COVID-19-DATA-ANALYSIS-AND-VISUALIZATION

A Project Work Report

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

BIG DATA AND ANALYTICS

Submitted by:

YASH SRIVASTAVA - 20BCS4443

KHWAHISH AGARWAL – 20BCS4447

YOGESH GELHOT – 20BCS4454

SWETA NEGI – 20BCS4422

Under the supervision of:

Puneet Kaur Bhatia

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUTE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI – 140413, PUNJAB

March 2022

Table of Contents

List of Tables

List of Figures

List of Symbols

Symbol	*Description*
A_{set}	*Area of steel reinforcement bars on tension face*
A_{Sc}	*Area Of steel reinforcement bars on compression face*
$A_{vs.}$	*Area of two legs of the closed stirrups*
b	*The breadth of rectangular beam section*
d	*Effective depth of rectangular beam section*
d'	*Effective cover on compression face*
f_{cave}	*Average compressive stress in concrete*
f_{Sc}	*Stress in steel on the compression side*
f_{ee}	*Characteristic strength of steel reinforcement bars Spacing of the stirrups*
S_{vi}	*Depth of neutral axis from compression face*

ABSTRACT

Since December 2019 the world is passing a deadly complaint caused by a new coronavirus nominated as severe acute respiratory pattern coronavirus 2(SARS- CoV- 2). The complaint associated with this contagion is known as COVID- 19. This project and paper focus on COVID- 19 grounded on freely available datasets including the bones in Coursera MATLAB course. Data analytics is handed on a number of aspects of COVID- 19 including the symptoms of this complaint, the difference of COVID-19 with other conditions caused by severe acute respiratory pattern (SARS), Middle East respiratory pattern (MERS), and swine flu. The impact of temperature on the spread of COVID- 19 is also bandied grounded on the datasets. also, data visualization is handed on the comparison of infections in males ladies which shows that males are more prone to this complaint and the aged people are more at threat. Grounded on the data, the pattern in the increase of verified cases is set up to be an exponential wind in nature. Eventually, the relative number of verified, recovered and death cases in different countries are shown with data visualization.

1. INTRODUCTION

1.1 Problem Definition

The COVID-19 outbreak, which first emerged in China, has spread worldwide. On March 11, 2020, the World Health Organization (WHO) declared COVID-19 as a pandemic. The disease has disrupted global trade, employment, and travel, and many governments had to take strict measures to control the spread of the virus and minimize the burden of morbidity and mortality so that health care systems remain functional. In many countries around the world, citizens have been recommended to stay at home and practice social distancing for as long as possible as a primary measure of preventing the spread of COVID-19.

Although apps are successfully used for managing chronic diseases, the ongoing COVID-19 pandemic has pushed the need for app solutions at the forefront to reduce the risk of cross-contamination caused by close contact. this technology has been leveraged in several ways to control the spread of COVID-19. Apps are accessible, acceptable, and easily adopted, and could support social distancing efforts. As such, they have been widely developed and implemented during the previous months to "flatten the curve" of the increasing number of COVID-19 cases, providing knowledge and information to civilians while attempting to relieve the pressure from health care systems.

1.2 Project Definition

The goal of our project is to create a MATLAB program that processes and visualizes COVID-19 pandemic data. Once you load it, you will get a single variable called covid data that is a large cell array. Each data cell for a given country and date contains a 2-element vector of doubles: the first element is the cumulative case count, while the second is the cumulative number of deaths.

Our program converts this data into a set of objects: one object per country and state. States should be contained by their countries. Countries could be stored in a vector of country objects in the app itself. Another way is to create an instance of the same class we use for countries and states, call it global, and have it store all the countries. The app would then contain the single global object as a property. This option would create a 3-level hierarchy: the global object stores data for the entire world and a vector of country objects, while the objects of countries that have states in the database would store their corresponding states. Again, we can use the same class definition for all three kinds of objects because they store essentially the same kind of data.

1.3 Software Specifications:

- 4 GB RAM minimum, 8 GB RAM recommended.
- 120GB of available disk space minimum, 4 GB RAM recommended.
- 1280 x 800 minimum screen resolution.
- Microsoft Windows 7/8/10 (32 or 64-bit).
- Application Browsers (like Chrome, Safari, Brave, etc.)
- MATLAB

Windows requirements

- Windows (10 recommended)
- 3 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB
- 1280 x 800 minimum screen resolution

Mac OS requirements

- Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (High Sierra)
- 3 GB RAM minimum, 8 GB RAM recommended
- GB of available disk space minimum, 4 GB
- 1280 x 800 minimum screen resolution.

2. LITERATURE SURVEY

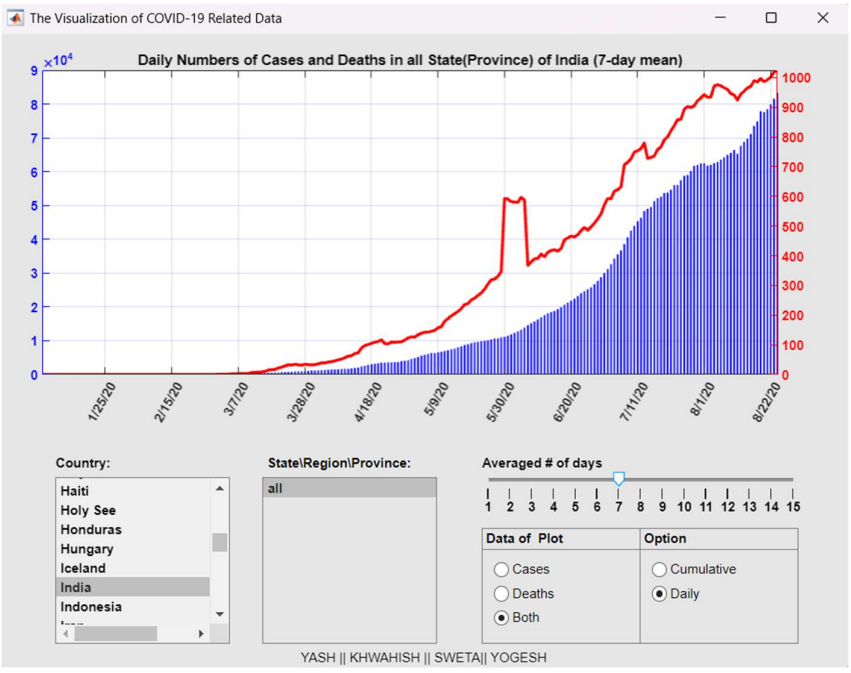
2.1 Existing System

In the existing systems, the majority of the data scientist are working on the ground level, they have their personal or government websites or application and users explore them via the internet and read and see detail about the current status of covid which is provided on the website or application. Multiple apps are also made but those were not very much convenient in order to use.

2.2 Proposed System

The aim of the proposed system is to develop a system of improved Application. The proposed system can overcome all the limitations of the existing system. The system provides the following services:

- Security of data
- Ensure data accuracy
- Proper control from admins
- Minimize manual data entry
- Minimum time is needed for various processes
- Greater efficiency
- Better service
- User friendless and interactive



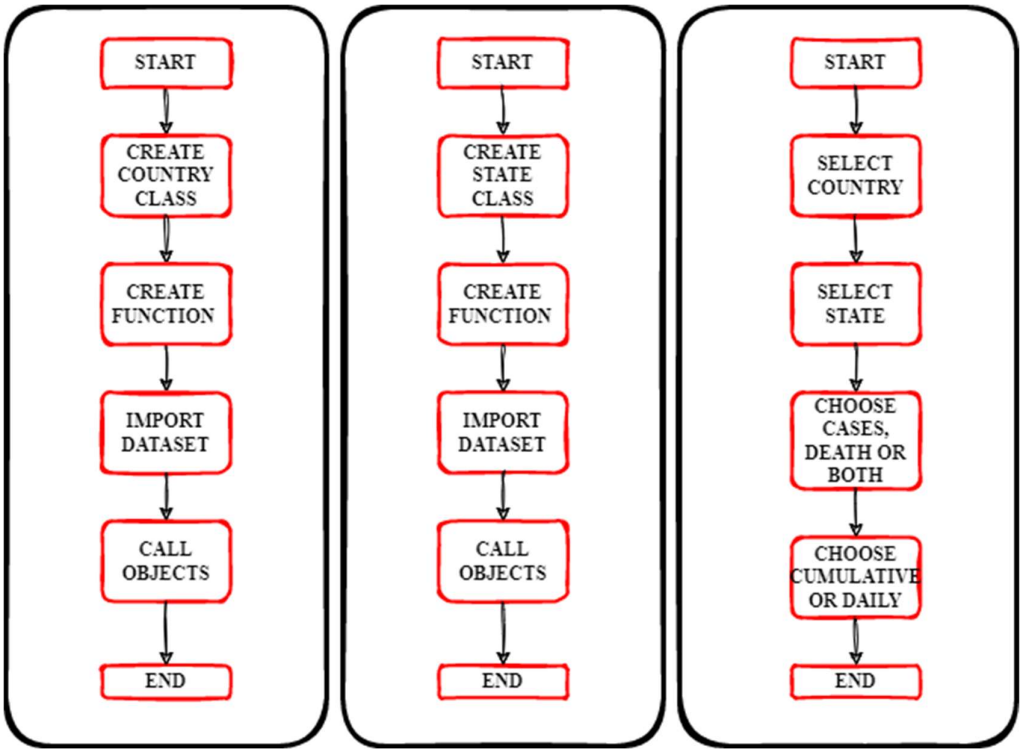
3. PROBLEM FORMULATION

This project is a App bases Application using the MATLAB concept of designing a application. This application helps the people to visualize the proper graph or trend of covid 19 to the needy ones. The app will act as an intermediate between government and people. There are three parts:

1. **COUNTRY CLASS: -** In this class we are creating function and object to call and fetch data from data.mat dataset to show all the country names in our app and we can easily choose the country to see covid cases of that country .
2. **STATE CLASS: - -** In this class we are creating function and object to call and fetch data from data.mat dataset to show all the states names in our app and we can easily choose the country and state to see covid cases of that state in chosen country.

3. **APP: -** After opening the App first we have an option to choose the country and states Then we have to select the option of cases and death and we can chose both to watch cases and death trend at a same time then we have an option to chose the data type like cumulative or daily type then we have a slider for average of 15 days data.

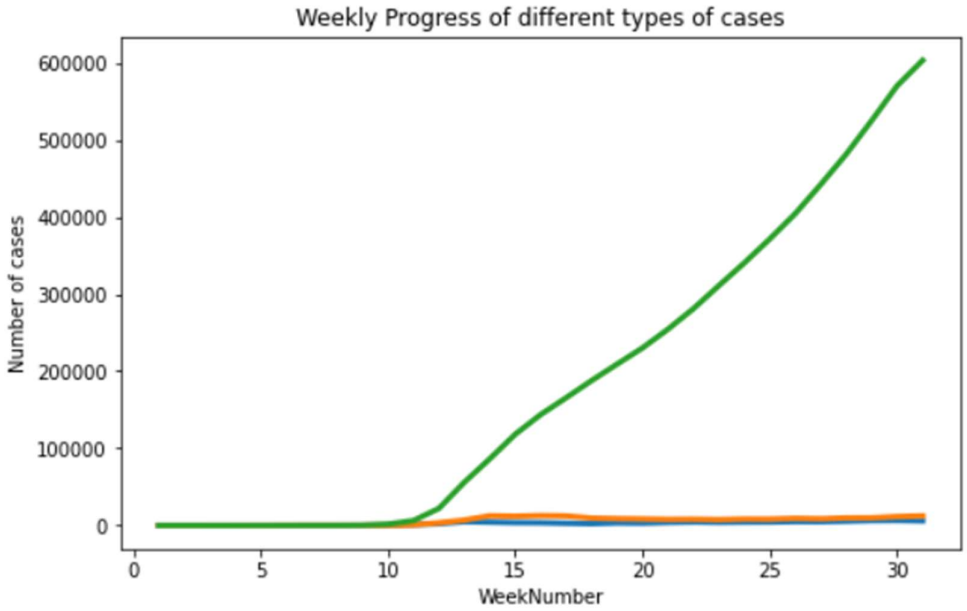
Activity program for country | Activity program for state | Activity diagram for App



4. METHODOLOGY

As the name itself means ‘a system of methods used in a particular area of study or activity’. In this section we are going to give a detailed explanation of workflow of our application and methods involved in it sequence diagram.

