Introduction

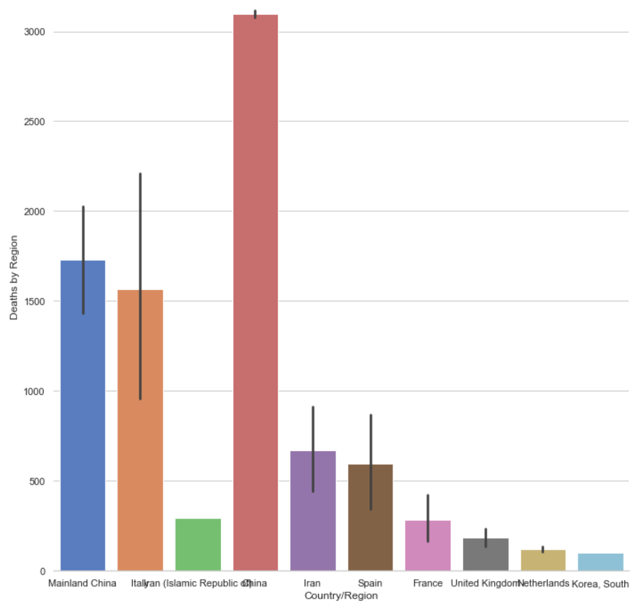
In the exploratory phase of the project, I started analyzing COVID-19 dataset open sourced by yahoo (https://github.com/yahoo/covid-19-data.git). It is a comprehensive and verified database updated daily. However, it requires additional time and wrangling of multiple datasets to prepare the data for the analysis for this project. I was able to get it working with pyspark set up. Spark is the industry standard framework for large scale data processing. It is non-trivial to get PySpark setup on Jupyter Notebook, hence this piece of analysis is in its own repo – (https://github.com/skopp002/covid_analysis). For subsequent project work, I switched back to CSSE dataset provided in Assignment details, in order to collaborate with the rest of the team members.

The approach taken for the project is to perform exploratory analysis on the data by looking at the datasets from geographic perspective and subsequently analyzing the timeseries for total confirmed cases and total death cases per Country. Eventually we will be using Facebook's Opensource library for forecasting based on historic trends.

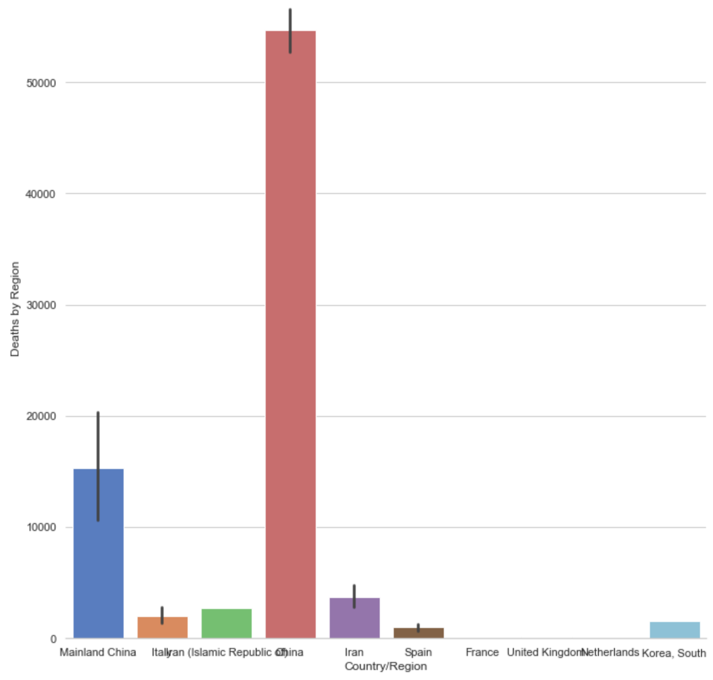
Data Exploration

Analysis in the following section is based on data below:
COVID-19/csse_covid_19_data/csse_covid_19_daily_reports/

Plotting the Deaths by location shows the below trend:

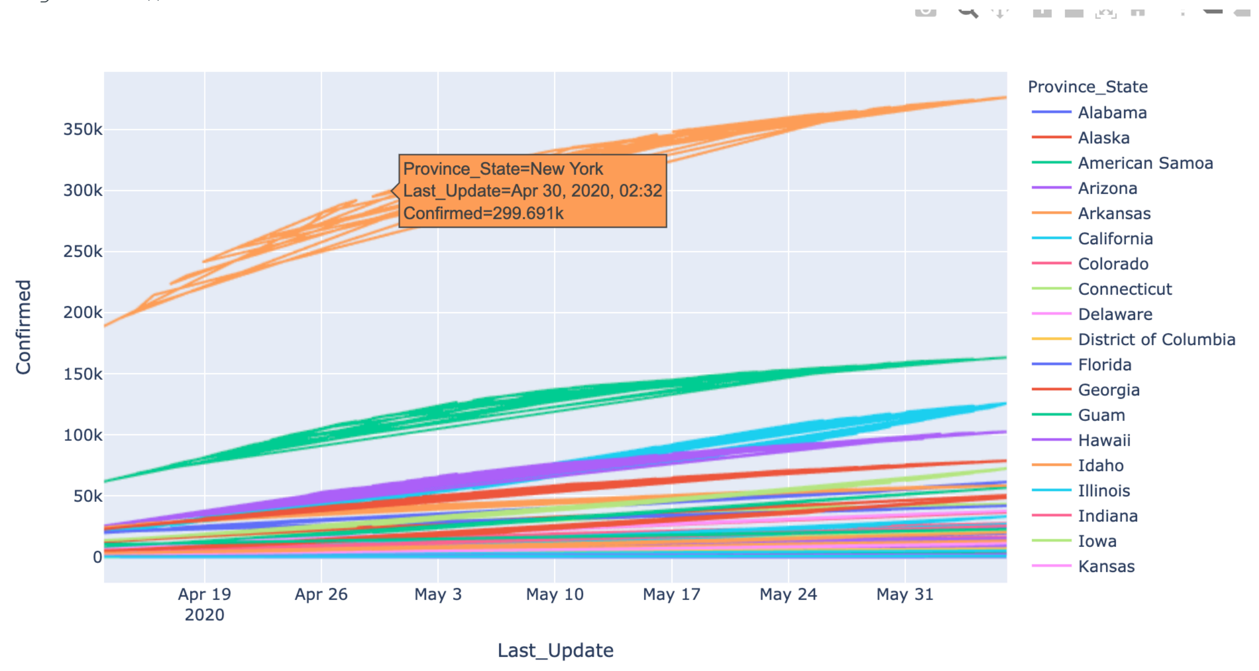


Below is the recovery ratio by Region:



In order to use the lookup data, I used the US case report dataset. Below is a timeseries based chart with US data:

```
fig = go.Figure()
fig = px.line(usdf, x="Last_Update", y="Confirmed",
color='Province_State')
fig.show()
```



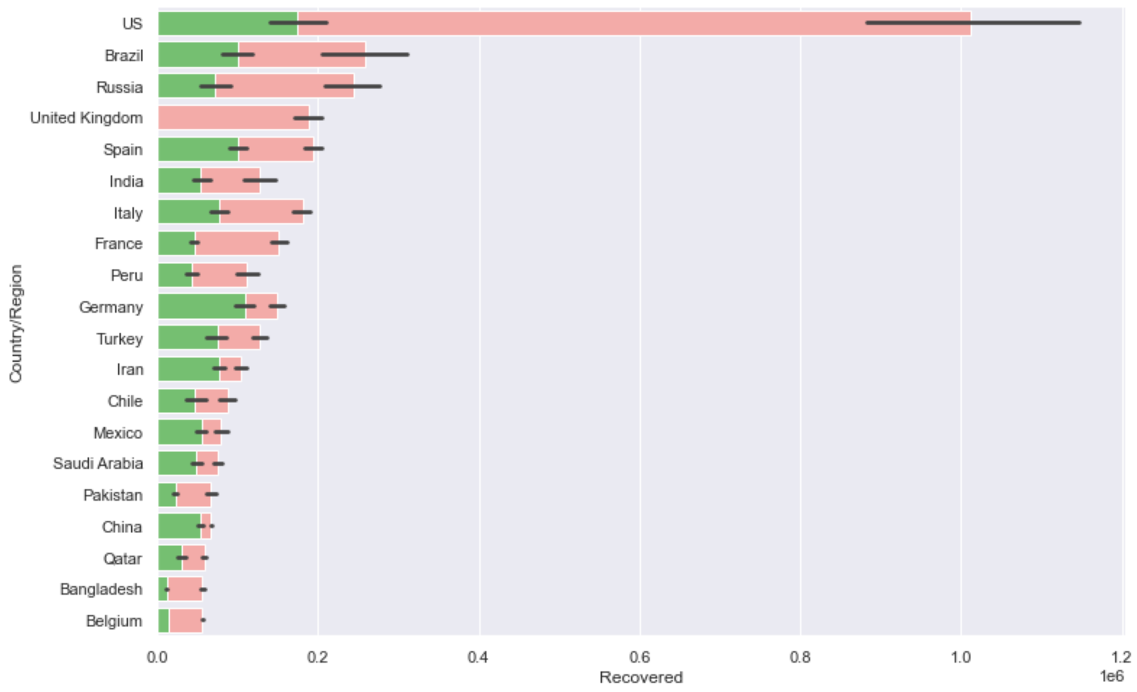
The above datasets help understand the overall impact. As we can see highest impact in US is in New York, which is evident from News Reports as well.

Switching to the global time series datasets:
COVID-19/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_<>_global.csv

Looking at the global timeseries for Confirmed, recovered and deaths, we see trends like below:

```
gl_top_ts=gl_ts[gl_ts['Confirmed']>50000]
f,ax=plt.subplots(figsize=(12,8))
data=gl_top_ts[['Country/Region','Confirmed','Deaths','Recovered','Active']]
data.sort_values('Confirmed',ascending=False,inplace=True)
sns.set_color_codes("pastel")
sns.barplot(x="Confirmed",y="Country/Region",data=data, label="Total",color='r')
sns.set_color_codes("muted")
sns.barplot(x="Recovered",y="Country/Region",data=data, label="Recovered",color='g')
```

In order to make useful interpretations from this data, I had to filter for countries with more than 50000 cases. Interesting observations from the below charts, United Kingdom shows very low, almost nil recovery rates and is lower than Italy.



China shows fewer cases going by the time series since most of the initial cases have either recovered or died.

Lets zoom in and understand the trends for countries with highest number of cases and interesting patterns:

Referring to SIR Model (Source: here),

By definition
S(t) are those susceptible but not yet infected with the disease
In our dataset, the *population* can be sort of mapped to susceptible.

