Covid_19_India data Visualization with live API

Date: 14-MAY-2020

BY: Ronak Sharma & Esha Patel

Abstract

Due to outburst of Corona-Virus Globally, Coronavirus possess a distinctive morphology, the name being derived from the outer fringe, or “corona” of embedded envelope protein, Coronavirus attracted little interest beyond causing mild upper respiratory tract infections. This project aspires to visualize the LIVE DATA(API) of Covid19 India and in near future you can also see the LIVE comparison of India with other Countries and World Wide scenario and This following project also tends to finds active cases and deaths in a particular State’s of India. This project has many future plans, currently project is in just its initial stage.

Data Visualizing & Analyzing Coronavirus: Getting the Dataset

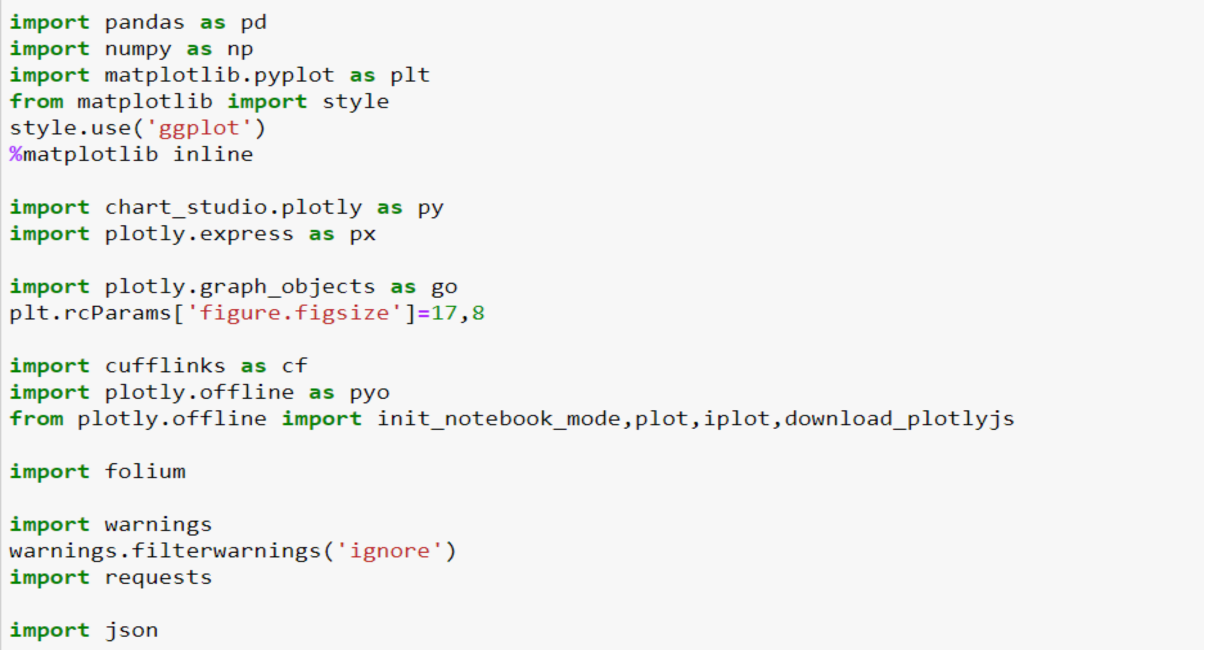
The dataset is been abstracted from JSON LIVE API of Covid19India and this json file is been converted and then furtherly used for the project. You may observe the dataset details following:

1. Dataset of India:
   a. Link of dataset (INDIA): ‘https://api.covid19india.org/data.json’
   b. Column Names: Active, confirmed, deaths, deltaconfirmed, deltadeaths, deltarecovered, lastupdatedtime, recovered, state, statecode

2. Dataset of World:
   a. Link of dataset (WORLD): ‘https://api.covid19api.com/summary’
   b. Column Names: Country, CountryCode, Slug, NewConfirmed, TotalConfirmed, NewDeaths, TotalDeaths, NewRecovered, TotalRecovered, Date