Type of Research done:
On March 11, 2020, the World Health Organization (WHO) declared the novel coronavirus (Covid19) outbreak as a global pandemic. In this paper, a time series analysis to predict the number of deaths in the United States starting from August 1st — August 21st and August 1st — November 1st is modelled and studied. The time series model that was selected to make the prediction.



By analysing the current situation and scenario we came up with this project idea.

Workflow of Our Application:
We are designing a app which enables users to easily watch the trends of covid 19 cases and death and using Time Series Analysis for prediction of upcoming covid waves.

Let’s see how this all works-

Application: -

1) Created a country class to extract the data of different country of covid 19 cases and deaths.
2) Created a state class to extract the data of different states of different countries to find and analysis the cases of covid 19 and prevent any future issue related covid in that region.
3) Created the app to visualized the data of different country and states in one place and have to many another features like we can see the trend of cases , deaths and both of particular

country and sates at same time then we have a another option of cases like cumulative or daily types of cases then we can also see the data in average using the slider.

Time Series Analysis: -

Time series analysis (TSA) is a statistical technique that consists of data points listed in time order. The x axis is made up of equally spaced points in time and the y axis contains the outcome values that are going to be projected from our model based on previous observed values. This technique is suitable for research questions such as forecasting future sales. The reason why time series analysis exists, is since the outcome variable in our model is dependent on one single explanatory variable only: time.

Suppose you run a shoe store and have the data available that tells you how many shoes you have sold in the past years. Given the data available, time series analysis would be applicable if you would like to predict how many shoes your store will sell in the future. In this case, the outcome variable would be the number of shoes sold and the one and only explanatory variable would be time.

Other forecasting algorithms such as linear regression or logistic regression use one or more explanatory variables. Further there is a difference when it comes to the assumptions when comparing linear regression, logistic regression, and the time series technique ARIMA.

In **Linear Regression** the following assumptions have to be met:

- Independence of observations.
- Homoscedasticity of errors (equal variance).
- A linear relationship.
- Errors are normally distributed.

In **Logistic Regression** the following assumptions have to be met:

- Dependent variable has to binary.
- Independence of observations.
- Linearity in the logit for continuous variables.
- Lack of influential outliers
- Absence of multicollinearity

5. OBJECTIVES

The objective of our application is to enable a link of communication and interactions among public and Government.

- During the current coronavirus pandemic, monitoring the evolution of COVID-19 cases is of utmost importance for the authorities to make informed policy decisions (e.g., lockdowns), and to raise awareness in the general public for taking appropriate public health measures.
- At the time of the pandemic outbreak, a lack of laboratory tests, materials, and human resources implied that the evolution of officially confirmed cases did not represent the total number of cases.
- Even now, there are significant differences across countries in terms of the availability of tests. For this reason, given the rapid progression of the pandemic, in some cases health authorities are forced to make important decisions based on sub-optimal data.
- For this reason, alternatives to testing that can be rapidly deployed are likely to help authorities, as well as the general population, to better understand the progress of a), particularly at its early stages or in low-income countries, where massive testing is unfeasible.
- To this end, we have created a system, named COVID-19-DATA-ANALYSIS-AND-VISUALIZATION to estimate the number of COVID-19 cases based on crowd-sourced open anonymous surveys.

6. TENTATIVE PLAN FOR THE PROJECT

In this project we have to visualized and analyses the data of covid 19 so for visualization we have MATLAB Application and for Analytics we have Time Series Analytics using Python and R.

Application:

1) **Introduction:**
The goal of our project is to create a MATLAB program that processes and visualizes COVID-19 pandemic data. Once you load it, you will get a single variable called covid data that is a large cell array. Each data cell for a given country and date contains a 2-element vector of doubles: the first element is the cumulative case count, while the second is the cumulative number of deaths.

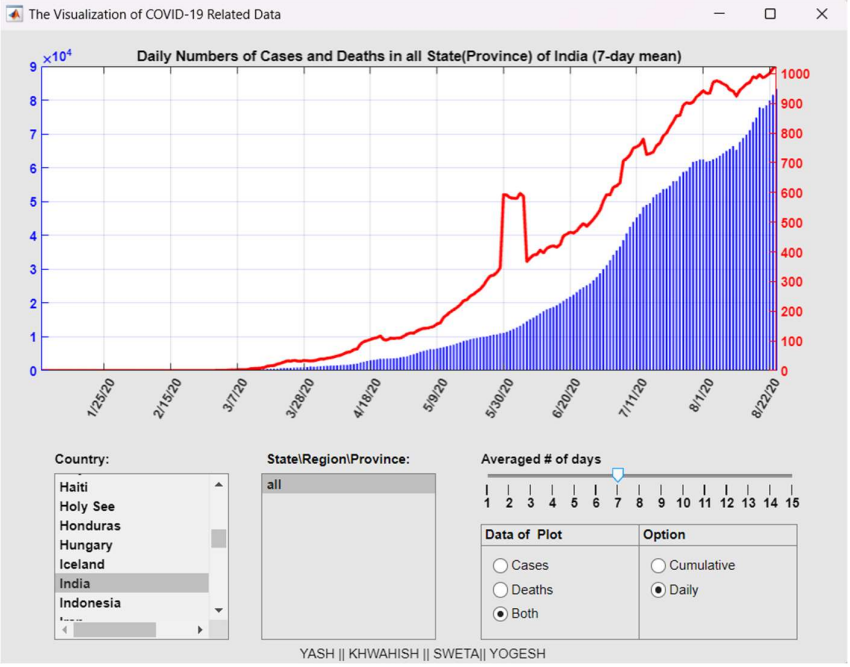
Our program converts this data into a set of objects: one object per country and state. States should be contained by their countries. Countries could be stored in a vector of country objects in the app itself. Another way is to create an instance of the same class we use for countries and states, call it global, and have it store all the countries. The app would then contain the single global object as a property. This option would create a 3-level hierarchy: the global object stores data for the entire world and a vector of country objects, while the objects of countries that have states in the database would store their corresponding states. Again, we can use the same class definition for all three kinds of objects because they store essentially the same kind of data.

2) **Data Collection:**
The data collection subsystem consists of 1) a user-centred web and mobile front-end interface, providing a straightforward and intuitive access to the surveys, and 2) a data collection back-end enabling response aggregation in a consistent and structured format to facilitate post-processing.

2.1) Front-end: Survey Design:

Usability, interaction, and user interfacing play key roles in the initial engagement and subsequent retention of participants. To this end, we pay attention to two main elements: 1) the appearance and usability of the front-end solutions, and 2) the contents and length of the survey.

The mobile survey applications have been designed to have minimal loading times, with lightweight graphical elements, a colour scheme and page layout suitable for all users, including visually impaired participants and participants in geographic locations where internet speeds may be poor. For instance, a tailor-made cache system has been built and deployed to minimize the survey loading time.



Front-end

2.2) Data Aggregation:

The back-end data collection engine was designed to provide seamless aggregation of the data in a consistent and structured format. Timeliness, consistency, and proper dissemination of the data were the three main pillars of the aggregation process. Covid 19 Surveys updates its estimates daily to provide a comparison with the estimates of officially confirmed cases, which are also updated once per day. This daily aggregation also serves as a privacy preserving measure, as we discuss in the next section.

During aggregation, survey responses are classified by country and stored in individual files named as coviddata.mat. Each row in the file corresponds to a single response and is composed of the elements that appear in Table 1: the date of the response, the country for which the response reports, the country ISO code, the region in the country for which the response reports (if any), the region ISO code, the language used to fill the survey, the answers to the survey questions (Var1,…Varn), a cookie that anonymously identifies a participant, and a campaign field that can be used to identify responses that correspond to specific survey dissemination campaigns.

A	B	C	D	E	F	G	H	I	J	K
Date	Country/R	Confirmed	Deaths	Recovered	Active	New cases	New death	New reco	WHO Region	

Table

Data Analysis:

The data has been drawn from “Covid 19 Full Grouped data” and consists of the necessary information to conduct the time series analysis. The variables that are relevant to answer our research question are the dates (2019/12/31–2020–08/01), total

deaths, new deaths and location (INDIA). The data has been cleaned and adjusted to satisfy all the necessary assumptions to use python & R to make the prediction.

The forecast of new deaths for the next 21 and 90 days reaches 18,589 (Total Deaths 171,903) and 82,653 (Total Deaths 235,967) respectively. The result of our projection has been very close when comparing it to CNN's projection. CNN projected on August 2nd that about 19,000 people could die between August 2nd and August 21st in the United States. In addition to that prediction, they also predicted on July 31st in their show "CNN Coronavirus Town Hall" the total numbers of death by November. CNN forecasted 231,000 deaths from Covid19 by November. The results of our Python and R Model are very close when comparing it to CNN's projection.

1) Why & What is Time Series Analysis:

Time series analysis (TSA) is a statistical technique that consists of data points listed in time order. The x axis is made up of equally spaced points in time and the y axis contains the outcome values that are going to be projected from our model based on previous observed values. This technique is suitable for research questions such as forecasting future sales. The reason why time series analysis exists, is since the outcome variable in our model is dependent on one single explanatory variable only: time.

Suppose you run a shoe store and have the data available that tells you how many shoes you have sold in the past years. Given the data available, time series analysis would be applicable if you would like to predict how many shoes your store will sell in the future. In this case, the outcome variable would be the number of shoes sold and the one and only explanatory variable would be time.

Other forecasting algorithms such as linear regression or logistic regression use one or more explanatory variables. Further there is a difference when it comes to the assumptions when comparing linear regression, logistic regression, and the time series technique ARIMA.

In Linear Regression the following assumptions must be met:

- Independence of observations.
- Homoscedasticity of errors (equal variance).
- A linear relationship.
- Errors are normally distributed.

In Logistic Regression the following assumptions have to be met:

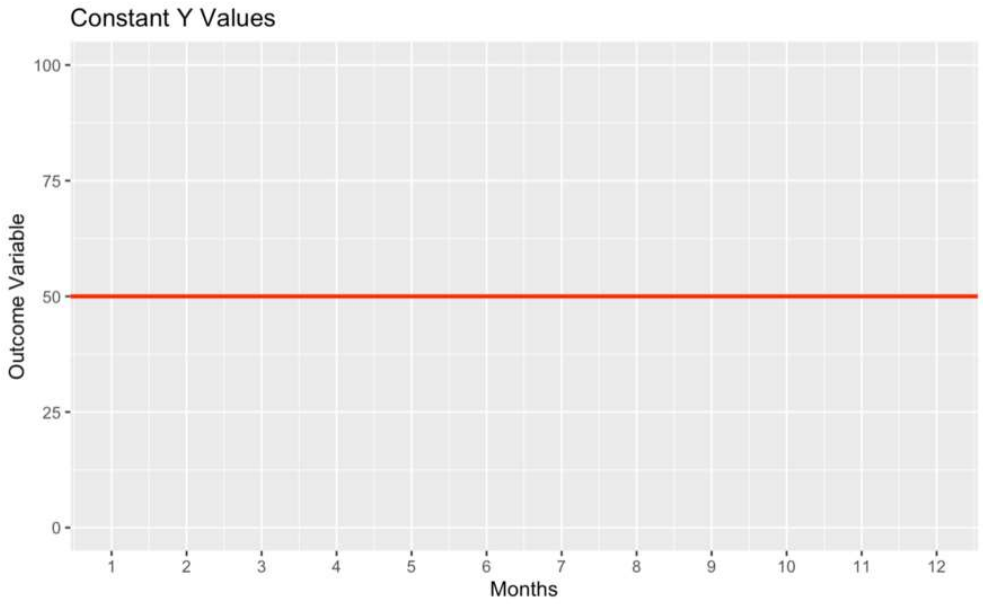
- Dependent variable has to binary.
- Independence of observations.
- Linearity in the logit for continuous variables.
- Lack of influential outliers
- Absence of multicollinearity

In Time Series Analysis ARIMA the following assumptions have to be met:

- Data must be stationary.
- Data should be univariate. As mentioned above TSA ARIMA works on a single variable only.
- Data should be in time series data format.

