의 Pie 활용



Recovered Cases
Active Cases
Deaths

```
labels = ["Active Cases", "Recovered Cases", "Deaths"]

sumactive= int(recent_Korea['Active'])
sumrecovered = int(recent_Korea['Recovered'])
sumdeaths = int(recent_Korea['Deaths'])

fig = px.pie(recent_Korea,
             values = [sumactive, sumrecovered, sumdeaths],
             names = labels,
             color_discrete_sequence = ['lightblue', 'khaki', 'lightcoral'])

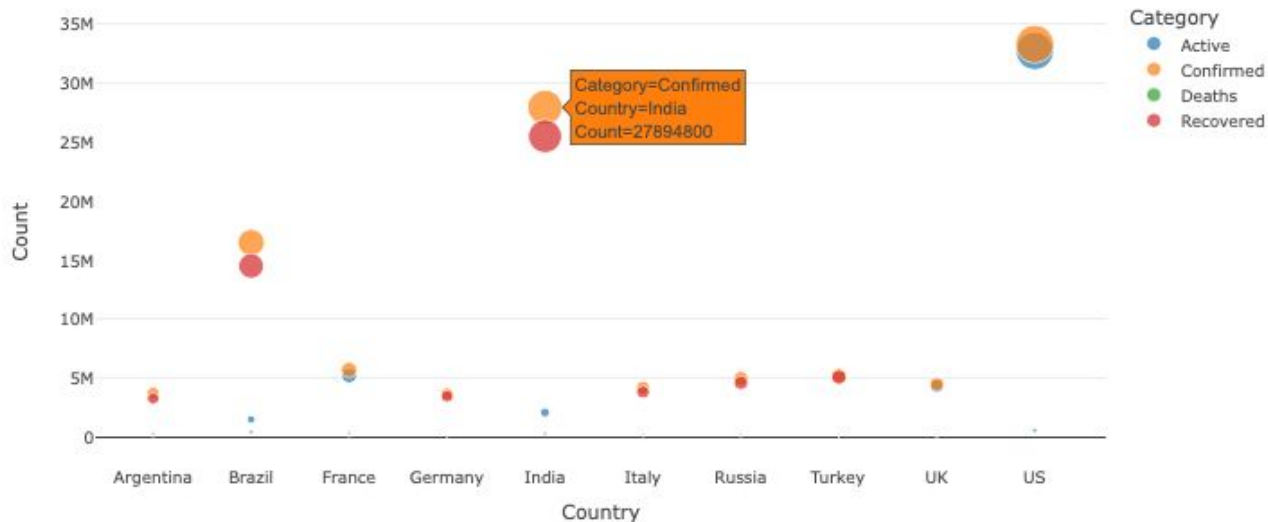
fig.update_traces(textposition = 'inside', textinfo = 'percent+label')
fig.update_layout(title = 'Korea Covid-19 Various Cases Chart ',
                  title_x = 0.45,
                  title_font= dict(size = 18, color = 'black' ))

fig.show()
```

