Author: Smridhi Mangla

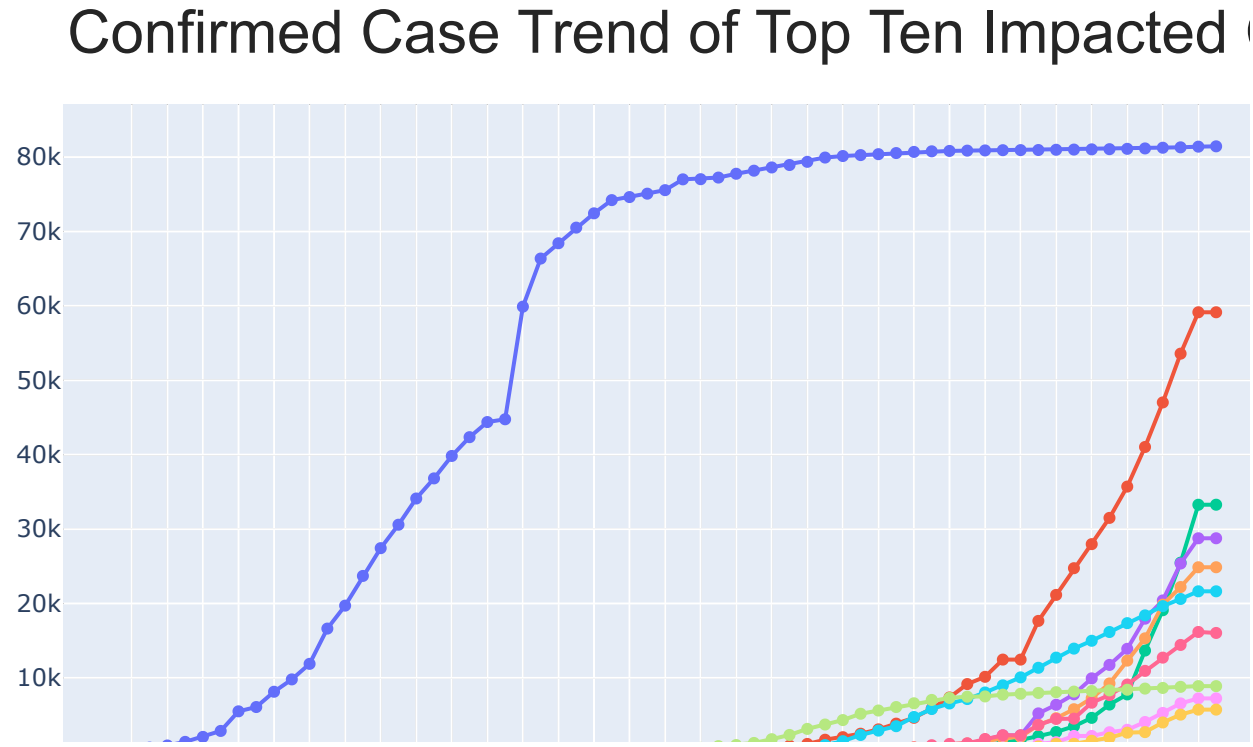
Line Plot

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import warnings
warnings.filterwarnings("ignore")


#download confirmed cases data from JHU dashboard
url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/archived_data/archive
df_confirmed = pd.read_csv(url)


#download recovered cases data from JHU dashboard
url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/archived_data/archive
df_recovered = pd.read_csv(url)


#download death cases data from JHU dashboard
url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/archived_data/archive
df_death = pd.read_csv(url)


df_tmp_subset = df_confirmed.drop(['Province/State','Lat','Long'],axis=1)
df_tmp_subset = pd.DataFrame(df_tmp_subset.groupby(['Country/Region'],as_index=False).sum())
df_tmp_subset = df_tmp_subset.sort_values(by=df_confirmed.columns[len(df_confirmed.columns)-1
df_topten_countries = df_tmp_subset[0:10]
df_topten_countries.shape
```