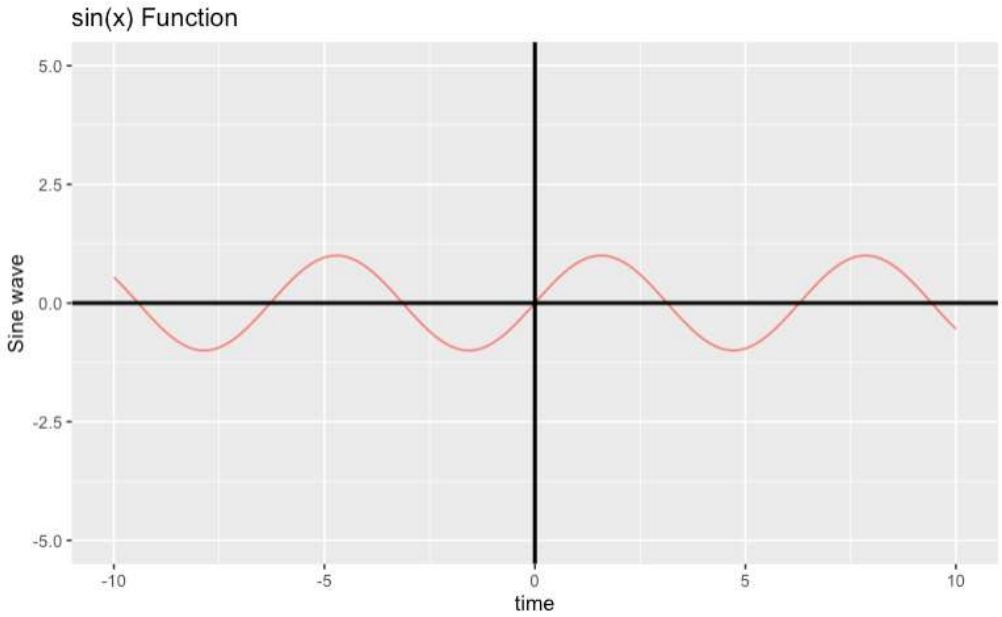
2) **When can we not use Time Series Analysis:**
There are times when time series analysis is not the best statistical technique to answer a research question. There are 3 points that make this technique not suitable.

a. When the data points are always constant.



The x axis has equally spaced points that represent the only variable time and the y values represent the outcome variable. Having constant data points throughout a period really makes this statistical technique not really useful. Consider the shoe store example again. If we sell in all months the same number of shoes, then there is no point of conducting a time series analysis to predict future sales, since the predicted outcome will always be the same.

b. When the data points represent a known function.

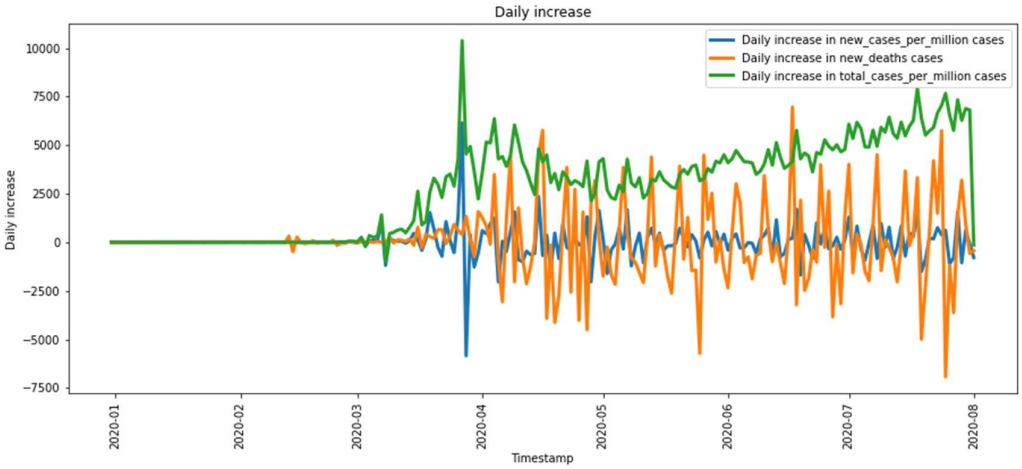


In the graph above we see the sin(x) function. If we add a value to this function, the predicted outcome can be easily computed. Using a time series analysis technique won't be necessary, since we will be able to calculate the predicted value by just plugging in the value into the sin(x) function. This applies for all other functions, such as cos(x) etc. Hence, if the graph of our data points looks like a function then time series analysis doesn't become applicable.

c. When our data is not stationary.

As mentioned above, one of the assumptions for time series analysis (ARIMA) is that the data must be stationary. To have stationary data, the following conditions have to be met:

Mean has to be constant according to the time.
Variance has to be equal in different time intervals from the mean. In other words, the distance of the points should be the same from the mean.
Covariance has also to be equal.

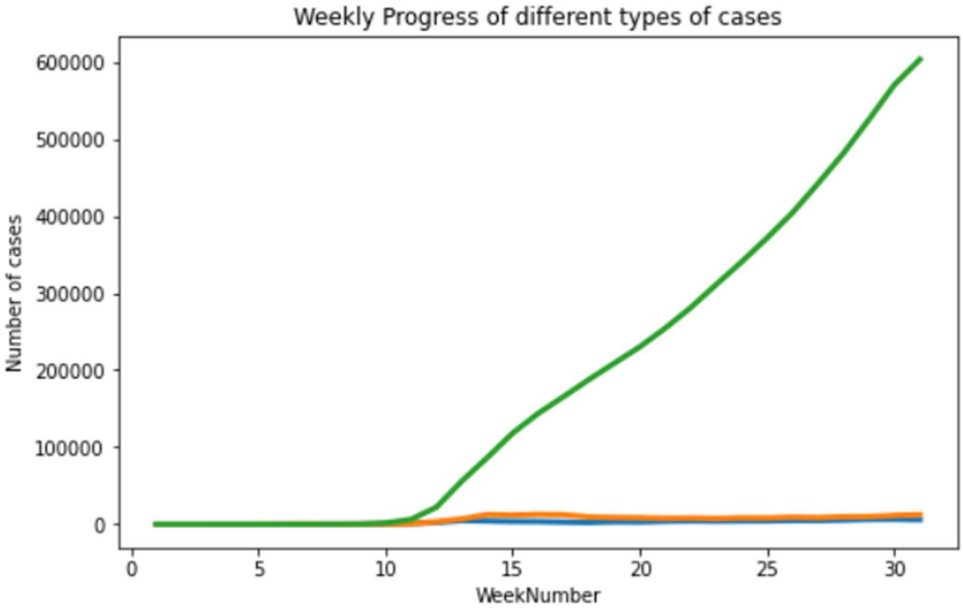


The graph above shows the number of deaths from January — August 1st due to Covid19. The red line displays the mean and as we can see above the mean is constantly increasing. First, we can see that the means aren't constant, therefore the data is violating the assumption for constant mean throughout the time. Secondly, we also see that the variances aren't equal. As we can see above the distance between the points and the mean line varies a lot. Consider the points spiking in mid-April and May. We can clearly see that the distances from the data point to the mean are not the same. Hence the data violates not only the equal mean assumptions, but also the equal variance assumption.

3) Components of Time Series Analysis:

There are three components that we need to understand when it comes to time series analysis.

a. General Trend



b.

b. Seasonality

Seasonality plays a big part when it comes to time series analysis. One example of seasonality is when you see a spike in our data points. Suppose you have the data set of air travellers throughout a year. The data points of that graph would show a big spike in the month of December, due to the Christmas break. On the other hand, we could potentially see a big dip in the month of March and April 2020 where Covid19 just began to spread. This would lead to a major decrease in the number of air travellers.

c. Irregular Fluctuations

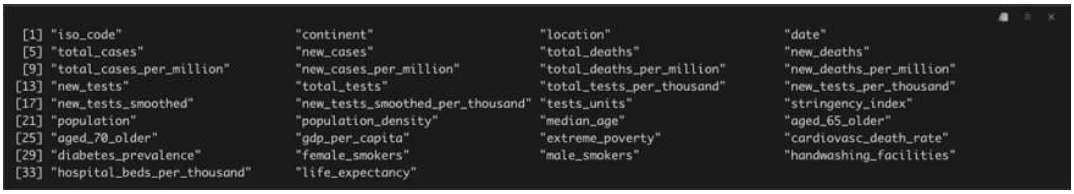
This component is referred to the random component of time series analysis. Irregular fluctuations are uncontrolled situations in which the y values would change. In the air

travellers example, flight cancelations due to weather conditions such as a storm would result in the number of air travellers to decrease. The cancelation of flights due to the storm would be an example of an uncontrolled situation in which the y value would be affected. This wouldn't mean necessarily that there will be a storm at the same time next year. This is where the random effect comes into place.

4) **Demonstration of Time Series Analysis:**
Research Question: How many people are going to die due to Covid19 in the United States from August 1st — August 21st and August 1st — November 1st . We are going to use time series analysis to answer this question. In particular we are going to use the ARIMA model. Time series analysis is the perfect statistical technique to answer this question, since we are going to project how many people are going to lose their lives unfortunately due to the disease Covid19.

Data Source: "Covid 19 Full Grouped csv"

The data consists of 34,033 rows and 34 columns. Below is a list of variables that are in the data.



Collection method: Observations

Data definitions: In order to work on the research question, we had to clean the data and extract the relevant variables that help us conduct the time series analysis. We created a data frame that consisted of the dates (2019/12/31–2020–08/01), total deaths, new deaths and location (INDIA).

"date" variable: Represents the dates that we are going to use for our analysis. Luckily the class of the date column was already date. If the date column wasn't a date, we would have to convert the class to date.

"total_deaths" variable: Represents the total number of Covid19 deaths in the United States. The class of this variable is numeric.

"new_deaths" variable: Represents the number of Covid19 deaths in the United States per day. The class of this variable is numeric.

"location" variable: Represents the name of the country. The class of this variable is factor.

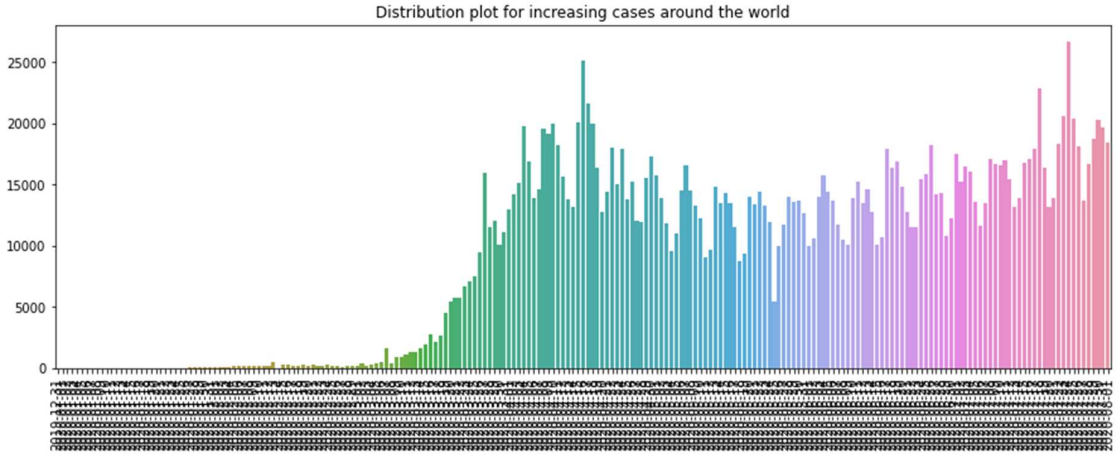
Summary of data post data cleaning:

```
Index
Min. :2019-12-31
1st Qu.:2020-02-22
Median :2020-04-16
Mean :2020-04-16
3rd Qu.:2020-06-08
Max. :2020-08-01

dat_demo

Min. : 0.0
1st Qu.: 0.0
Median : 500.0
Mean : 713.1
3rd Qu.:1167.0
Max. :4928.0
```

When looking at the summary of our data, we can see that our time series index starts from 2019/12/08 and ends in 2020/08/01. Further we can see that the average number of deaths per day is 713 in India. The maximum number of deaths reached to 4928 on April 16.

Time Series Analysis:

When conducting the time series analysis on our data, we first created the date sequence using the zoo library which is a well-known R package when it comes to this particular statistical technique. After that, we check if our data doesn't violate any assumption for time series analysis. We must make sure that the data is stationary. In other words, it needs to have equal mean, variance, and covariance across the different time intervals.