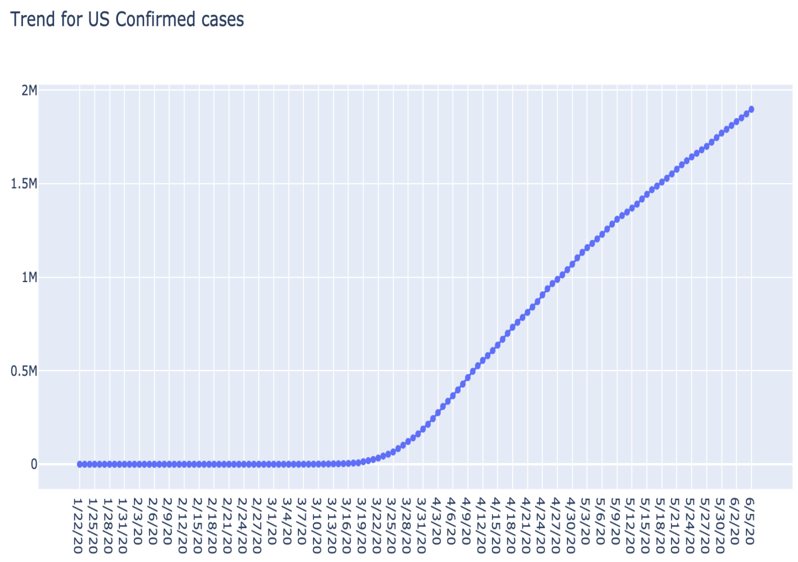
I(t) is the number of infectious individuals;
Confirmed cases can all be considered to belong to this category.

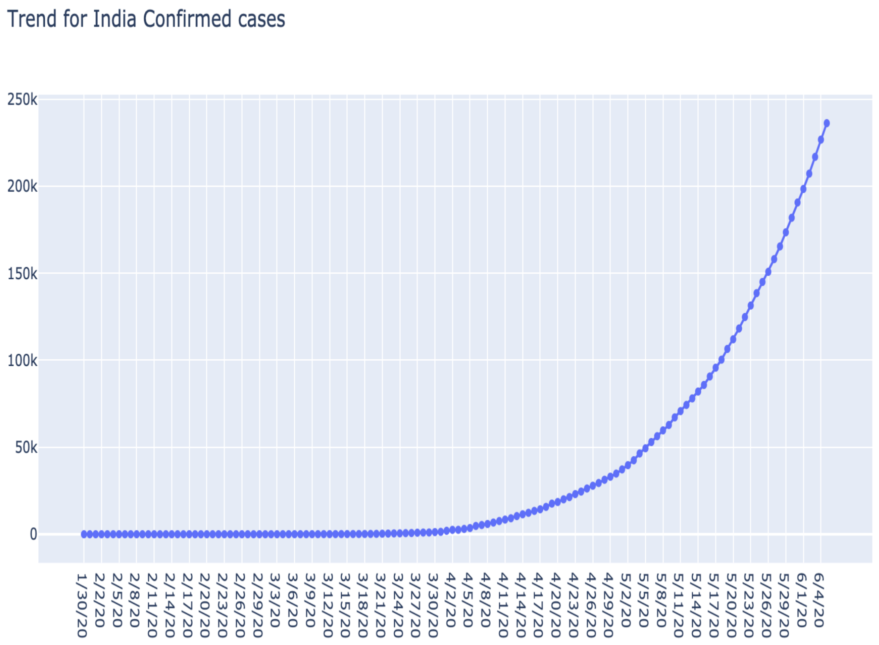
R(t) are those individuals who have recovered from the disease and now have immunity to it or are dead.

Using *plotly*, we can create interactive graphs. Here is what the graphs look like:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=gl_top_ts['Date'],y=gl_top_ts['Confirmed']
                        ,mode='lines+markers',name='Confirmed Time series'))
fig.update_layout(title_text='Trend across countries with highest confirmed cases')