Visualization: scatter charts

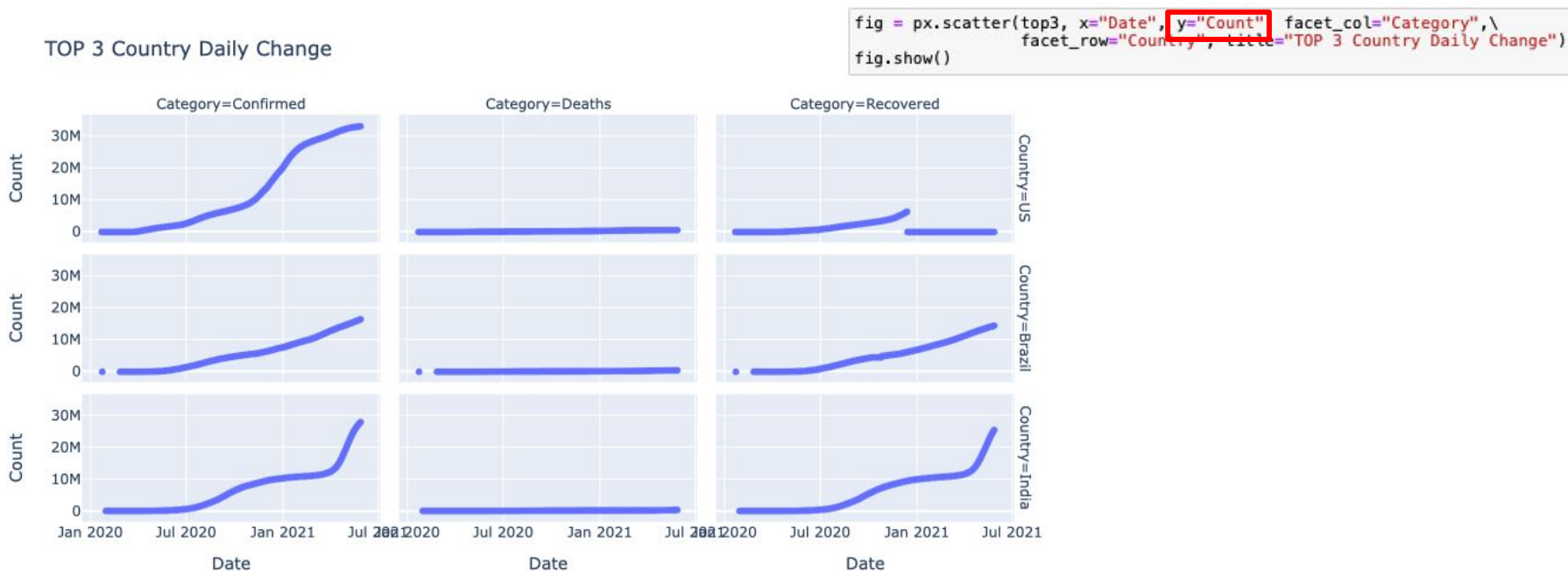
□ 누적 확진자 Top10 국가: Plotly의 Scatter 활용

```
#Plotly, Scatter사용
fig = px.scatter(top10, x="Country", y="Count", color="Category", \
                 template="xgridoff", size="Count", title='TOP 10 Covid-19 Country')
fig.show()
```



Visualization: scatter charts

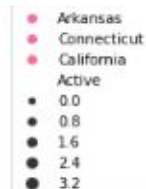
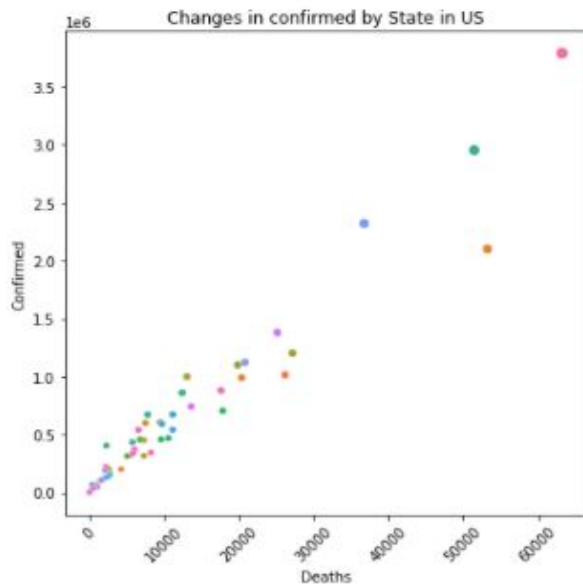
❑ 누적 확진자 Top3 국가의 하루 변화량: Plotly의 Scatter 활용



Visualization: scatter charts

미국 주별 확진자, 사망자 수: Seaborn의 Scatter 활용

```
f, ax = plt.subplots(figsize=(6.5, 6.5))
tmp = sns.scatterplot(x="Deaths", y="Confirmed", hue="State", size="Active", data=us_data)
plt.xticks(rotation=45)
ax.legend(bbox_to_anchor=(1.1, 1.05), title="State") # legend 위치 조정
tmp.set_title("Changes in confirmed by State in US")
```



Visualization: scatter charts

❑ Scatter charts

원의 크기를 컬럼 값으로 설정할 수 있고, 범주를 색으로 구분하여 한 그래프에서 표현할 수 있다.