Code: -

1) Application: -

country.m

```
classdef Country
    %UNTITLED2 Summary of this class goes here
    %   Detailed explanation goes here

    properties
        Name
        States State
        StateNameList
        CumulativeCases
        CumulativeDeaths
        DailyCases
        DailyDeaths
    end

    methods
        function obj = Country( name, state, data )
            obj.Name = name;
            obj.StateNameList = state;
            if isempty(state)
                obj.States = '';
                obj.StateNameList = '';
            else
                obj.StateNameList = state(2:end);
                for i = 1:length(obj.StateNameList)
                    obj.States(i) = State(state{i+1},data(i+1,:));
                end
            end
            dataMat = cell2mat(data);
            obj.CumulativeCases = dataMat(1,1:2:end);
            obj.DailyCases = max(diff(dataMat(1,1:2:end),1,2),0);
            obj.CumulativeDeaths = dataMat(1,2:2:end);
            obj.DailyDeaths = max(diff(dataMat(1,2:2:end),1,2),0);

        end
    end

end
```

state.m

```
classdef State
    %UNTITLED5 Summary of this class goes here
    %   Detailed explanation goes here

    properties
        Name
        CumulativeCases
        CumulativeDeaths
        DailyCases
        DailyDeaths
    end

    methods
        function obj = State( name, data )
            obj.Name = name;
            dataMat = cell2mat(data);
            obj.CumulativeCases = dataMat(1:2:end);
            obj.DailyCases = max(diff(dataMat(1:2:end),1,2),0);
            obj.CumulativeDeaths = dataMat(2:2:end);
            obj.DailyDeaths = max(diff(dataMat(2:2:end),1,2),0);
        end

    end

end
```

covid19_app.mlapp

```
classdef covid19_app < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TheVisualizationofCOVID19RelatedDataUIFigure  matlab.ui.Figure
        YASHKHWAHISHSWETAYOGESHLabel  matlab.ui.control.Label
        OptionButtonGroup             matlab.ui.container.ButtonGroup
        DailyButton                   matlab.ui.control.RadioButton
        CumulativeButton              matlab.ui.control.RadioButton
        AverageSlider                 matlab.ui.control.Slider
        AveragedofdaysSliderLabel     matlab.ui.control.Label
        DataofPlotButtonGroup         matlab.ui.container.ButtonGroup
        BothButton                    matlab.ui.control.RadioButton
        DeathsButton                  matlab.ui.control.RadioButton
        CasesButton                   matlab.ui.control.RadioButton
        StateListBox                  matlab.ui.control.ListBox
        StateRegionProvinceLabel      matlab.ui.control.Label
        CountryListBox                matlab.ui.control.ListBox
        CountryLabel                  matlab.ui.control.Label
        UIAxes                        matlab.ui.control.UIAxes
    end


    properties (Access = private)
        Countries Country
        CountryNameList
```

```
        Date
        CurrentCountry
        CurrentState
        DayAverage
        Type
        Option
    end

    methods (Access = private)
        function PlotCurves(app)
            barcolor = [0 0 255]/255;
            linecolor = [255 0 0]/255;
            avr = app.DayAverage;
            tp = app.Type;
            cc = app.CurrentCountry;
            cs = app.CurrentState;
            op = app.Option;

            dname = tp;
            if isequal(tp,'Both')
                dname = 'Cases and Deaths';
            end
            app.UIAxes.Title.String = [op,' Numbers of ',dname ,' in ',cs,'
State(Province) of ', cc, ' (',num2str(avr),'-day mean)'];

            Flag = [strcmp(cc,'All Countries'), strcmp(cs,'all'),
strcmp(tp,{'Cases','Deaths','Both'}), strcmp(op,{'Cumulative','Daily'})];
            code = bin2dec(num2str(Flag));
            indexc = strcmp(app.CountryNameList,cc);
            if sum(indexc) == 0
                indexs = [];
            else
                indexs = strcmp(app.Countries(indexc).StateNameList, cs);
            end
            for i = 1:length(app.CountryNameList)
                YAA(i,:) = app.Countries(i).DailyCases;
                SHH(i,:) = app.Countries(i).DailyDeaths;
                SRR(i,:) = app.Countries(i).CumulativeCases;
                VAS(i,:) = app.Countries(i).CumulativeDeaths;
            end
            [r,c] = size(SRR);
            cases = [];
            deaths = [];
            switch code
                case {5}
                    cases = movmean(app.Countries(indexc).States(indexs).DailyCases,
avr,'Endpoints','discard');
                    deaths = movmean(app.Countries(indexc).States(indexs).DailyDeaths,
avr,'Endpoints','discard');
                case {37}
                    cases = movmean(app.Countries(indexc).DailyCases,
avr,'Endpoints','discard');
                    deaths = movmean(app.Countries(indexc).DailyDeaths,
avr,'Endpoints','discard');
                case {69,101}
                    cases = movmean(sum(YAA,1), avr,'Endpoints','discard');
                    deaths = movmean(sum(SHH,1), avr,'Endpoints','discard');
                case {6}
                    cases = movmean(app.Countries(indexc).States(indexs).CumulativeCases,
avr,'Endpoints','discard');
```

```
                    deaths =
movmean(app.Countries(indexc).States(indexs).CumulativeDeaths, avr,'Endpoints','discard');
                case {38}
                    cases = movmean(app.Countries(indexc).CumulativeCases,
avr,'Endpoints','discard');
                    deaths = movmean(app.Countries(indexc).CumulativeDeaths,
avr,'Endpoints','discard');
                case {70,102}
                    cases = movmean(sum(SRR,1), avr,'Endpoints','discard');
                    deaths = movmean(sum(VAS,1), avr,'Endpoints','discard');
                case {9}
                    deaths = movmean(app.Countries(indexc).States(indexs).DailyDeaths,
avr,'Endpoints','discard');
                case {41}
                    deaths = movmean(app.Countries(indexc).DailyDeaths,
avr,'Endpoints','discard');
                case {73,105}
                    deaths = movmean(sum(SHH,1), avr,'Endpoints','discard');
                case {10}
                    deaths =
movmean(app.Countries(indexc).States(indexs).CumulativeDeaths, avr,'Endpoints','discard');
                case {42}
                    deaths = movmean(app.Countries(indexc).CumulativeDeaths,
avr,'Endpoints','discard');
                case {74,106}
                    deaths = movmean(sum(SRR,1), avr,'Endpoints','discard');
                case {17}
                    cases = movmean(app.Countries(indexc).States(indexs).DailyCases,
avr,'Endpoints','discard');
                case {49}
                    cases = movmean(app.Countries(indexc).DailyCases,
avr,'Endpoints','discard');
                case {81,113}
                    cases = movmean(sum(YAA,1), avr,'Endpoints','discard');
                case {18}
                    cases = movmean(app.Countries(indexc).States(indexs).CumulativeCases,
avr,'Endpoints','discard');
                case {50}
                    cases = movmean(app.Countries(indexc).CumulativeCases,
avr,'Endpoints','discard');
                case {82,114}
                    cases = movmean(sum(SRR,1), avr,'Endpoints','discard');
            end
            if ~isempty(cases) && ~isempty(deaths)
                yyaxis(app.UIAxes, 'left');
                bar(app.UIAxes,cases, 0.5,'FaceColor',barcolor);
                app.UIAxes.YColor = barcolor;
                yyaxis(app.UIAxes, 'right');
                plot(app.UIAxes,deaths,'LineWidth',2,'Color',linecolor);
                app.UIAxes.YColor = linecolor;
                app.UIAxes.XLim = [1,length(cases)];
            else
                if isempty(deaths)
                    yyaxis(app.UIAxes, 'left');
                    bar(app.UIAxes,cases, 0.5,'FaceColor',barcolor);
                    app.UIAxes.YColor = barcolor;
                    yyaxis(app.UIAxes, 'right');
                    cla(app.UIAxes);
                    app.UIAxes.XLim = [1,length(cases)];
                else
                    yyaxis(app.UIAxes, 'right');
```

```
                plot(app.UIAxes,deaths,'LineWidth',2,'Color',linecolor);
                app.UIAxes.YColor = linecolor;
                yyaxis(app.UIAxes, 'left');
                cla(app.UIAxes);
                app.UIAxes.XLim = [1,length(deaths)];
            end
        end
        app.UIAxes.YLim = [0,inf];
        lx = length(app.UIAxes.XTick);
        ld = length(app.Date);
        cut = ceil((avr-1)/2);
        app.UIAxes.XTickLabel = app.Date(1+cut:ceil(ld/lx):end-cut);
        app.UIAxes.XTickLabelRotation = 60;
        app.UIAxes.FontWeight = 'bold';
    end
end


% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)
        load("covid_data.mat");
        Data = covid_data;
        %             data = cell2mat(Data(2:end,3:end));
        app.Date = Data(1,3:end);
        app.CountryNameList = unique(Data(2:end,1));
        for i = 1:length(app.CountryNameList)
            name = app.CountryNameList{i};
            index = strcmp(name,Data(:,1));
            app.Countries(i) = Country(name, Data(index,2), Data(index,3:end));
        end
        app.CountryListBox.Items = [ 'All Countries'; app.CountryNameList ];
        app.CurrentCountry = 'All Countries';
        app.StateListBox.Items = {'all'};
        app.CurrentState = 'all';
        app.Type = 'Cases';
        app.Option = 'Cumulative';
        app.DayAverage = 1;
        PlotCurves(app);
    end

    % Value changing function: AverageSlider
    function AverageSliderValueChanging(app, event)
        app.DayAverage = round(event.Value);
        PlotCurves(app);
    end

    % Value changed function: CountryListBox
    function CountryListBoxValueChanged(app, event)
        app.CurrentCountry = app.CountryListBox.Value;

        indexc = strcmp(app.CountryNameList,app.CurrentCountry);
        if sum(indexc) == 0
            app.StateListBox.Items = {'all'};
        else
            app.StateListBox.Items = ['all'; app.Countries(indexc).StateNameList];
        end
        app.CurrentState = 'all';
        PlotCurves(app);
```

```
        end

        % Selection changed function: DataofPlotButtonGroup
        function DataofPlotButtonGroupSelectionChanged(app, event)
            app.Type = app.DataofPlotButtonGroup.SelectedObject.Text;
            PlotCurves(app);
        end

        % Selection changed function: OptionButtonGroup
        function OptionButtonGroupSelectionChanged(app, event)
            app.Option = app.OptionButtonGroup.SelectedObject.Text;
            PlotCurves(app);
        end

        % Value changed function: StateListBox
        function StateListBoxValueChanged(app, event)
            app.CurrentState = app.StateListBox.Value;
            PlotCurves(app);
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create TheVisualizationofCOVID19RelatedDataUIFigure and hide until all
components are created
            app.TheVisualizationofCOVID19RelatedDataUIFigure = uifigure('Visible', 'off');
            app.TheVisualizationofCOVID19RelatedDataUIFigure.Color = [0.902 0.902 0.902];
            app.TheVisualizationofCOVID19RelatedDataUIFigure.Position = [100 100 775 580];
            app.TheVisualizationofCOVID19RelatedDataUIFigure.Name = 'The Visualization of
COVID-19 Related Data';

            % Create UIAxes
            app.UIAxes = uiaxes(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            title(app.UIAxes, 'Daily Numbers of Cases and Deaths in China (7-day mean)')
            app.UIAxes.XTickLabelRotation = 0;
            app.UIAxes.YTickLabelRotation = 0;
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.GridAlpha = 0.15;
            app.UIAxes.MinorGridAlpha = 0.25;
            app.UIAxes.Box = 'on';
            app.UIAxes.Position = [29 224 712 343];

            % Create CountryLabel
            app.CountryLabel = uilabel(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.CountryLabel.HorizontalAlignment = 'right';
            app.CountryLabel.VerticalAlignment = 'top';
            app.CountryLabel.FontWeight = 'bold';
            app.CountryLabel.Position = [48 183 58 15];
            app.CountryLabel.Text = 'Country: ';

            % Create CountryListBox
            app.CountryListBox =
uilistbox(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.CountryListBox.Items = {'Empty'};
            app.CountryListBox.ValueChangedFcn = createCallbackFcn(app,
@CountryListBoxValueChanged, true);
```

```
            app.CountryListBox.FontWeight = 'bold';
            app.CountryListBox.BackgroundColor = [0.9412 0.9412 0.9412];
            app.CountryListBox.Position = [52 25 159 152];
            app.CountryListBox.Value = 'Empty';

            % Create StateRegionProvinceLabel
            app.StateRegionProvinceLabel =
uilabel(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.StateRegionProvinceLabel.BackgroundColor = [0.902 0.902 0.902];
            app.StateRegionProvinceLabel.HorizontalAlignment = 'right';
            app.StateRegionProvinceLabel.VerticalAlignment = 'top';
            app.StateRegionProvinceLabel.FontWeight = 'bold';
            app.StateRegionProvinceLabel.Position = [241 183 139 15];
            app.StateRegionProvinceLabel.Text = 'State\Region\Province: ';

            % Create StateListBox
            app.StateListBox =
uilistbox(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.StateListBox.Items = {'Empty'};
            app.StateListBox.ValueChangedFcn = createCallbackFcn(app,
@StateListBoxValueChanged, true);
            app.StateListBox.FontWeight = 'bold';
            app.StateListBox.BackgroundColor = [0.902 0.902 0.902];
            app.StateListBox.Position = [241 25 159 152];
            app.StateListBox.Value = 'Empty';

            % Create DataofPlotButtonGroup
            app.DataofPlotButtonGroup =
uibuttongroup(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.DataofPlotButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@DataofPlotButtonGroupSelectionChanged, true);
            app.DataofPlotButtonGroup.Title = 'Data of  Plot';
            app.DataofPlotButtonGroup.BackgroundColor = [0.902 0.902 0.902];
            app.DataofPlotButtonGroup.FontWeight = 'bold';
            app.DataofPlotButtonGroup.Position = [441 25 145 106];

            % Create CasesButton
            app.CasesButton = uiradiobutton(app.DataofPlotButtonGroup);
            app.CasesButton.Text = 'Cases';
            app.CasesButton.Position = [11 60 57 15];
            app.CasesButton.Value = true;

            % Create DeathsButton
            app.DeathsButton = uiradiobutton(app.DataofPlotButtonGroup);
            app.DeathsButton.Text = 'Deaths';
            app.DeathsButton.Position = [11 38 61 15];

            % Create BothButton
            app.BothButton = uiradiobutton(app.DataofPlotButtonGroup);
            app.BothButton.Text = 'Both';
            app.BothButton.Position = [11 16 47 15];

            % Create AveragedofdaysSliderLabel
            app.AveragedofdaysSliderLabel =
uilabel(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.AveragedofdaysSliderLabel.HorizontalAlignment = 'right';
            app.AveragedofdaysSliderLabel.VerticalAlignment = 'top';
            app.AveragedofdaysSliderLabel.FontWeight = 'bold';
            app.AveragedofdaysSliderLabel.Position = [435 183 116 15];
            app.AveragedofdaysSliderLabel.Text = 'Averaged # of days';
```

```
            % Create AverageSlider
            app.AverageSlider =
uislider(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.AverageSlider.Limits = [1 15];
            app.AverageSlider.ValueChangingFcn = createCallbackFcn(app,
@AverageSliderValueChanging, true);
            app.AverageSlider.MinorTicks = [];
            app.AverageSlider.FontWeight = 'bold';
            app.AverageSlider.Position = [447 173 278 3];
            app.AverageSlider.Value = 1;

            % Create OptionButtonGroup
            app.OptionButtonGroup =
uibuttongroup(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.OptionButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@OptionButtonGroupSelectionChanged, true);
            app.OptionButtonGroup.Title = 'Option';
            app.OptionButtonGroup.BackgroundColor = [0.902 0.902 0.902];
            app.OptionButtonGroup.FontWeight = 'bold';
            app.OptionButtonGroup.Position = [585 25 145 106];

            % Create CumulativeButton
            app.CumulativeButton = uiradiobutton(app.OptionButtonGroup);
            app.CumulativeButton.Text = 'Cumulative';
            app.CumulativeButton.Position = [11 60 84 15];
            app.CumulativeButton.Value = true;

            % Create DailyButton
            app.DailyButton = uiradiobutton(app.OptionButtonGroup);
            app.DailyButton.Text = 'Daily';
            app.DailyButton.Position = [11 38 50 15];

            % Create YASHKHWAHISHSWETAYOGESHLabel
            app.YASHKHWAHISHSWETAYOGESHLabel =
uilabel(app.TheVisualizationofCOVID19RelatedDataUIFigure);
            app.YASHKHWAHISHSWETAYOGESHLabel.Position = [276 1 234 22];
            app.YASHKHWAHISHSWETAYOGESHLabel.Text = 'YASH || KHWAHISH || SWETA|| YOGESH';

            % Show the figure after all components are created
            app.TheVisualizationofCOVID19RelatedDataUIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = covid19_app

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TheVisualizationofCOVID19RelatedDataUIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
```

```
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TheVisualizationofCOVID19RelatedDataUIFigure)
        end
    end
end
```

