Johns Hopkins COVID-19 Data Analysis - version 1

Data : https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

Code developed by Swanand Mhalagi

For further info

https://www.linkedin.com/in/swanand-mhalagi-851613a7/

https://github.com/swanandM

https://medium.com/@swan1991m

Below code uses Linux packages like NumPy, Pandas, Plotly and Cufflinks

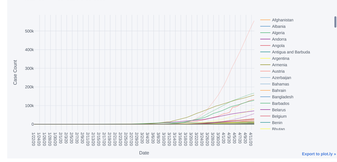
In [1]:
```
#silent download of the CSV files
!wget -N -q --timestamping https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv

!wget -N -q --timestamping https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv

!wget -N -q --timestamping https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv
```

In [2]:
```
import numpy as np # linear algebra
import pandas as pd
import cufflinks as cf
import plotly.offline
import plotly.graph_objects as go

cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)

# show all pandas columns and only 2 decimal points
pd.set_option("display.max.columns", None)
pd.set_option("display.precision", 3)
```

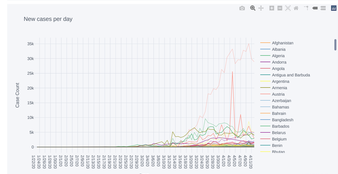
In [3]:
```
df_csv = pd.read_csv('time_series_covid19_confirmed_global.csv')
df_reco = pd.read_csv('time_series_covid19_recovered_global.csv')
df_ded = pd.read_csv('time_series_covid19_deaths_global.csv')
```

In [4]:
```
def world_case_status(df_csv):
    df = df_csv.drop(['Province/State'],inplace=True)
    #Delete unnecessary columns
    df_csv_conf = df_csv_conf.drop(['Province/State','Lat','Long'], axis=1)
    #set date as index
    df_csv_conf = df_csv_conf.set_index('Country/Region')
    #Transpose data
    #Diff between row column
    df_world = df_csv_conf.diff(axis=1)
    #df_world.head()
    df_world = df_world.T
    df_world
    df_csv_conf = df_csv_conf.T
    return df_csv_conf.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='Total Confirmed Case Distribution'), df_world.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='New cases per day')
```

In [5]:
```
world_case_status(df_csv)
```



Export to plot.ly »



Export to plot.ly »

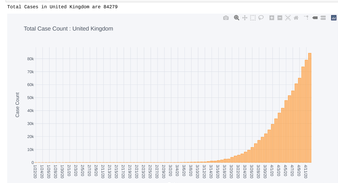
Out[5]: (None, None)

In [ ]:
```
```

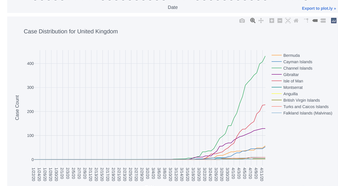
In [6]:
```
def for_country(df_csv, Country):
    df = df_csv[df_csv['Country/Region'].str.contains(Country)]
    Region = df['Country/Region'].values[0]
    #State Wise
    df_State_conf = df[df['Province/State'].notnull()]
    df_State_conf.head()
    df_State_conf = df_State_conf.set_index('Province/State')
    #Select column
    df_State_conf = df_State_conf.loc[:,'1/22/20':]
    df_State_conf = df_State_conf.T

    #Select data for whole country
    df_Country_conf = df[df['Province/State'].isnull()]
    #df_Country_conf
    total = df_Country_conf.iloc[:,4].values[0]
    print('Total Cases in '+ Region +' are '+ str(total) )
    df_Country_conf = df_Country_conf.loc[:,'1/22/20':]
    df_Country_conf = df_Country_conf.T

    #Daily increase rate
    daily_inc = df_Country_conf.T
    daily_inc = daily_inc.diff(axis=1)
    daily_inc = daily_inc.T

    return df_Country_conf.iplot(kind='bar', xTitle='Date', yTitle='Case Count', title='Total Case Count : ' + Region ), df_State_conf.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='Case Distribution for ' + Region), daily_inc.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='Daily increase rate : ' + Region )
```

In [7]:
```
for_country(df_csv, "United Kingdom")
```

```
Total Cases in United Kingdom are 84279
```



Export to plot.ly »



Export to plot.ly »

Export to plot.ly »

Out[7]: (None, None, None)

In [8]:
```
def con_rec_ded(Country):
    #look for particular country
    series = df_csv[df_csv['Country/Region'].str.contains(Country)]
    series = series[series['Province/State'].isnull()]
    # Change the row indexes
    series.index = ['Confirmed']
    series

    #Dead Cases
    df_ded_con = df_ded[df_ded['Country/Region'].str.contains(Country)]
    df_ded_con = df_ded_con[df_ded_con['Province/State'].isnull()]
    # Change the row indexes
    df_ded_con.index = ['Dead']
    df_ded_con

    #Recovered cases
    df_reco_con = df_reco[df_reco['Country/Region'].str.contains(Country)]
    df_reco_con = df_reco_con[df_reco_con['Province/State'].isnull()]
    # Change the row indexes
    df_reco_con.index = ['Recovered']
    df_reco_con

    #Select column
    df_ded_con = df_ded_con.loc[:,'1/22/20':]
    df_ded_con = df_ded_con
    df_reco_con = df_reco_con.loc[:,'1/22/20':]
    df_reco_con = df_reco_con.T
    series = series.loc[:,'1/22/20':]
    series = series.T

    result = pd.concat([series, df_ded_con, df_reco_con], axis=1, join='inner')
    return result.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='Case Distribution for ' + Country)
```

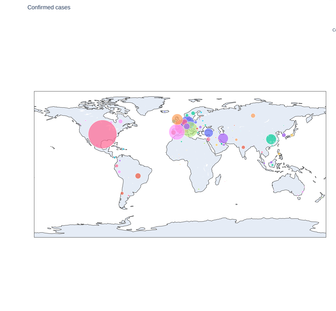
In [9]:
```
con_rec_ded("United Kingdom")
```

Export to plot.ly »

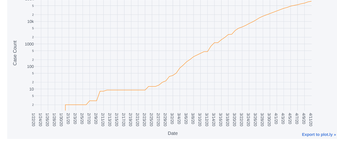
In [10]:
```
#World Map
import plotly.express as px
```

In [11]:
```
def world_plot(date):
    df_world_today = df_csv.loc[:,date:]
    df_info = df_csv.loc[:,'Province/State':'Long']
    df_world_today = pd.concat([df_info, df_world_today], axis=1, join='inner')
    return px.scatter_geo(df_world_today, lat='Lat', lon='Long', color="Country/Region",
                          hover_name="Country/Region", size=date,
                          title='Confirmed cases',
                          size_max=int(60),
                          width=1200, height=1000,)
```

In [12]:
```
world_plot("4/13/20")
```



In [61]:
```
#Total Count Log graph
df_Country_conf = df_Country_conf.T
df_Country_conf.iplot(mode='lines', xTitle='Date', yTitle='Case Count', title='Logarithmic Case Distribution for ' + Region, yaxis_type="log")
```



Export to plot.ly »