COVID19 Visualization using Geographic Choropleth Maps in Python Using Plotly

Visualizing the outbreak of COVID-19

Imports :

We will start by importing all the modules necessary to create our visualizations. Here the data collection is done through the API so ‘urllib3, requests’ are imported , for plotting ‘plotly’ imports are required as well as for data storing and accessing ‘csv ’ and ‘pandas’ are imported.

In [1]:

```
import urllib3
import csv
import requests
import plotly as py
from plotly.offline import download_plotlyjs,init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
import pandas as pd
import plotly.graph_objects as go
```

Making a request with requests , r is our response object. We can get all information we need from this object.

In [2]:

```
r = requests.get('https://api.covid19api.com/summary')
json_data = r.json()
#print(json_data)
```

In [3]:

```
fname = "outputbyJSON.csv"
```

Here we are copying the json to csv file why because it’s easy to manipulate and use. The writerow module is used to write rows, and we are iterating through json_data[‘Countries’] to get Country, CountryCode, NewConfirmed, TotalConfirmed, NewDeaths, TotalDeaths, NewRecovered, TotalRecovered, Date.

In [4]:

```
with open(fname,"w") as file:
    csv_file = csv.writer(file)
    csv_file.writerow(["Country","CountryCode","NewConfirmed","TotalConfirmed"
                       ,"NewDeaths","TotalDeaths","NewRecovered","TotalRecovered","Date","text"])
    for item in json_data['Countries']:
        csv_file.writerow([item['Country'],item['CountryCode'],
                           item['NewConfirmed'],item['TotalConfirmed'],
                           item['NewDeaths'],item['TotalDeaths'],item['NewRecovered'],
                           item['TotalRecovered'],item['Date'],
                           'TotalConfirmed = '+str(item['TotalConfirmed'])
                           +'      TotalDeaths = '+str(item['TotalDeaths'])+
                           '      TotalRecovered = '+str(item['TotalRecovered'])
                          ])
```

In [5]:

```
df= pd.read_csv("outputbyJSON.csv",encoding='ISO-8859-1')
```

In [6]:

```
df.head()
```

Out[6]:

	Country	CountryCode	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecovered	TotalRecovered	Date
0	ALA Aland Islands	AX	0	0	0	0	0	0	2020-04-19T08:22:44Z
1	Afghanistan	AF	27	933	0	30	13	112	2020-04-19T08:22:44Z
2	Albania	AL	9	548	0	26	19	302	2020-04-19T08:22:44Z
3	Algeria	DZ	116	2534	3	367	48	894	2020-04-19T08:22:44Z
4	American Samoa	AS	0	0	0	0	0	0	2020-04-19T08:22:44Z

Geographical Plotting

The map is plotted using Plotly's graph_objs module that we imported. It requires two important parameters that have to be passed as arguments, data and layout.Each of these parameters consists of a dictionary of parameters and arguments.

Initializing the DATA variable.

type : 'choropleth' specifies that we are plotting a choropleth map.

locations : The names of countries we want to plot.

locationmode : It specifies that the plotting level is country wise. The value can be one of 3,-"ISO-3" , "USA- states" , "country names".

colorscale : The colour set used to plot the map.Available color scales are 'Greys', 'YlGnBu', 'Greens', 'YlOrRd', 'Bluered', 'RdBu', 'Reds', 'Blues', 'Picnic', 'Rainbow', 'Portland', 'Jet', 'Hot', 'Blackbody', 'Earth', 'Electric', 'Viridis', 'Cividis'

text: The textual information that needs to be displayed for each country on hover.

z: The value or factor that is used to distinguish the countries. These values are used by the colour scale.

colorbar: A dictionary of parameters and arguments to customize the display of colorbar. Used to control the properties of the colorbar such as length, title, axis etc.

In [7]:

```
data = dict(type = 'choropleth',
            locations = df['Country'],
             locationmode = 'country names',
             z = df['TotalConfirmed'],
             text =df['Country'],
             colorscale= 'agsunset',
             reversescale = False,
             marker = dict(line = dict(color='white',width=1)),

             colorbar = {'title':'members'}
            )
```

Initializing the LAYOUT variable.

geo: The parameter sets the properties of the map layout. The scope parameter sets the scope of the map. Scope can have any of the 7 values- "world" | "usa" | "europe" | "asia" | "africa" | "north america" | "south america" .