2) Time Series Analysis: -

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
  for filename in filenames:
    print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import datetime as dt
from datetime import timedelta
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from statsmodels.tsa.api import Holt
```

```
covid = pd.read_csv("../input/covid-19-data/owid-covid-data.csv")
covid.head(10)
```

```
print("Size/Shape of the dataset",covid.shape)
print("Checking for null values",covid.isnull().sum())
print("Checking Data-type",covid.dtypes)
```

```
covid.isnull().sum()
```

```
covid["date"] = pd.to_datetime(covid["date"])
```

In [17]:

```
datewise =
covid.groupby(["date"]).agg({"total_cases_per_million":"sum","new_deaths":"sum","new_cases_per_million":"sum"})
```

```
print("Basic Information")
print("Total number of Confirmed cases around the world",datewise["total_cases_per_million"].iloc[-1])
print("Total number of Recovered cases around the world",datewise["new_deaths"].iloc[-1])
print("Total number of Death cases around the world",datewise["new_cases_per_million"].iloc[-1])
print("Total number of Active cases around the world",(datewise["total_cases_per_million"].iloc[-1]-
datewise["new_deaths"].iloc[-1]-datewise["new_cases_per_million"].iloc[-1]))
print("Total number of increasing cases around the world",(datewise["new_cases_per_million"].iloc[-
1]+datewise["new_deaths"].iloc[-1]))
```

```
plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["total_cases_per_million"]-datewise["new_deaths"]-
datewise["new_cases_per_million"])
plt.title("Distributions plot for Active Cases")
plt.xticks(rotation=90)
```

```
plt.figure(figsize=(15,5))
sns.barplot(x=datewise.index.date,y=datewise["new_cases_per_million"]+datewise["new_deaths"])
plt.title("Distribution plot for increasing cases around the world")
plt.xticks(rotation=90)
```

```
datewise["WeekofYear"] = datewise.index.weekofyear
week_num = []
weekwise_new_cases_per_million = []
weekwise_new_deaths = []
weekwise_total_cases_per_million = []
w = 1
for i in list(datewise["WeekofYear"].unique()):
  weekwise_new_cases_per_million.append(datewise[datewise["WeekofYear"]==i]["new_cases_per_million"].iloc[-
1])
  weekwise_new_deaths.append(datewise[datewise["WeekofYear"]==i]["new_deaths"].iloc[-1])
  weekwise_total_cases_per_million.append(datewise[datewise["WeekofYear"]==i]["total_cases_per_million"].iloc[-
1])
  week_num.append(w)
  w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num,weekwise_new_cases_per_million,linewidth=3)
plt.plot(week_num,weekwise_new_deaths,linewidth =3)
plt.plot(week_num,weekwise_total_cases_per_million,linewidth = 3)
plt.xlabel("WeekNumber")
plt.ylabel("Number of cases")
plt.title("Weekly Progress of different types of cases")
```

```
ig,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))
sns.barplot(x= week_num,y=pd.Series(weekwise_new_deaths).diff().fillna(0),ax=ax1)
sns.barplot(x= week_num,y=pd.Series(weekwise_new_cases_per_million).diff().fillna(0),ax=ax2)
ax1.set_xlabel("Week Number")
ax2.set_xlabel("Week Number")
ax1.set_ylabel("Numberof new_cases_per_million cases")
ax2.set_ylabel("Numberof new_deaths cases")
ax1.set_title("Weekly increase in number of new_cases_per_million cases")
```

```
ax2.set_title("Weekly increase in number of new_deaths Cases")
plt.show()
```

```
print("Average increase in number of new_cases_per_million cases
everyday:",np.round(datewise["new_cases_per_million"].diff().fillna(0).mean()))
print("Average increase in number of new_deaths cases
everyday:",np.round(datewise["new_deaths"].diff().fillna(0).mean()))
print("Average increase in number of total_cases_per_million cases
everyday:",np.round(datewise["total_cases_per_million"].diff().fillna(0).mean()))

plt.figure(figsize=(15,6))
plt.plot(datewise["new_cases_per_million"].diff().fillna(0),label="Daily increase in new_cases_per_million
cases",linewidth=3)
plt.plot(datewise["new_deaths"].diff().fillna(0),label="Daily increase in new_deaths cases",linewidth=3)
plt.plot(datewise["total_cases_per_million"].diff().fillna(0),label="Daily increase in total_cases_per_million
cases",linewidth=3)
plt.xlabel("Timestamp")
plt.ylabel("Daily increase")
plt.title("Daily increase")
plt.legend()
plt.xticks(rotation=90)
plt.show()
```

```
countrywise=
covid[covid["date"]==covid["date"].max()].groupby(["location"]).agg({"new_cases_per_million":"sum","new_deaths
":"sum","total_cases_per_million":"sum"}).sort_values(["new_cases_per_million"],ascending=False)
countrywise["New_cases"]=(countrywise["new_cases_per_million"]/countrywise["total_cases_per_million"])*100
countrywise["New_deaths"]=(countrywise["new_deaths"]/countrywise["total_cases_per_million"])*100
```

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(25,10))
top_15confirmed = countrywise.sort_values(["new_cases_per_million"],ascending=False).head(15)
top_15deaths = countrywise.sort_values(["new_deaths"],ascending=False).head(15)
sns.barplot(x=top_15confirmed["new_cases_per_million"],y=top_15confirmed.index,ax=ax1)
ax1.set_title("Top 15 countries as per number of confirmed cases")
sns.barplot(x=top_15deaths["new_deaths"],y=top_15deaths.index,ax=ax2)
ax1.set_title("Top 15 countries as per number of death cases")
```

```
india_data = covid[covid["location"]=="India"]
datewise_india =
india_data.groupby(["date"]).agg({"new_cases_per_million":"sum","new_deaths":"sum","total_cases_per_million":"su
m"})
print(datewise_india.iloc[-1])
print("Total Active Cases",datewise_india["new_cases_per_million"].iloc[-1]-datewise_india["new_deaths"].iloc[-1]-
datewise_india["new_cases_per_million"].iloc[-1])
```

```
datewise_india["WeekofYear"] = datewise_india.index.weekofyear
week_num_india = []
india_weekwise_new_cases_per_million = []
india_weekwise_new_deaths = []
india_weekwise_total_cases_per_million = []
w = 1
for i in list(datewise_india["WeekofYear"].unique()):
```

```
india_weekwise_new_cases_per_million.append(datewise_india[datewise_india["WeekofYear"]==i]["new_cases_per
_million"].iloc[-1])
   india_weekwise_new_deaths.append(datewise_india[datewise_india["WeekofYear"]==i]["new_deaths"].iloc[-1])

india_weekwise_total_cases_per_million.append(datewise_india[datewise_india["WeekofYear"]==i]["new_deaths"].i
loc[-1])
   week_num_india.append(w)
   w=w+1
plt.figure(figsize=(8,5))
plt.plot(week_num_india,india_weekwise_new_cases_per_million,linewidth=3)
plt.plot(week_num_india,india_weekwise_new_deaths,linewidth =3)
plt.plot(week_num_india,india_weekwise_total_cases_per_million,linewidth = 3)
plt.xlabel("WeekNumber")
```

```
max_ind = datewise_india["total_cases_per_million"].max()
print("It took",datewise_india[datewise_india["total_cases_per_million"]>0].shape[0],"days in India to
reach",max_ind,"Confirmed Cases")
```

```
datewise["Days Since"]=datewise.index-datewise.index[0]
datewise["Days Since"] = datewise["Days Since"].dt.days
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[:int(datewise.shape[0]*0.95):]
model_scores=[]
```

```
lin_reg = LinearRegression(normalize=True)
svm = SVR(C=1,degree=5,kernel='poly',epsilon=0.001)
lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["total_cases_per_million"]).reshape(-
1,1))
svm.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(train_ml["total_cases_per_million"]).reshape(-1,1))
```

```
prediction_valid_lin_reg = lin_reg.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
prediction_valid_svm = svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))
```

```
new_date = []
new_prediction_lr=[]
new_prediction_svm=[]
for i in range(1,18):
  new_date.append(datewise.index[-1]+timedelta(days=i))
  new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0][0])
  new_prediction_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0])
pd.set_option("display.float_format",lambda x: '%.f' % x)
model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_svm),columns =
["Dates","LR","SVR"])
model_predictions.head(5)
```