fig.show()
```

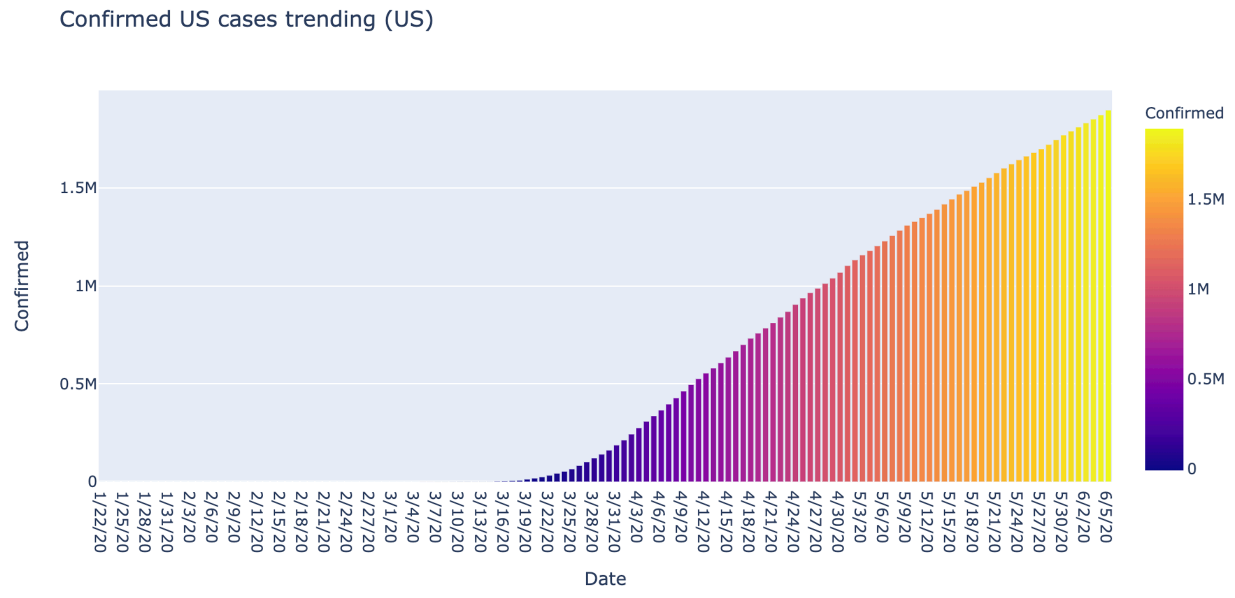
When filtered for US the chart shows:

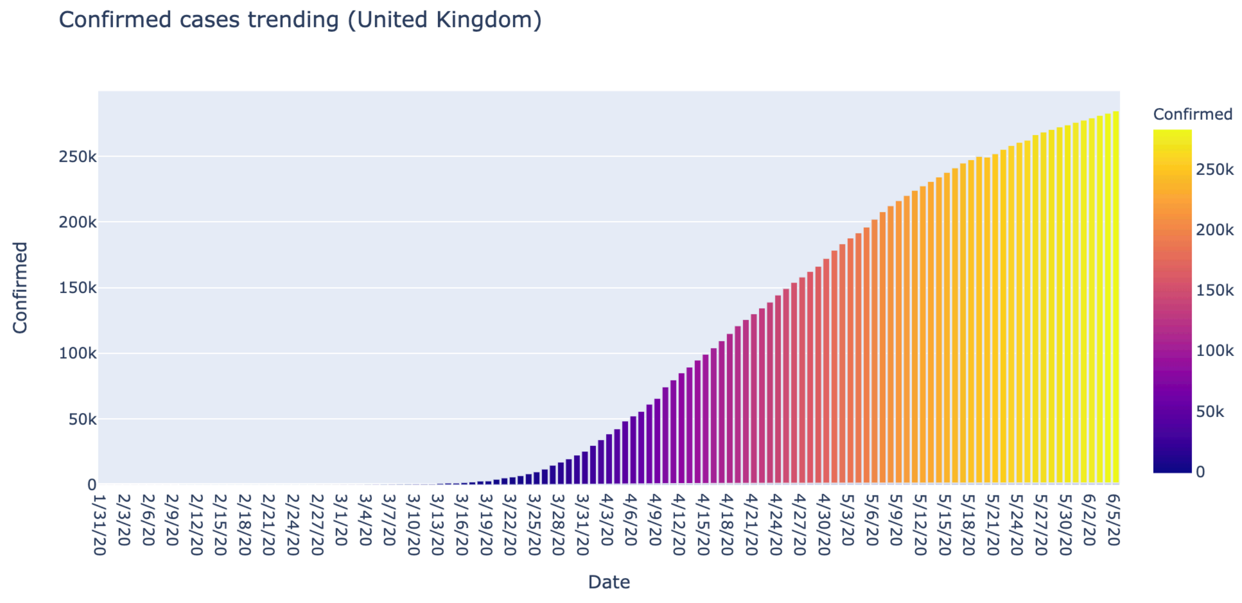


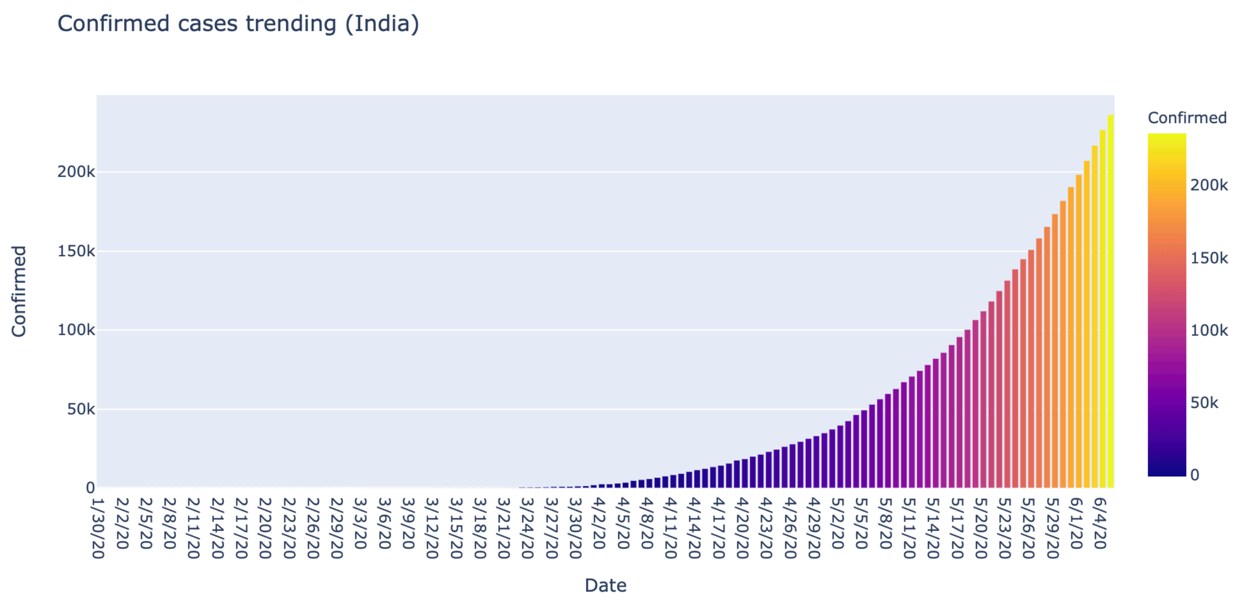


It is evident from the above trends that India is still in the initial stages of the pandemic. The pandemics are typically exponential curves since the way they infect goes in exponential fashion. For instance, 1 infected person could spread to 2. 2 -> 4 -> 8.. etc. It could be much more

Another interesting trending view is below:

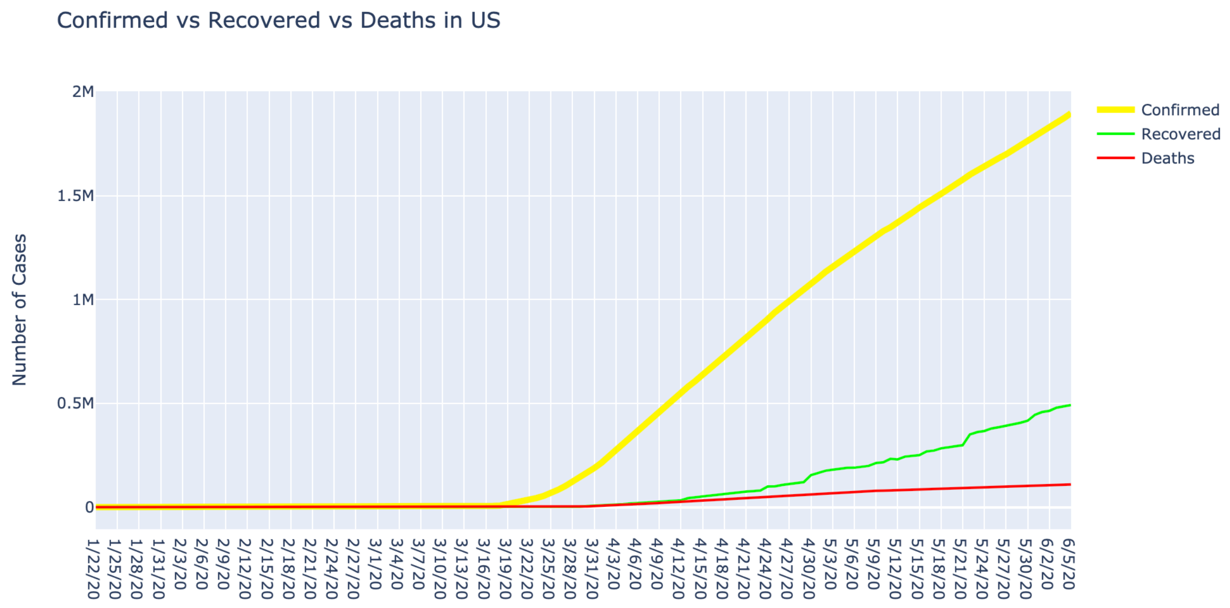






Forecasting

Next, looking at the trending to fit the SIR model. Let's focus on US to build the model.



For a pandemic, typical curves would be something like below where confirmed cases would plateau and then begin to decline, Death rates would also decline and Recovery will spike high:

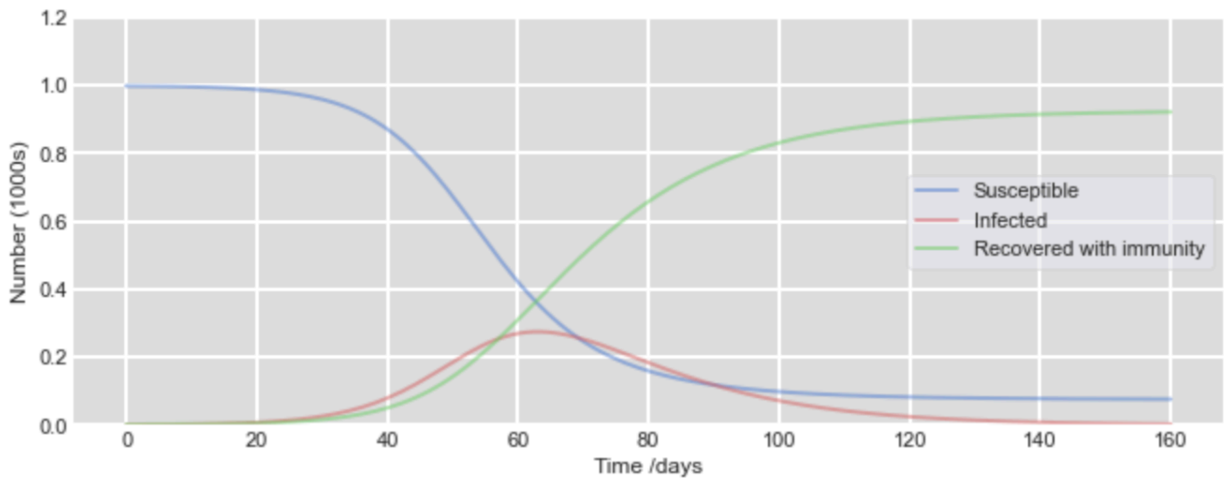
Referring to SIR Model,
By definition
S(t) are those susceptible but not yet infected with the disease
The entire population is susceptible to infection.