```
(10, 63)
```

```
import plotly.graph_objects as go

x_axis = df_topten_countries.columns

y0 = df_topten_countries.iloc[0,1:].values.flatten().tolist()
y1 = df_topten_countries.iloc[1,1:].values.flatten().tolist()
y2 = df_topten_countries.iloc[2,1:].values.flatten().tolist()
y3 = df_topten_countries.iloc[3,1:].values.flatten().tolist()
y4 = df_topten_countries.iloc[4,1:].values.flatten().tolist()
y5 = df_topten_countries.iloc[5,1:].values.flatten().tolist()
y6 = df_topten_countries.iloc[6,1:].values.flatten().tolist()
y7 = df_topten_countries.iloc[7,1:].values.flatten().tolist()
y8 = df_topten_countries.iloc[8,1:].values.flatten().tolist()
y9 = df_topten_countries.iloc[9,1:].values.flatten().tolist()
```

```
y9 = df_topten_countries.iloc[9,1:].values.flatten().tolist()

# Create traces
fig = go.Figure()
annotations = []
fig.add_trace(go.Scatter(x=x_axis[1:], y=y0,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[0,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y1,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[1,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y2,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[2,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y3,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[3,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y4,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[4,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y5,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[5,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y6,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[6,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y7,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[7,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y8,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[8,0]))
fig.add_trace(go.Scatter(x=x_axis[1:], y=y9,
                    mode='lines+markers',
                    name=df_topten_countries.iloc[9,0]))
# Title
annotations.append(dict(xref='paper', yref='paper', x=0.0, y=1.05,
                              xanchor='left', yanchor='bottom',
                              text='Confirmed Case Trend of Top Ten Impacted Countries',
                              font=dict(family='Arial',
                                        size=30,
                                        color='rgb(37,37,37)'),
                              showarrow=False))
fig.update_layout(annotations=annotations)
fig.show()
```



Bubble Map Plot

https://plotly.com/python/bubble-maps/

```
# first, subset rows from CONFIRMED that have the US as a country

df_confirmed_usa = df_confirmed[df_confirmed['Country/Region'] == 'US']
df_confirmed_usa = df_confirmed_usa.rename(columns= {df_confirmed_usa.columns[len(df_confirme

# let's drop all rows where there is a comma in the field for "Province/State"
# return ALL ROWS where it does not contain a comma (that's what ~ means)
df_confirmed_usa = df_confirmed_usa[~df_confirmed_usa['Province/State'].str.contains(",")]

df_confirmed_usa.drop(df_confirmed_usa.iloc[:,4:-1], axis=1, inplace=True)
df_confirmed_usa.drop("Country/Region", axis=1)
df_confirmed_usa = df_confirmed_usa.dropna(axis=0)
df_confirmed_usa.shape
```

```
(57, 5)
```

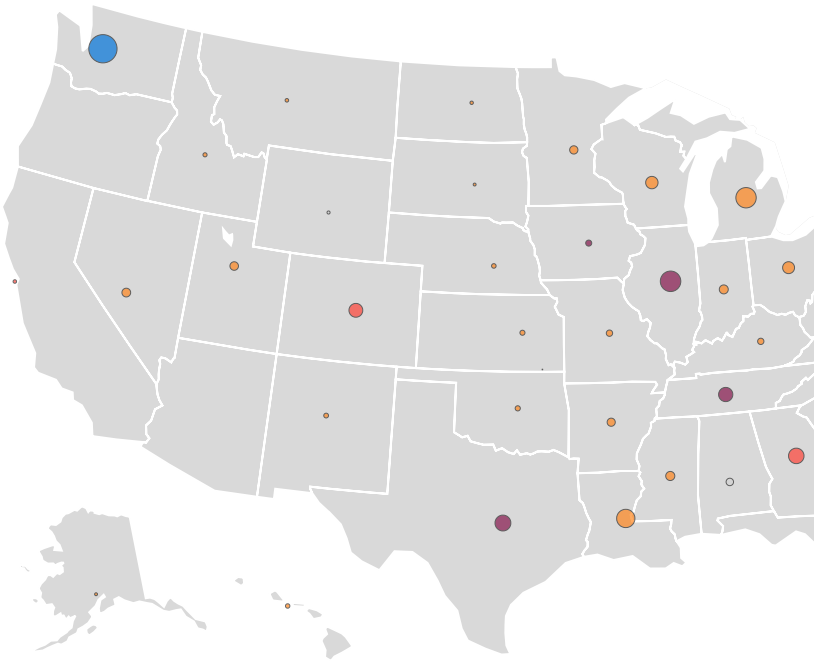
```
!pip install chart-studio