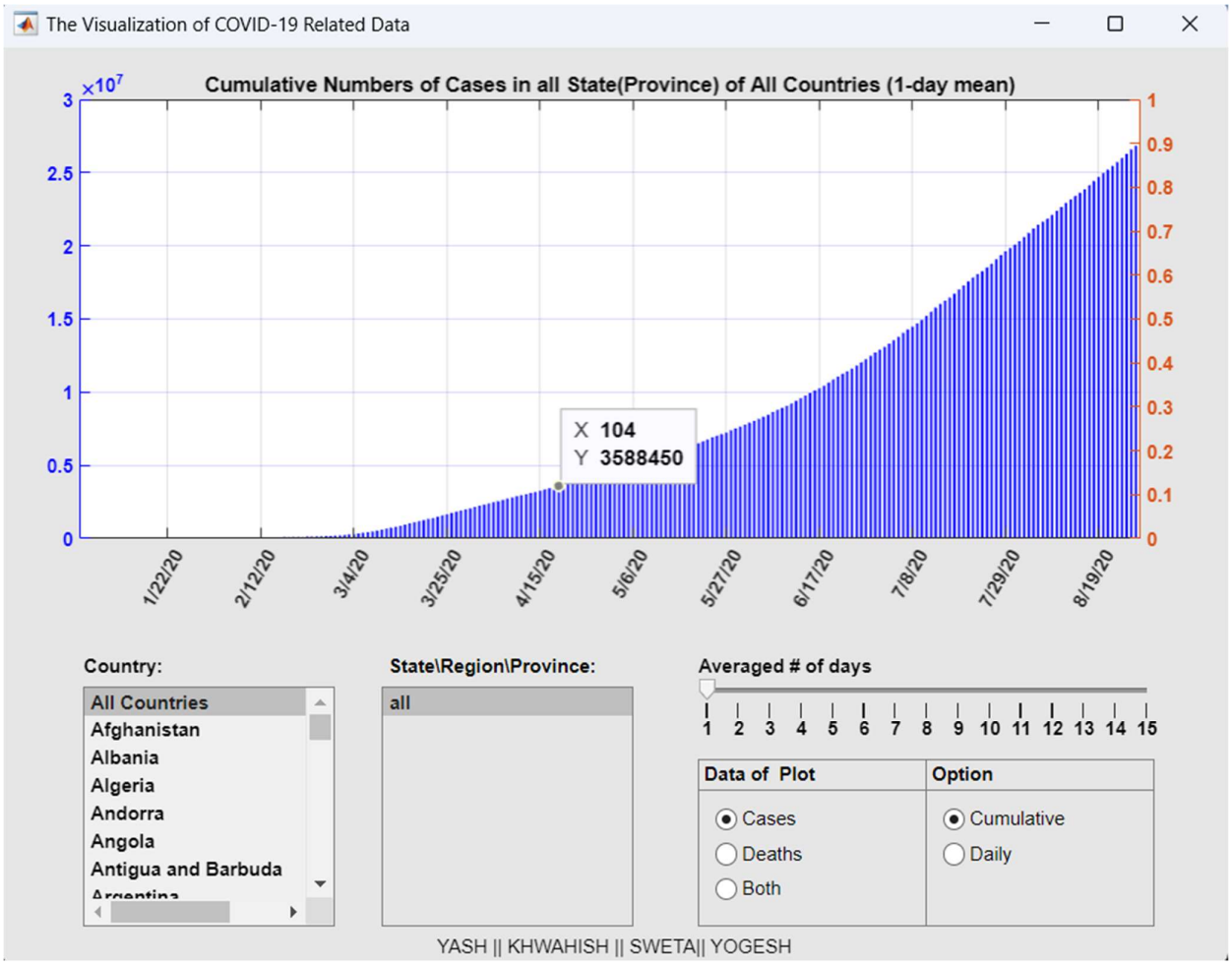
```
model_train=datewise.iloc[:int(datewise.shape[0]*0.85)]
valid=datewise.iloc[int(datewise.shape[0]*0.85):]
```

```
holt=Holt(np.asarray(model_train["total_cases_per_million"])).fit(smoothing_level=1.4,smoothing_slope=0.2)
y_pred = valid.copy()
```

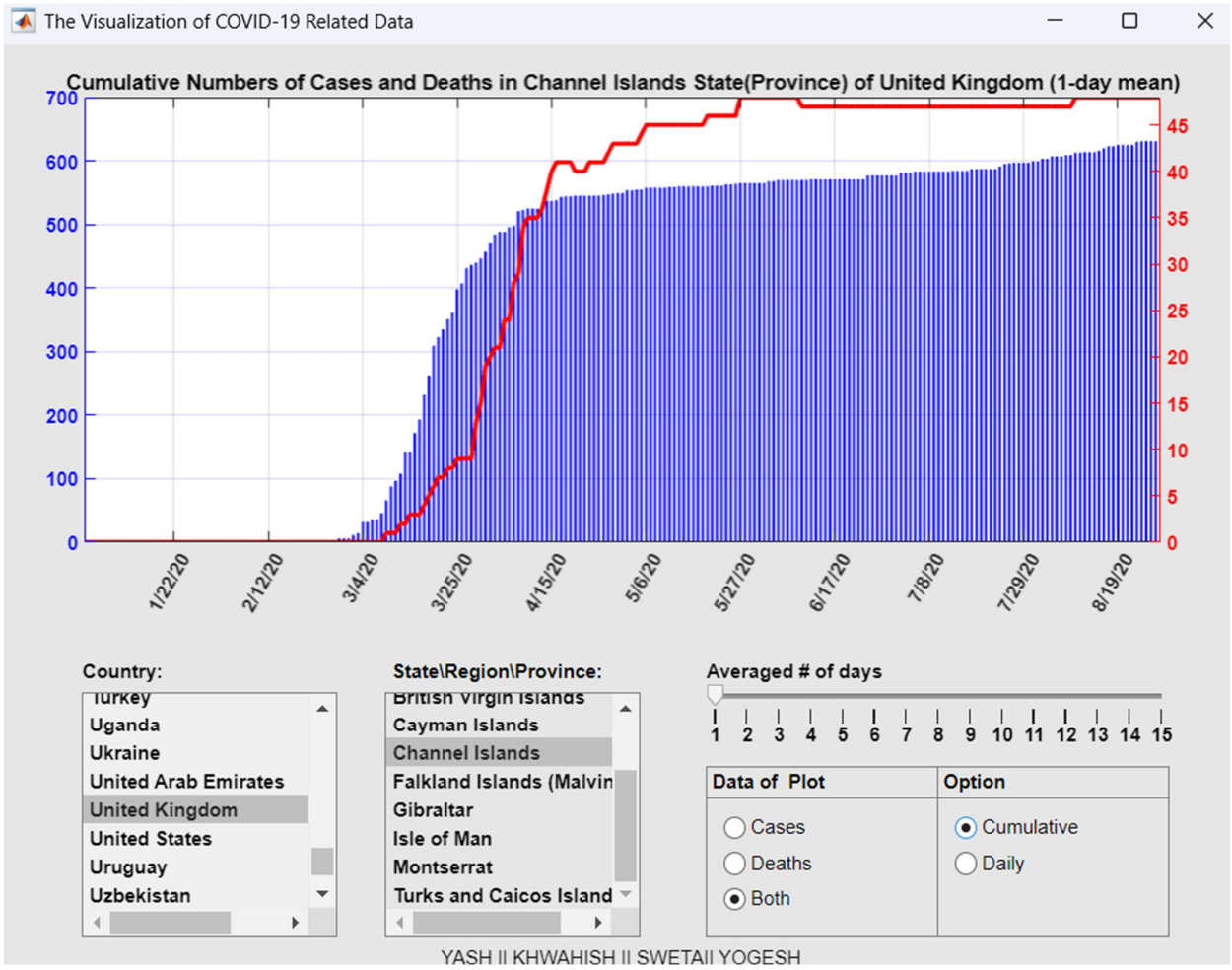
```
y_pred["Holt"]=holt.forecast(len(valid))
```

```
holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
  holt_new_date.append(datewise.index[-1]+timedelta(days=i))
  holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holts Linear Model Prediction"]=holt_new_prediction
model_predictions.head()
```

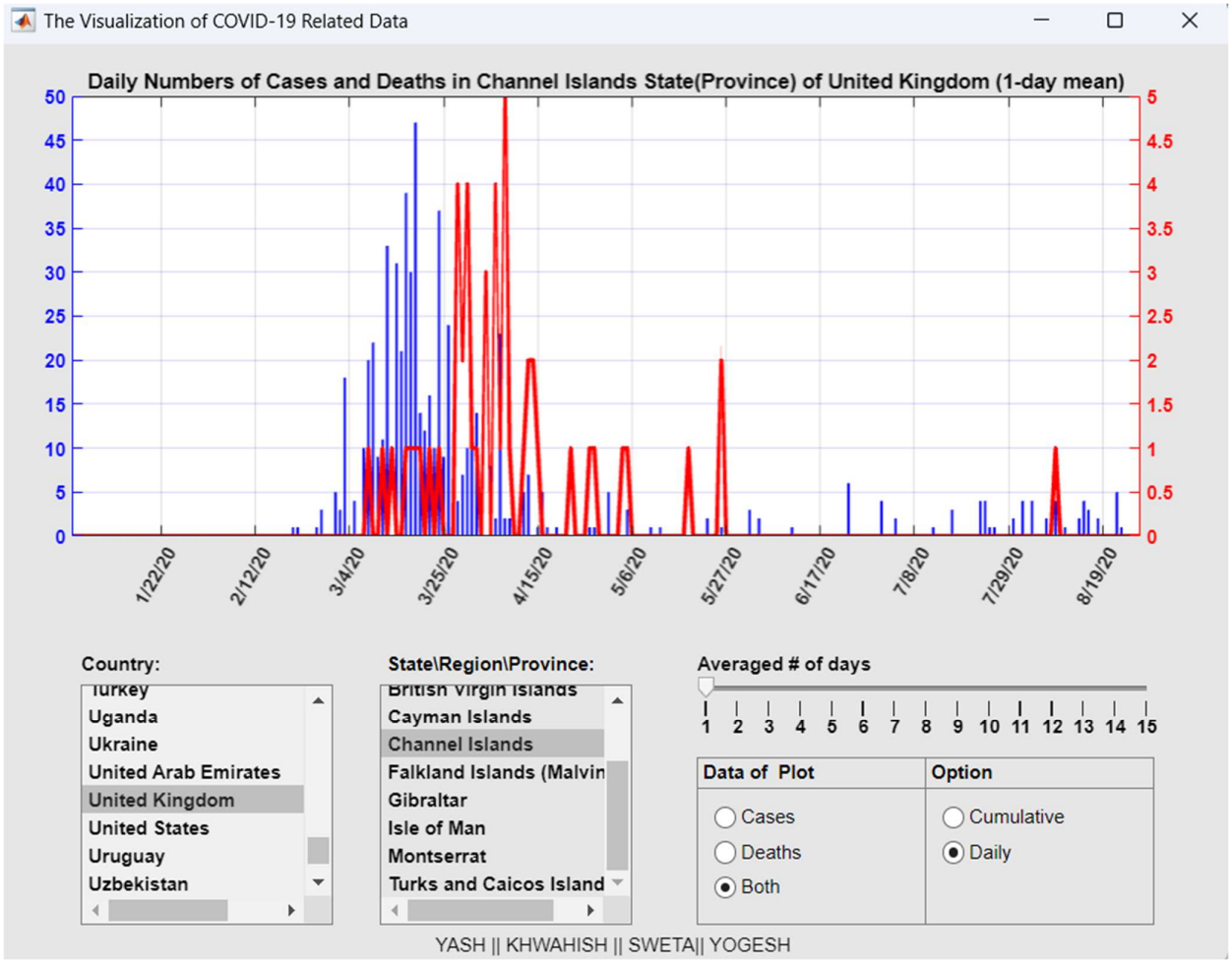
OUTPUTS:



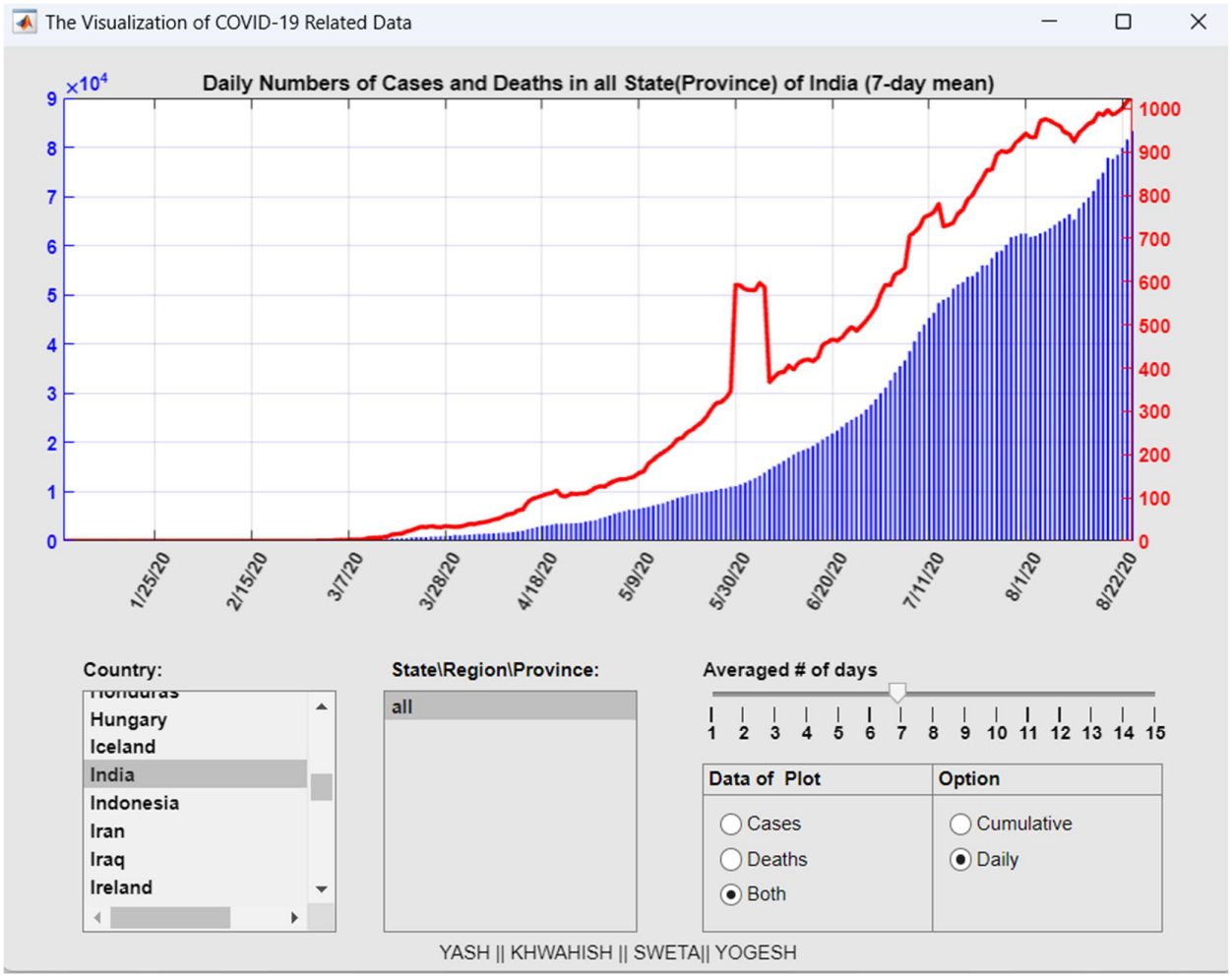
On Opening App



Cumulative Data of Different countries and their states



Daily Data of Different countries and their states

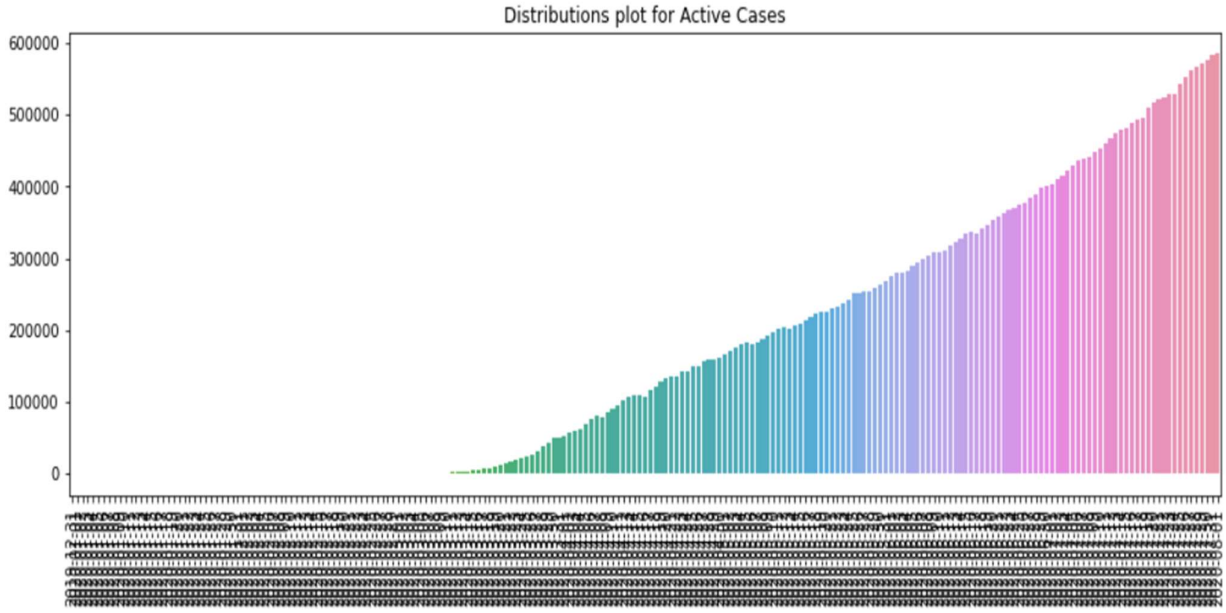


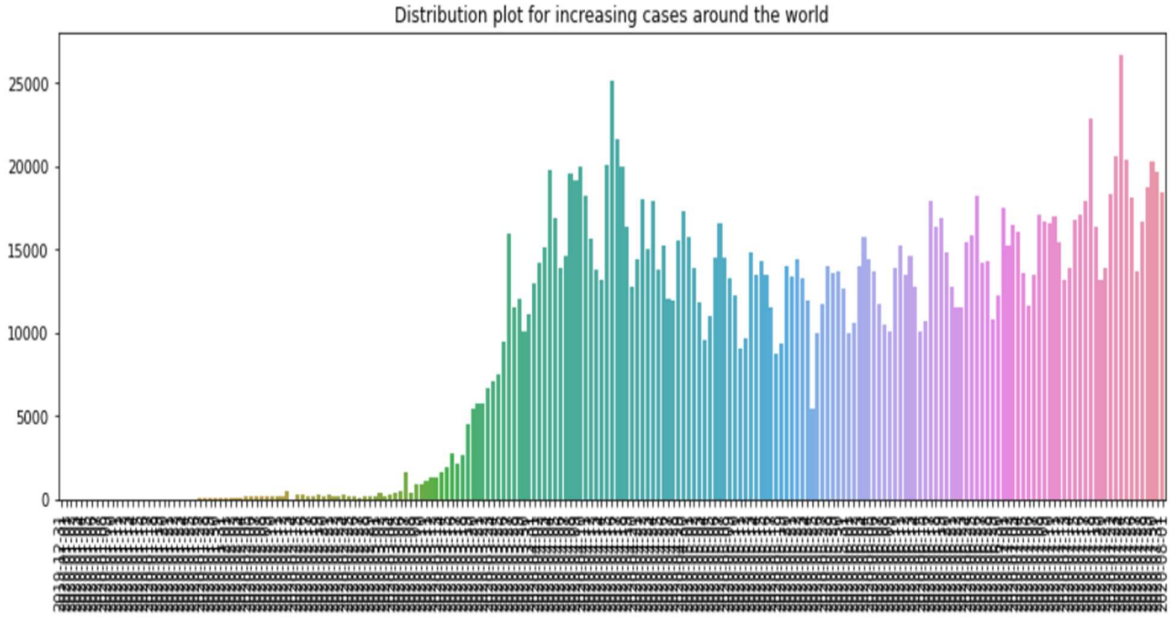
7 day mean data of cases and deaths in India

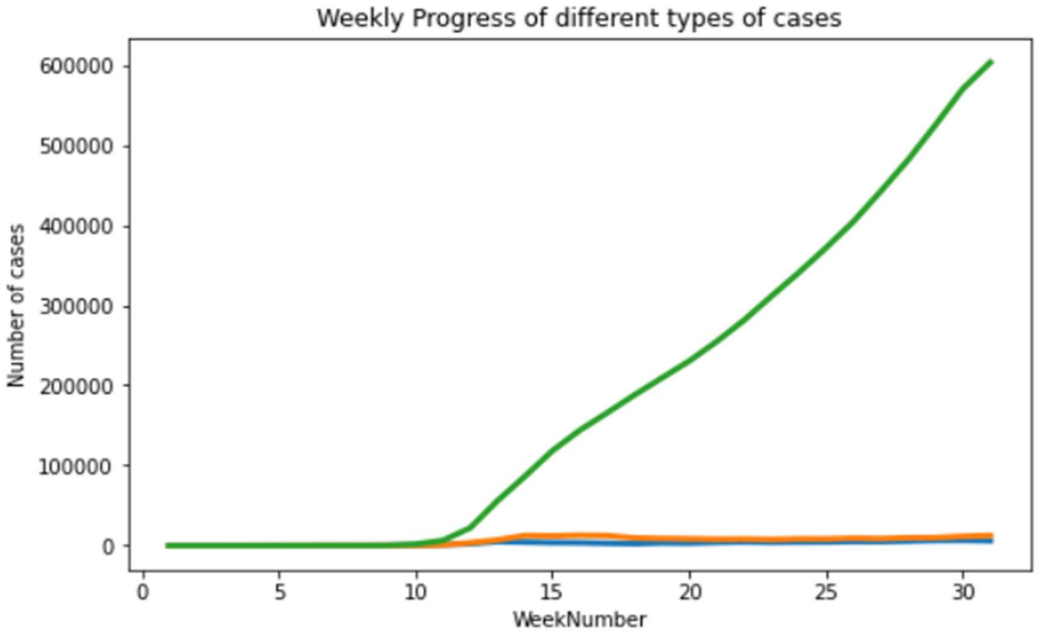
	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million	...	aged_70_older	gdp_per_capita	extreme_poverty	cardiovasc_death_rate	diabetes_prevale
0	AFG	Asia	Afghanistan	2019-12-31	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
1	AFG	Asia	Afghanistan	2020-01-01	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
2	AFG	Asia	Afghanistan	2020-01-02	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
3	AFG	Asia	Afghanistan	2020-01-03	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
4	AFG	Asia	Afghanistan	2020-01-04	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
5	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
6	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
7	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
8	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	
9	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	0.0	0.0	0.0	0.0	...	1.337	1803.987	NaN	597.029	

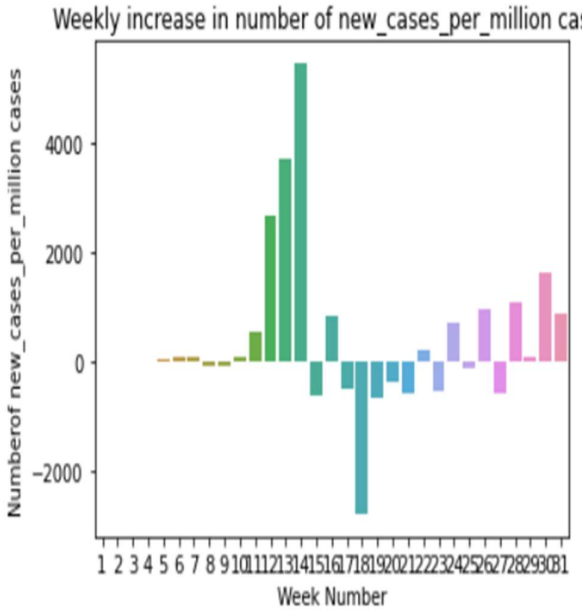
10 rows × 34 columns

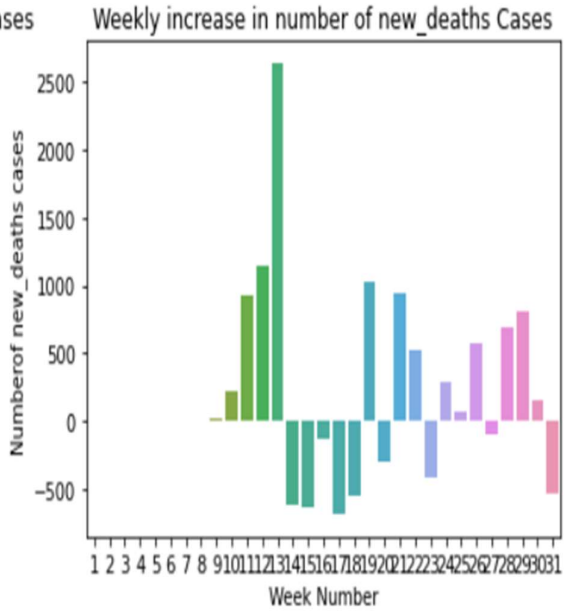
covid.head(10)