I(t) is the number of infectious individuals;
Confirmed cases can all be considered to belong to this category

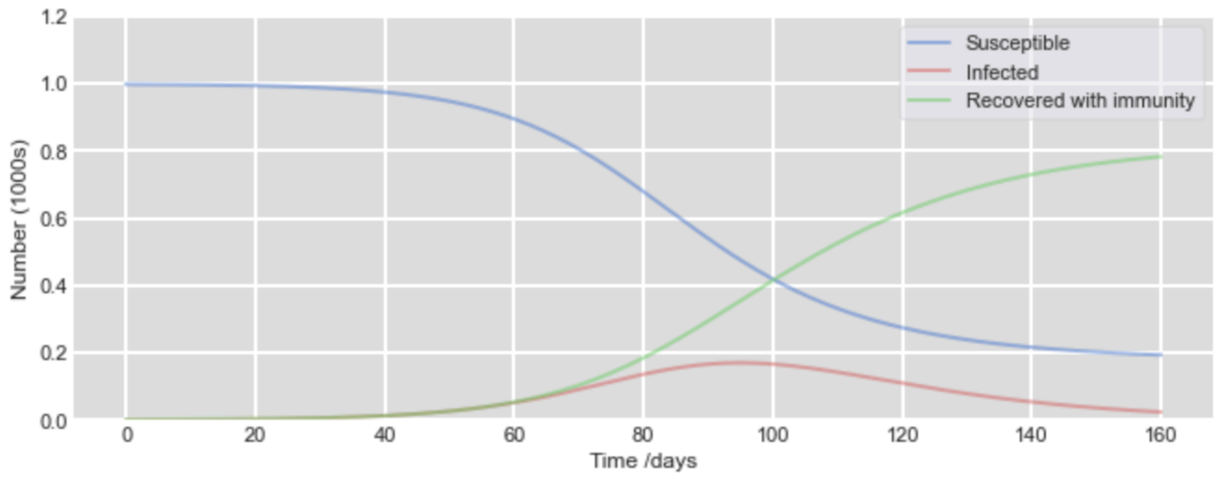
R(t) are those individuals who have recovered from the disease and now have immunity to it.
Both Recovered and Death cases are sort of in this category for this analysis

β is the contact rate. Usually for any pandemic it is considered to be 0.2 let's take β to be 0.2

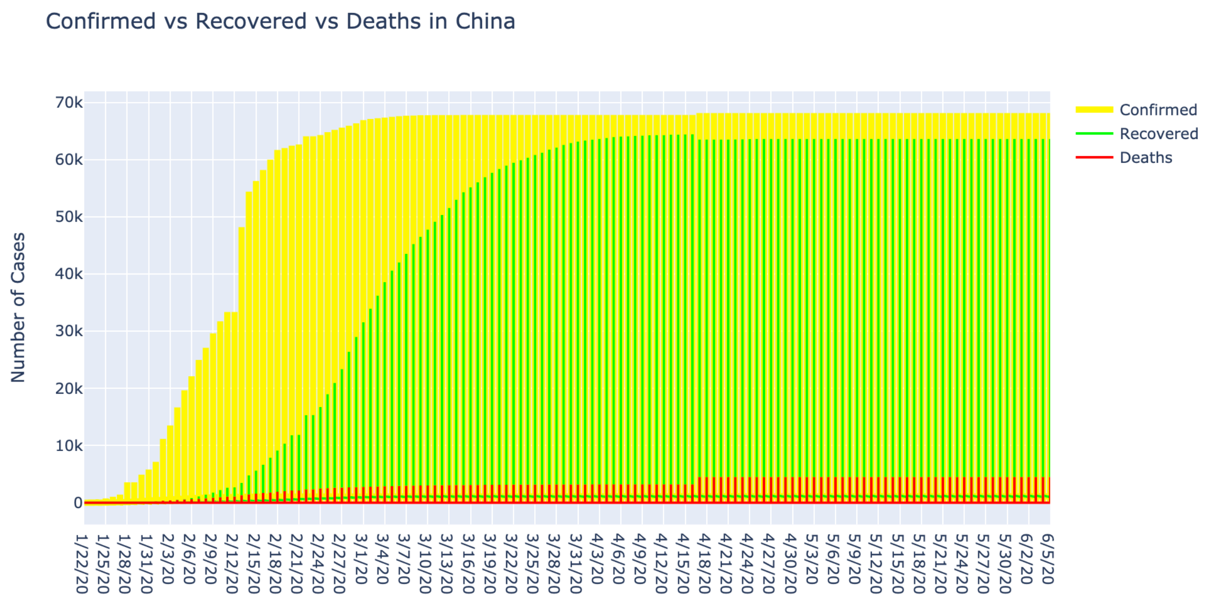
γ is the mean recovery rate. We know that it is 14 days since that is the typical quarantine time we are asked to wait.



By reducing the contact rate to 0.15 to represent lockdown, we can show through the plots how the death rate goes lower and the rate at which the entire population is infected also diminishes



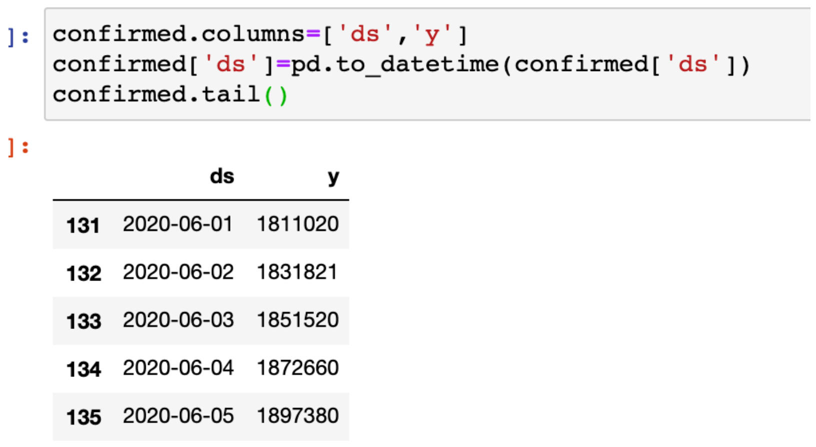
As we can see from all our trends, the pandemic is still on the rise, the plateauing state has probably arrived in places which were initially hit for instance like China.