import plotly.express as px

import chart_studio.plotly as py
import plotly.graph_objects as go
```

```
Collecting chart-studio
  Downloading https://files.pythonhosted.org/packages/ca/ce/330794a6b6ca4b9182c38fc69dd2
     |████████████████████████████████| 71kB 2.1MB/s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from chart
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from ch
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packa
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

```
df_confirmed_usa['text'] = df_confirmed_usa['Province/State'] + '<br>Confirmed Count ' + (df_
limits = [(0,2),(3,10),(11,20),(21,50),(50,3000)]
colors = ["rgb(0,116,217)","rgb(255,65,54)","rgb(133,20,75)","rgb(255,133,27)","lightgrey"]
cities = []
scale = 20


for i in range(len(limits)):
    lim = limits[i]
    df_sub = df_confirmed_usa[lim[0]:lim[1]]
    city = go.Scattergeo(
        locationmode = 'USA-states',
        lon = df_sub['Long'],
        lat = df_sub['Lat'],
        text = df_sub['text'],
        marker = go.scattergeo.Marker(
            size = df_sub['ConfirmedCount']/scale,
            color = colors[i],
            line = go.scattergeo.marker.Line(
                width=0.5, color='rgb(40,40,40)'
            ),
            sizemode = 'area'
        ),
        name = '{0} - {1}'.format(lim[0],lim[1]) )
    cities.append(city)

layout = go.Layout(
        title = go.layout.Title(
            text = 'Geospatial Plot:Confirmed Cases in US Cities/Provinces<br>(Click legend t
        ),
        showlegend = True,
        geo = go.layout.Geo(
            scope = 'usa',
            projection = go.layout.geo.Projection(
                type='albers usa'
            ),
            showland = True
```

```
            showland = True,
            landcolor = 'rgb(217, 217, 217)',
            subunitwidth=1,
            countrywidth=1,
            subunitcolor="rgb(255, 255, 255)",
            countrycolor="rgb(255, 255, 255)"
        )
    )

fig = go.Figure(data=cities, layout=layout)
fig.show()
```

Geospatial Plot:Confirmed Cases in US Cities/Provinces (Click legend to toggle traces)



```
## Heat map table
```