Distributions plot for Active Cases
600000
500000
400000
300000
200000
100000
0

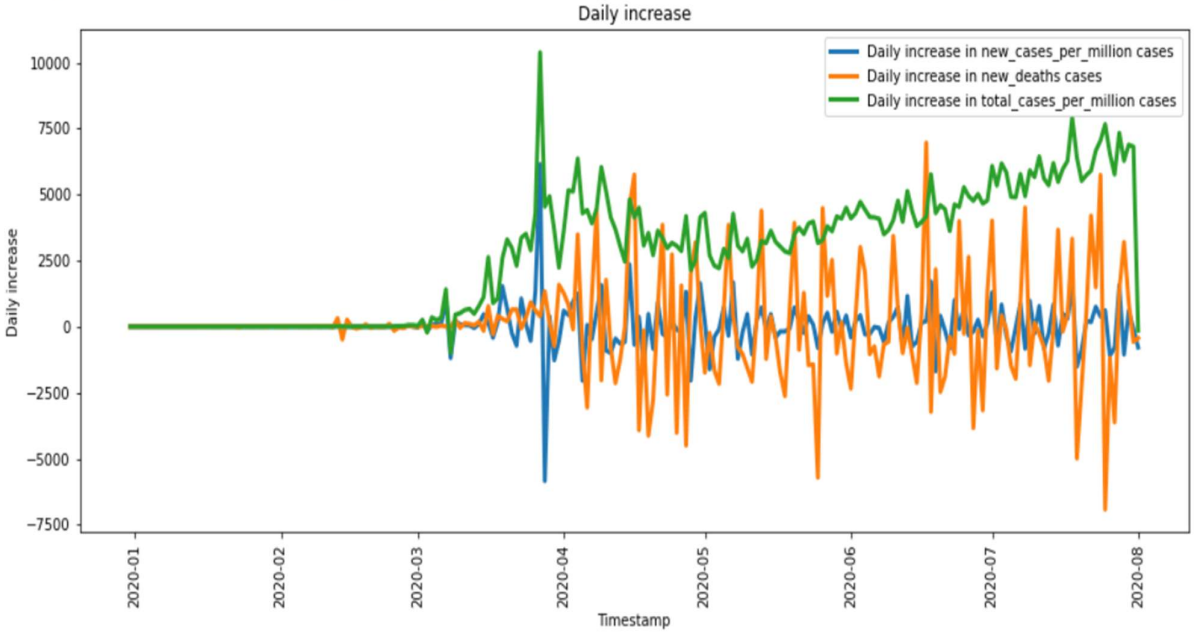

Distribution plot for increasing cases around the world
25000
20000
15000
10000
5000
0


Weekly Progress of different types of cases
Number of cases
WeekNumber
600000
500000
400000
300000
200000
100000
0
0
5
10
15
20
25
30

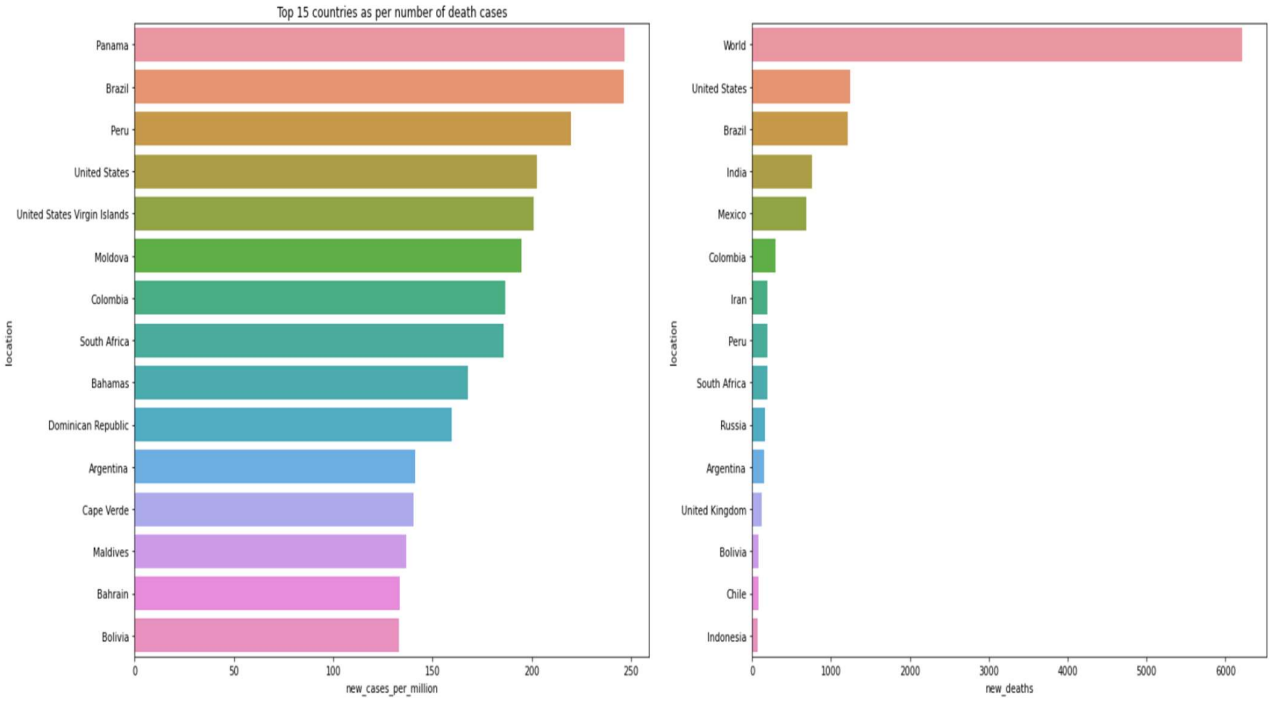

Weekly increase in number of new_cases_per_million cases
Numberof new_cases_per_million cases
Week Number
4000
2000
0
−2000


Weekly increase in number of new_deaths Cases
Numberof new_deaths cases
Week Number
2500
2000
1500
1000
500
0
−500

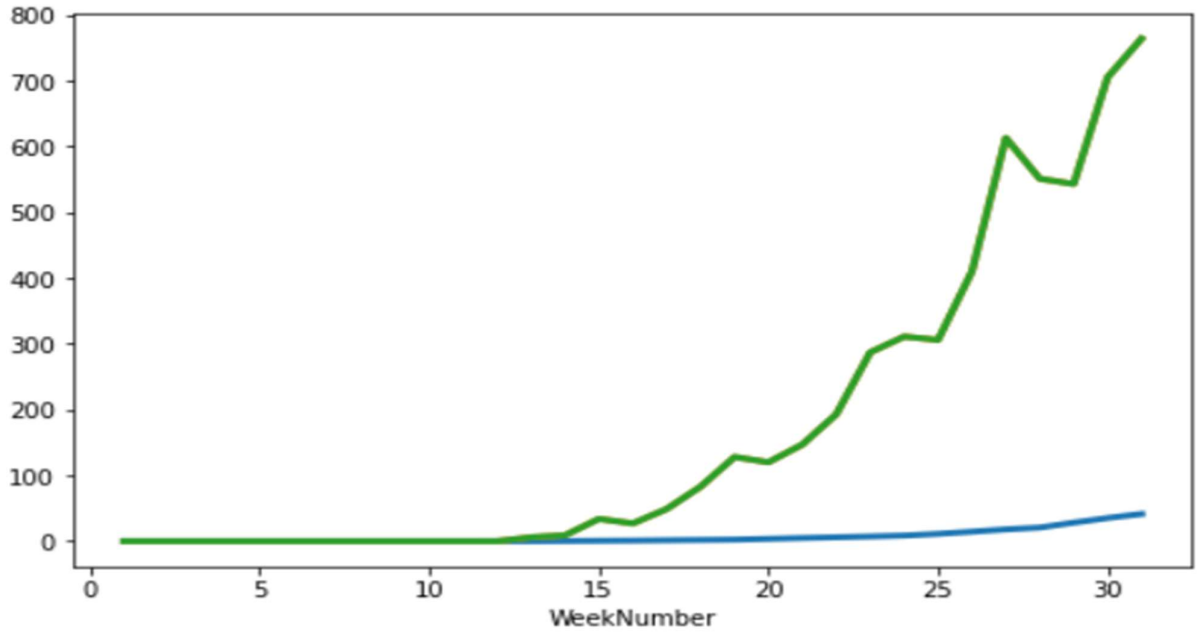
```
Average increase in number of new_cases_per_million cases everyday: 28.0
Average increase in number of new_deaths cases everyday: 58.0
Average increase in number of total_cases_per_million cases everyday: 2805.0
```




```
Text(0.5, 1.0, 'Top 15 countries as per number of death cases')
```



```
Text(0.5, 0, 'WeekNumber')
```



:

	Dates	LR	SVR
0	2020-08-02	460686	908711
1	2020-08-03	463367	929691
2	2020-08-04	466049	951064
3	2020-08-05	468730	972834
4	2020-08-06	471411	995008

	Dates	LR	SVR	Holts Linear Model Prediction
0	2020-08-02	460686	908711	568600
1	2020-08-03	463367	929691	573295
2	2020-08-04	466049	951064	577991
3	2020-08-05	468730	972834	582687
4	2020-08-06	471411	995008	587382

SYSTEM TRAILS

After completion of App, we will try the application on different system with different configurations.

This step is performed to check the compatibility of our app.

8. CONCLUSION:

We outline the system design and estimating techniques for COVID-19-DATA-ANALYSIS-AND-VISUALIZATION in this project, which employs open surveys to track the spread of the COVID-19 pandemic. Our graphical estimates, which offer an estimated of the cumulative and active number of cases in various geographic locations, need a significant amount of data from active participants yet offer meaningful depictions of the pandemic's progression in various places.

The most important challenge and limitation of Corona Surveys is the volume of survey responses is Corona Surveys' most significant obstacle and restriction. In this regard, it's crucial to share our graphical estimates to retain and engage users as much as possible. To facilitate the spread of our graphical visualisations and hence boost user recruitment, we intend to incorporate a projection of the number of cases and fatalities based on recent data for various geographic locations in the future.

Our heuristic outlier identification techniques are also susceptible to future advancements that could make them more resistant to malevolent reactions. As Corona Surveys develops, elements like the number of responses each day may be added to help identify specific malicious attacks that open internet surveys may be vulnerable to.

Although more comparisons in various countries are needed, our initial analysis, which compared the findings of Corona Surveys with a serology investigation in Spain, produced outstanding results and supported open surveys and indirect reporting as possible sources of information to follow pandemics. Considering the "wisdom of the crowd" phenomenon, it would be interesting to discuss the minimum number of responses needed to generate estimates that are reasonably accurate. As the number of responses rises, individual errors in over- or underestimation are balanced out, and our outlier detection methods function better. The minimum number of responses will inevitably depend on factors like population dispersal and cultural influences on behaviour, however our initial estimation is that by indirectly giving data

for a portion of the population equivalent to that of a massive serology study, we can already provide valuable estimates.

In conclusion, massive serology testing is ultimately the standard to accurately estimate the prevalence of COVID-19 in a region. However, this has its limitations, since it requires time until deployment, involves massive resources, and is unfeasible in some scenarios and countries. As an example, in the current outbreak in India as of April 2021, the level of underreporting is likely to be very which matches what is observe in the outputs. In these scenarios, we believe indirect reporting can provide a viable alternative to obtain early approximations of prevalence. Although Corona Surveys is a work in progress and much fine tuning is still required, we believe it provides a proof of concept of indirect reporting, as well as early results on its feasibility.