In [8]:

```
layout = dict(title = 'World wide #TotalConfirmed# covid19 cases',
              geo = dict(showframe = False,
                         projection = {'type':'natural earth'})
              )
#"natural earth"
```

Initializing the Figure object by passing data and layout as arguments.

In [9]:

```
choromap = go.Figure(data=[data],layout=layout)
```

Plotting the map.

In [11]:

```
#iplot(choromap)
#plot(choromap,validate=False,filename='TotalConfirmed.html')
```

In [ ]:

```

```

In [14]:

```
data = dict(type = 'choropleth',
           locations = df['Country'],
            locationmode = 'country names',
            z = df['TotalDeaths'],
            text = df['Country'],
            colorscale= 'agsunset',
            reversescale = False,
            marker = dict(line = dict(color='white',width=1)),

            colorbar = {'title':'members'}
           )
```

In [15]:

```
layout = dict(title = 'World wide total #TotalDeaths# covid19 cases',
              geo = dict(showframe = False,
                         projection = {'type':'orthographic'})
              )
```

In [16]:

```
choromap = go.Figure(data=[data],layout=layout)
```

In [17]:

```
#iplot(choromap)
#plot(choromap,validate=False,filename='TotalDeaths.html')
```

In [ ]:

```

```

In [18]:

```
data = dict(type = 'choropleth',
           locations = df['Country'],
            locationmode = 'country names',
            z = df['TotalRecovered'],
            text = df['Country'],
            colorscale= 'agsunset',
            reversescale = False,
            marker = dict(line = dict(color='white',width=1)),
            colorbar = {'title':'members'}
           )
```

In [19]:

```
layout = dict(title = 'World wide total #TotalRecovered# covid19 cases',
             geo = dict(showframe = False,
                        projection = {'type':"mollweide"})
             )
```

In [20]:

```
choromap = go.Figure(data=[data],layout=layout)
```

In [21]:

```
#iplot(choromap)
#plot(choromap,validate=False,filename='TotalRecovered.html')
```

In [ ]:

```

```

In [22]:

```
df= pd.read_csv("outputbyJSON.csv",encoding='ISO-8859-1')
```

In [28]:

```
piePlot =
go.Pie(labels=df['Country'],values=df['TotalRecovered'],title='TotalRecovered',titleposition='top
left')
```

In [29]:

```
#iplot([piePlot])
```

In [30]:

```
piePlot =
go.Pie(labels=df['Country'],values=df['TotalDeaths'],title='TotalDeaths',titleposition='top left')
```

In [31]:

```
#iplot([piePlot])
```

In [32]:

```
piePlot =
go.Pie(labels=df['Country'],values=df['TotalConfirmed'],title='TotalConfirmed',titleposition='top
left')
```

In [33]:

```
#iplot([piePlot])
```

In [ ]:

```

```

In [ ]:

In [34]:

```
trace = go.Bar(x=df['Country'],y=df['TotalConfirmed'])
```

In [35]:

```
data = [trace]
```

In [36]:

```
fig = go.Figure(data=data)
```

In [37]:

```
#iplot(fig)
```

In [39]:

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

In [40]:

```
df= pd.read_csv("outputbyJSON.csv",encoding='ISO-8859-1')
fig = px.treemap(df, path=['Country'], values=df['TotalConfirmed'],
                  title="World wide total #TotalConfirmed# covid19 cases ",
                  color_continuous_scale='RdBu',
                 )
#fig.show()
```

In [41]:

```
df= pd.read_csv("outputbyJSON.csv",encoding='ISO-8859-1')

fig = px.treemap(df, path=['Country'], values=df['TotalDeaths'],
                  title="World wide total #TotalDeaths# covid19 cases ",
                  color_continuous_scale='RdBu',
                 )
#fig.show()
```

In [43]:

```
df= pd.read_csv("outputbyJSON.csv",encoding='ISO-8859-1')
fig = px.treemap(df, path=['Country'], values=df['TotalRecovered'],
                  title="World wide total #TotalRecovered# covid19 cases ",
                  color_continuous_scale='RdBu',
                 )
#fig.show()
```

In [ ]:

```

```