Based on this analysis, let us try to create a prediction model. We will use Facebook's Opensource library - Prophet. Prophet uses generalized additive model which in laymen terms means, instead of using coefficients for every parameter, the model will have functions for every parameter and the final result will be an additive of multiple such derivatives. This idea is similar to Neural Networks concept.

Prophet detects changes in trends based on change points in the data. Hence it is crucial to have the “date” column in our dataset to be in “Datetime” format and also have the column name “ds”. The count of Confirmed cases can be mapped to ‘y’ which is also a required label. Prophet also allows customizations to specify seasonality. Since COVID is not a seasonal pandemic, we will stick to defaults which is “no seasonality”

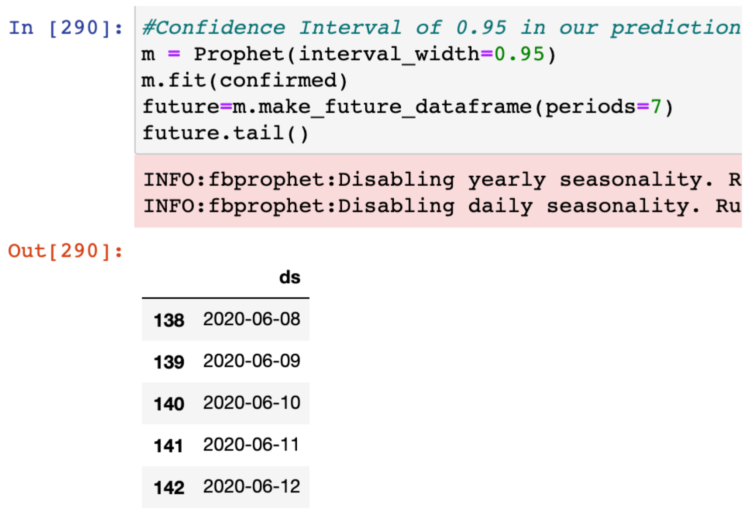
We can create a model with these 2 variables and check the prediction. The way prophet model works is by splitting the data in train and test.
The latest data we have from the dataset is:



```
]: confirmed.columns=['ds','y']
   confirmed['ds']=pd.to_datetime(confirmed['ds'])
   confirmed.tail()
]:
```

	ds	y
131	2020-06-01	1811020
132	2020-06-02	1831821
133	2020-06-03	1851520
134	2020-06-04	1872660
135	2020-06-05	1897380

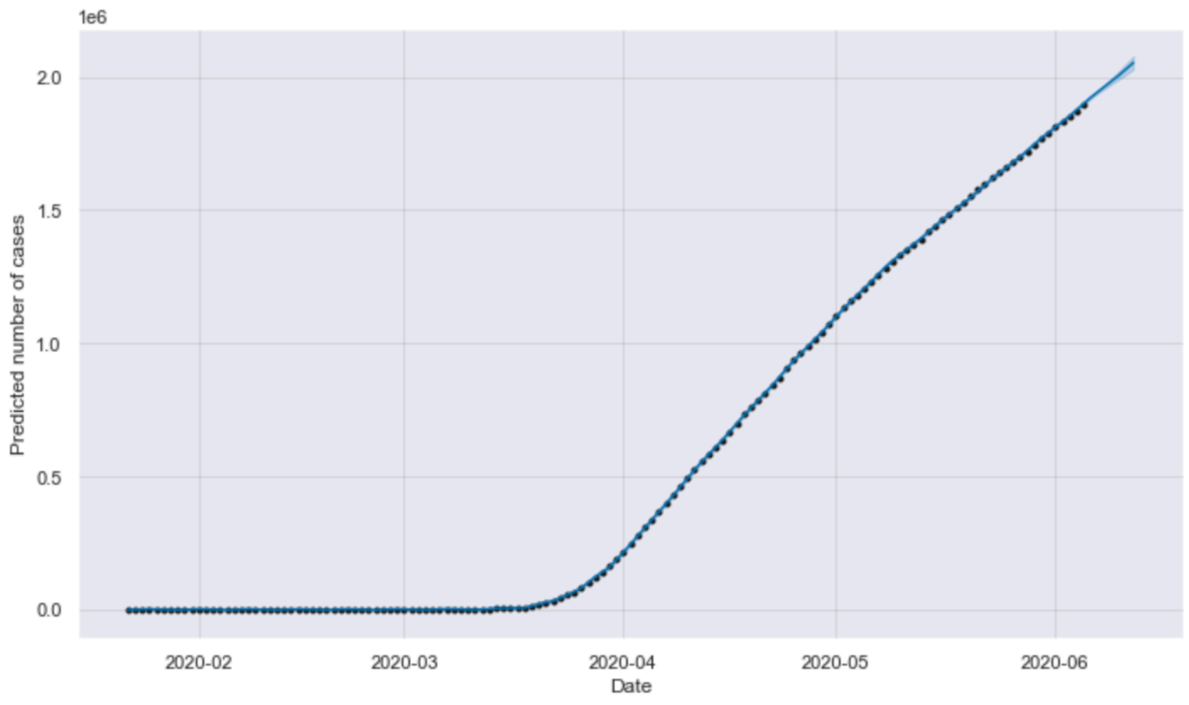
We can also specify the period to project in future:

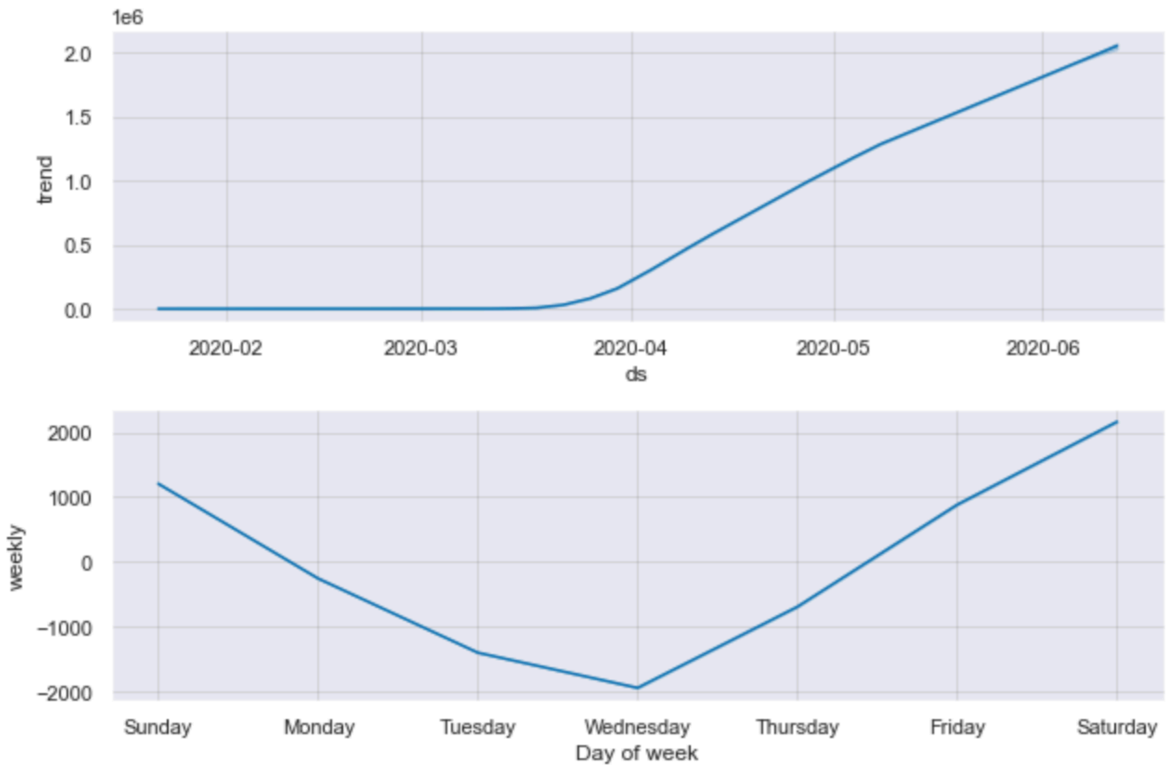