Heat map table

```
#confirmed case count in USA
df_usa_conf = df_confirmed[df_confirmed['Country/Region']=="US"]
df_usa_conf = df_usa_conf.rename(columns= {df_usa_conf.columns[len(df_usa_conf.columns)-1]:"C

# let's drop all rows where there is a comma in the field for "Province/State"
```

```
# return ALL ROWS where it does not contain a comma (that's what ~ means)
df_usa_conf = df_usa_conf[~df_usa_conf['Province/State'].str.contains(",")]

#new cases each day
df_state = df_usa_conf['Province/State']
df_dates = df_usa_conf.iloc[:,-23:-1]
df_dates = df_dates.diff(axis=1)
df_usa_conf = pd.concat([df_state,df_dates.iloc[:,1:],df_usa_conf['ConfirmedCount']],axis=1)
df_usa_conf.shape
```

```
(58, 23)
```

```
#death case count in USA
df_usa_death = df_death[df_death['Country/Region']=="US"]
df_usa_death = df_usa_death.rename(columns= {df_usa_death.columns[len(df_usa_death.columns)-1
df_usa_death = df_usa_death.loc[:, ['Province/State','DeathCount']]

# let's drop all rows where there is a comma in the field for "Province/State"
# return ALL ROWS where it does not contain a comma (that's what ~ means)
df_usa_death = df_usa_death[~df_usa_death['Province/State'].str.contains(",")]

#merge and treat NaN
df_table_set = pd.merge(df_usa_conf, df_usa_death, on='Province/State', how='outer')
df_table_set = df_table_set.fillna(0)
df_table_set.shape

#sort table in descending order of confirmed cases
df_table_set = df_table_set.sort_values(by='ConfirmedCount', ascending=False)


x = ["Diamond Princess","Wuhan Repatriated","Puerto Rico",
     "Guam", "Northern Mariana Islands","Grand Princess",
     "District of Columbia", "US", "Virgin Islands", "United States Virgin Islands"]

df_table_set = df_table_set[~df_table_set['Province/State'].isin(x)]

df_table_set.shape
```

```
(50, 24)
```

```
df_table_set.columns = ["State/Province","D21-7", "D21-6", "D21-5", "D21-4", "D21-3", "D21-2"
                        "D14-7", "D14-6", "D14-5", "D14-4", "D14-3", "D14-2", "D14-1",
                        "D7-7", "D7-6", "D7-5", "D7-4", "D7-3", "D7-2", "D7-1",
                        "ConfirmedCount","DeathCount"]

df_table_set['ConfirmedCount'] = df_table_set['ConfirmedCount'].astype('object')
df_table_set['DeathCount'] = df_table_set['DeathCount'].astype('object')

for column in df_table_set.columns:
  if df_table_set[column].dtype == 'float':
    df_table_set[column] = df_table_set[column].astype('int')
```

```
USAState = [('USA_States', col) for col in df_table_set.columns if 'State/Province' in col]
colsLast7 = [('Last_Seven_Days', col) for col in df_table_set.columns if 'D7-' in col]
colsLast14 = [('Last_Fourteen_Days', col) for col in df_table_set.columns if 'D14-' in col]
colsLast21 = [('Last_TwentyOne_Days', col) for col in df_table_set.columns if 'D21-' in col]
totalStats_conf = [('Total_Stats', col) for col in df_table_set.columns if 'ConfirmedCount' i
totalStats_death = [('Total_Stats', col) for col in df_table_set.columns if 'DeathCount' in c

th_props = [
  ('font-size', '12px'),
  ('text-align', 'left'),
  ('font-weight', 'bold'),
  ('color', '#6d6d6d'),
  ('background-color', '#f7f7f9')
  ]

td_props = [
  ('font-size', '11px')
  ]
styles = [
  dict(selector="th", props=th_props),
  dict(selector="td", props=td_props)
  ]

df_table_set.columns = pd.MultiIndex.from_tuples(USAState + colsLast21 + colsLast14 + colsLas
df_table_set = df_table_set.style.background_gradient(cmap='YlOrBr', axis=1).set_table_styles
print("                                          NEW CASES IN THE LAST THREE WEEKS
print("                                          Last Updated On:",df_confirmed.columns[len(
df_table_set
```