Importing Libraries and Importing Datasets

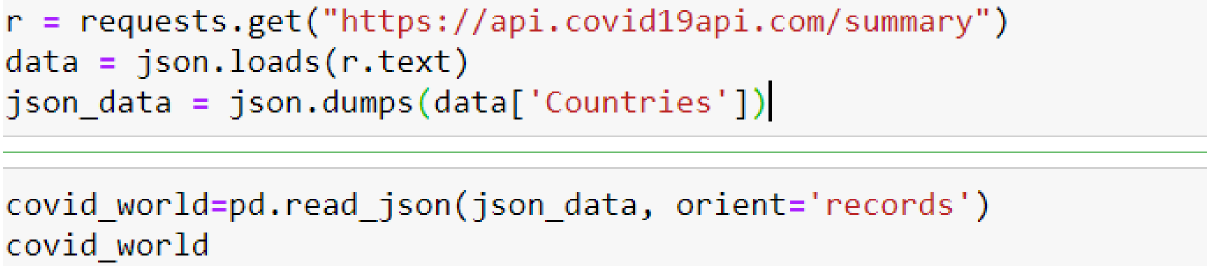
a. Importing libraries for further use in our project for our convenience:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
%matplotlib inline

import chart_studio.plotly as py
import plotly.express as px

import plotly.graph_objects as go
plt.rcParams['figure.figsize']=17,8

import cufflinks as cf
import plotly.offline as pyo
from plotly.offline import init_notebook_mode,plot,iplot,download_plotlyjs

import folium

import warnings
warnings.filterwarnings('ignore')
import requests

import json
```

Fig 1. Importing libraries

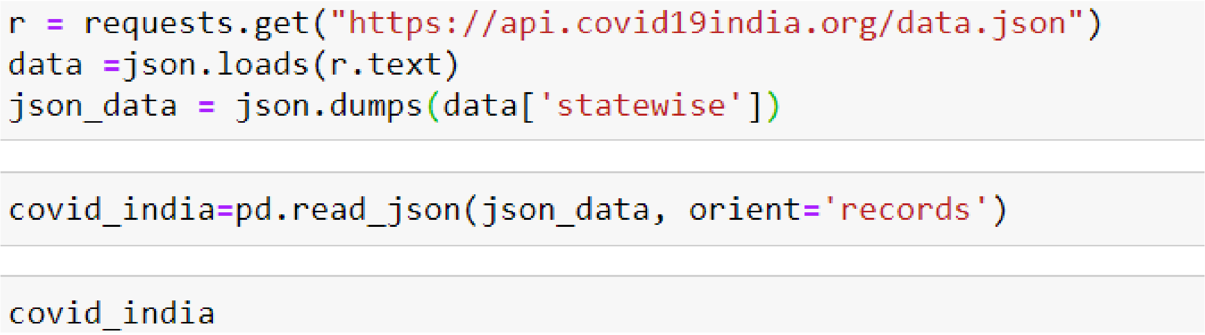
b. Requesting API and converting into readable format which is into a table and store that table into a variable:



```
r = requests.get("https://api.covid19api.com/summary")
data = json.loads(r.text)
json_data = json.dumps(data['Countries'])
```

```
covid_world=pd.read_json(json_data, orient='records')
covid_world
```

fig 2. Requesting API of Covid World



```
r = requests.get("https://api.covid19india.org/data.json")
data =json.loads(r.text)
json_data = json.dumps(data['statewise'])
```

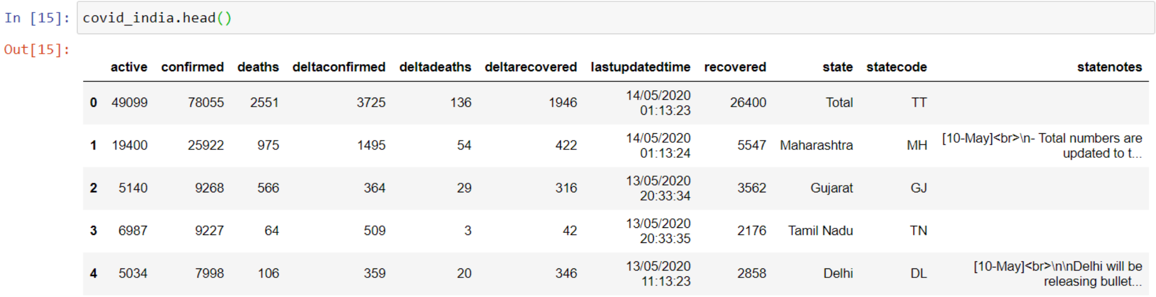
```
covid_india=pd.read_json(json_data, orient='records')
```