```
In [290]: #Confidence Interval of 0.95 in our prediction
          m = Prophet(interval_width=0.95)
          m.fit(confirmed)
          future=m.make_future_dataframe(periods=7)
          future.tail()

          INFO:fbprophet:Disabling yearly seasonality. R
          INFO:fbprophet:Disabling daily seasonality. Ru

Out[290]:
```

	ds
138	2020-06-08
139	2020-06-09
140	2020-06-10
141	2020-06-11
142	2020-06-12

Given the future dataframe, we can make use of the predict method to have the estimated y i.e total number of cases. We can create similar plots for Confirmed, Recovered and Death cases as well. Based on this mapping, the predictor output is as below:





The predictions assume similar trends and transmitting rates as historic data.

Below is the accuracy of the model:

```
  horizon           mse          rmse           mae      mape     mdape  \
0  1 days  2.058718e+08  14348.234857  12527.701915  0.008794  0.005799
1  2 days  2.368942e+08  15391.368024  12514.703693  0.008304  0.005831
2  3 days  3.621879e+08  19031.235707  15223.975487  0.010074  0.006032
3  4 days  4.546979e+08  21323.647155  17588.255652  0.011630  0.006506
4  5 days  6.280586e+08  25061.097342  21827.571200  0.014507  0.010161
5  6 days  7.758749e+08  27854.531159  25228.866076  0.016651  0.013757
6  7 days  8.857064e+08  29760.820082  27073.107869  0.017758  0.016215
```

Appendix

During my research on approaches to visualize and plot the data on a geographical map, I came across matplotlib's toolkit based on *basemap*. It has multiple dependencies and eventually fails with build error on basemap toolkit for Mac OS. During the troubleshooting of the errors I came across cartopy. Cartopy depends on PROJ as well but can be installed without building it on the laptop. Also, Cartopy is an more evolved than basemap. It was an interesting exploration of exploratory tools ranging from basemap -> Cartopy -> GeoView -> finally folium.

Visually exploring the metadata and comparing it to fields in the dataset, it is clear that "regionId" in the dataset corresponds to "id" in the metadata.
The approach on data analytics is:

1. Plot the population with total deaths, total cases and total recovered on the map which will help determine the impact
2. Plot timeseries graphs for the datasets with ReferenceDate on X-axis and the same 4 variables – (population with total deaths, total cases and total recovered) plotted with line charts

Lot of the data is null as we can see from the below analysis:

```
totalDeaths                  105334
totalConfirmedCases            4319
totalRecoveredCases          254833
totalTestedCases             256837
numPositiveTests              14316
numDeaths                    116200
numRecoveredCases            241075
avgWeeklyDeaths              134210
avgWeeklyConfirmedCases       41479
avgWeeklyRecoveredCases      264271
woeId                          9678
```

Further details on this approach and the github repo is here
https://github.com/skopp002/covid_analysis

References

- https://www.youtube.com/watch?v=_Hi6_JQesSQ
- https://scipython.com/book/chapter-8-scipy/additional-examples/the-sir-epidemic-model/
- https://en.wikipedia.org/wiki/Additive_model
- https://towardsdatascience.com/forecasting-with-prophet-d50bbfe95f91
- https://plotly.com/python/reference/