Wide format: 원의 크기를 다른 범주의 값으로 표현하기 좋고, x,y축 변수를 다르게 조합하면서 x,y 변수의 상관관계를 분석할 수 있다.

Long format: 범주 별로 색을 구분하는데 좋고, 2개의 범주를 가지는 데이터를 범주에 따라 row, col로 그래프를 분할해서 볼 수 있다.

❑ Plotly

scatter matrix를 지원해서 변수 사이의 상관관계를 볼 수 있다.

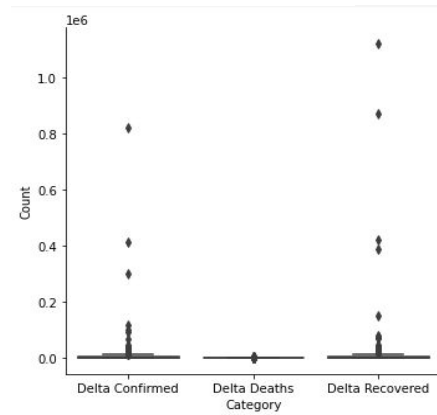
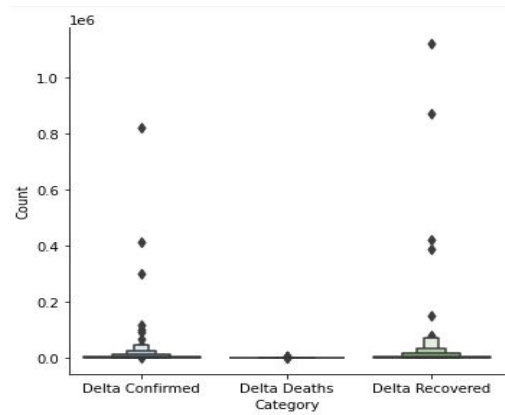
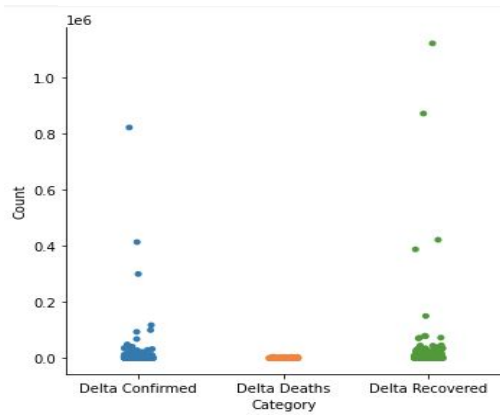
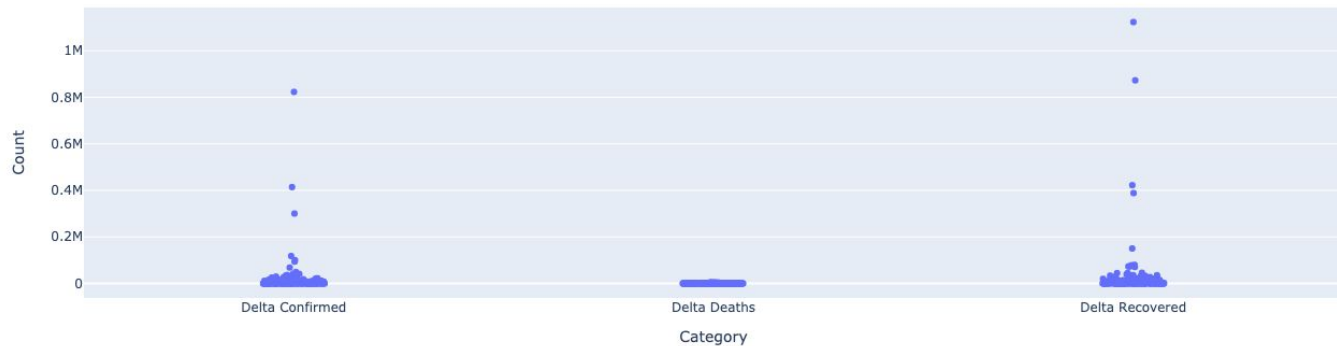
❑ Seaborn