NEW CASES IN THE LAST THREE WEEKS
Last Updated On: 3/23/20

	USA_States	Last_TwentyOne_Days							Last_Fourteen_Days							Last_Se	
	State/Province	D21-7	D21-6	D21-5	D21-4	D21-3	D21-2	D21-1	D14-7	D14-6	D14-5	D14-4	D14-3	D14-2	D14-1	D7-7	D7
1	New York	0	0	0	0	0	0	0	0	173	47	108	93	104	207	235	739
0	Washington	0	0	0	0	0	0	0	0	267	99	76	126	4	71	261	172
9	New Jersey	0	0	0	0	0	0	0	0	15	8	6	0	40	29	80	89
2	California	0	0	0	0	0	0	0	0	144	33	44	61	58	86	131	141
12	Illinois	0	0	0	0	0	0	0	0	12	13	7	14	18	29	12	56
44	Michigan	0	0	0	0	0	0	0	0	0	2	0	14	9	8	20	12
36	Louisiana	0	0	0	0	0	0	0	0	1	5	13	17	41	14	45	60
8	Florida	0	0	0	0	0	0	0	0	15	13	7	15	26	39	40	61
3	Massachusetts	0	0	0	0	0	0	0	0	92	3	13	15	15	26	33	21
11	Texas	0	0	0	0	0	0	0	0	13	8	6	16	14	15	13	25
6	Georgia	0	0	0	0	0	0	0	0	17	6	8	11	24	33	22	25
13	Pennsylvania	0	0	0	0	0	0	0	0	12	4	6	19	6	19	11	35
18	Tennessee	0	0	0	0	0	0	0	0	7	2	9	8	6	7	13	22
7	Colorado	0	0	0	0	0	0	0	0	15	19	11	4	52	30	29	0
30	Wisconsin	0	0	0	0	0	0	0	0	3	3	2	11	8	5	15	25
28	Ohio	0	0	0	0	0	0	0	0	3	1	1	8	13	11	13	17
16	North Carolina	0	0	0	0	0	0	0	0	7	0	8	2	7	9	5	26
15	Maryland	0	0	0	0	0	0	0	0	8	1	3	6	8	6	9	19
31	Connecticut	0	0	0	0	0	0	0	0	2	1	2	6	11	2	6	38
19	Virginia	0	0	0	0	0	0	0	0	7	2	8	13	11	4	4	18
45	Mississippi	0	0	0	0	0	0	0	0	0	0	1	0	5	4	3	8
21	Indiana	0	0	0	0	0	0	0	0	6	5	2	0	3	4	5	5
17	South Carolina	0	0	0	0	0	0	0	0	7	3	2	1	6	9	5	14
24	Nevada	0	0	0	0	0	0	0	0	4	3	7	3	4	3	21	11
34	Utah	0	0	0	0	0	0	0	0	2	1	2	4	1	18	11	12
26	Minnesota	0	0	0	0	0	0	0	0	3	2	4	5	7	14	19	6
40	Arkansas	0	0	0	0	0	0	0	0	0	1	5	0	6	4	6	0
10	Oregon	0	0	0	0	0	0	0	0	15	4	5	6	2	4	3	27
20	Arizona	0	0	0	0	0	0	0	0	6	3	0	0	3	1	5	2
52	Alabama	0	0	0	0	0	0	0	0	0	0	0	5	1	6	17	10
37	Missouri	0	0	0	0	0	0	0	0	1	0	0	1	2	1	1	5
22	Kentucky	0	0	0	0	0	0	0	0	6	2	2	4	0	6	1	5
14	Iowa	0	0	0	0	0	0	0	0	8	5	3	1	0	1	5	0
43	Maine	0	0	0	0	0	0	0	0	0	0	0	1	2	9	5	15
29	Rhode Island	0	0	0	0	0	0	0	0	3	2	0	9	6	0	1	2
33	Oklahoma	0	0	0	0	0	0	0	0	2	0	0	0	2	3	3	9
25	New Hampshire	0	0	0	0	0	0	0	0	4	1	1	0	1	6	4	9
35	Kansas	0	0	0	0	0	0	0	0	1	0	0	4	3	0	3	7
47	New Mexico	0	0	0	0	0	0	0	0	0	3	2	5	0	3	4	6
38	Vermont	0	0	0	0	0	0	0	0	1	0	1	0	3	3	4	0
27	Nebraska	0	0	0	0	0	0	0	0	3	2	5	3	1	3	1	3
32	Hawaii	0	0	0	0	0	0	0	0	2	0	0	0	2	2	1	3
41	Delaware	0	0	0	0	0	0	0	0	0	1	0	3	2	1	1	8
42	Idaho	0	0	0	0	0	0	0	0	0	0	0	1	1	3	0	3
46	Montana	0	0	0	0	0	0	0	0	0	1	0	0	4	2	0	2
48	North Dakota	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2
51	Wyoming	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	8
49	South Dakota	0	0	0	0	0	0	0	0	0	8	0	0	1	0	1	1
39	Alaska	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2