Wide format: y축 변수를 여러 개 지정할 수 없다.

Long format: hue옵션으로 y값에 대한 범례들을 지정할 수 있다.

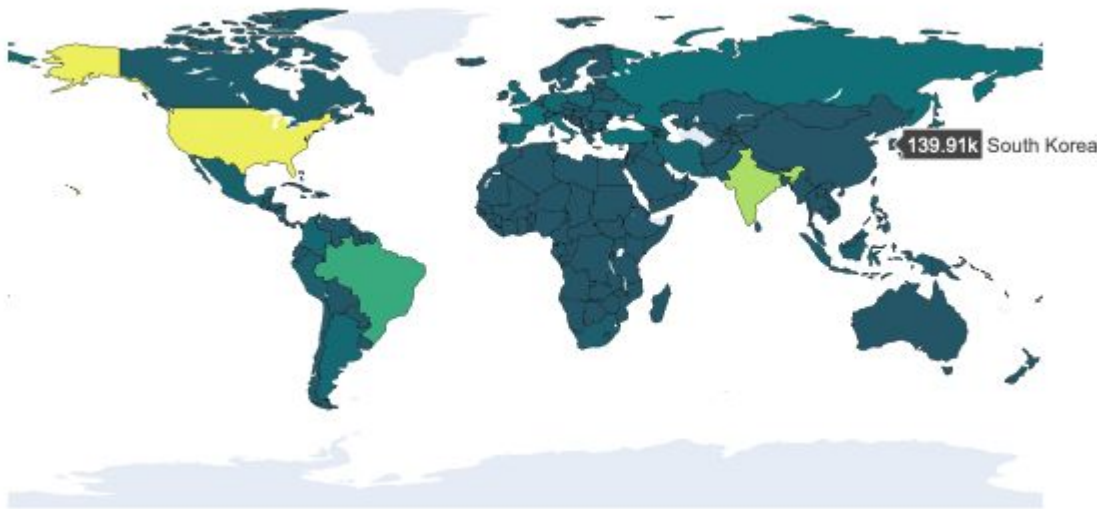
단점) 범례의 종류가 많은 경우 legend가 그래프를 가려서 따로 위치를 지정해줘야 하기 때문에 불편하다.

Visualization: box chart



Visualization: Map charts

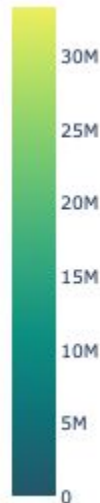
□ 국가별 누적 확진자 수 분포: Plotly의 Map 활용



```
# <Plotly - Map : 최근 확진자수 시각화>
fig = go.Figure(data=go.Choropleth(
    locations = df_countries['Country'], # 나라 이름 column
    locationmode = 'country names', # locationmode = worldwide
    z = df_countries['Confirmed'], # 확진자 수 column
    colorscale = 'aggrnyl',
    marker_line_color = 'black',
    marker_line_width = 0.4,
))

import datetime
date = df.sort_values('Date', ascending = False).reset_index()['Date'][0]

fig.update_layout(
    title_text = 'World-Wide Confirmed Cases Distribution Map ',
    title_x = 0.5,
    geo=dict(
        showframe = False,
        showcoastlines = False,
        projection_type = 'equirectangular'
    )
)
```



Thank You

1조의 발표였습니다. 감사합니다.