```
covid_india
```

fig 3. Requesting API of Covid India

Visualizing Dataset and Understanding the Dataset

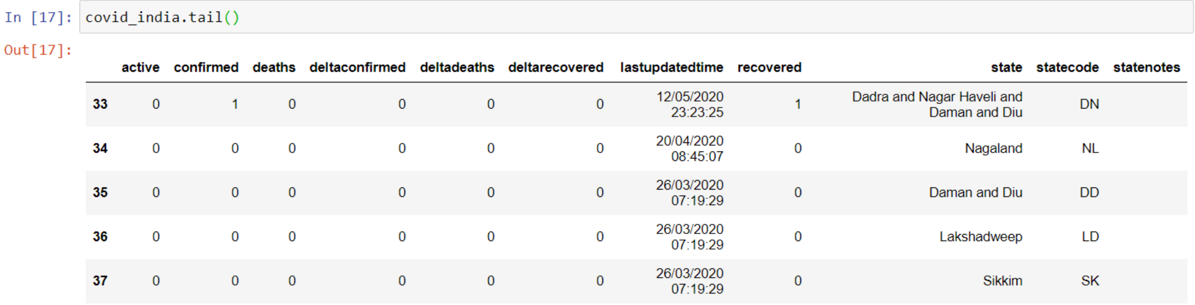
a. Visualizing head and tail of the data frame



```
In [15]: covid_india.head()
Out[15]:
```

	active	confirmed	deaths	deltaconfirmed	deltadeaths	deltarecovered	lastupdatedtime	recovered	state	statecode	statenotes
0	49099	78055	2551	3725	136	1946	14/05/2020 01:13:23	26400	Total	TT	
1	19400	25922	975	1495	54	422	14/05/2020 01:13:24	5547	Maharashtra	MH	[10-May] \n- Total numbers are updated to t...
2	5140	9268	566	364	29	316	13/05/2020 20:33:34	3562	Gujarat	GJ	
3	6987	9227	64	509	3	42	13/05/2020 20:33:35	2176	Tamil Nadu	TN	
4	5034	7998	106	359	20	346	13/05/2020 11:13:23	2858	Delhi	DL	[10-May] \n\nDelhi will be releasing bullet...

Fig 4. Head of the data

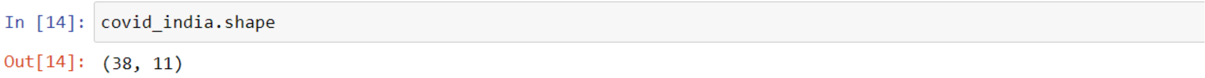


```
In [17]: covid_india.tail()
Out[17]:
```

	active	confirmed	deaths	deltaconfirmed	deltadeaths	deltarecovered	lastupdatedtime	recovered	state	statecode	statenotes
33	0	1	0	0	0	0	12/05/2020 23:23:25	1	Dadra and Nagar Haveli and Daman and Diu	DN	
34	0	0	0	0	0	0	20/04/2020 08:45:07	0	Nagaland	NL	
35	0	0	0	0	0	0	26/03/2020 07:19:29	0	Daman and Diu	DD	
36	0	0	0	0	0	0	26/03/2020 07:19:29	0	Lakshadweep	LD	
37	0	0	0	0	0	0	26/03/2020 07:19:29	0	Sikkim	SK	

Fig 5. Tail of the data

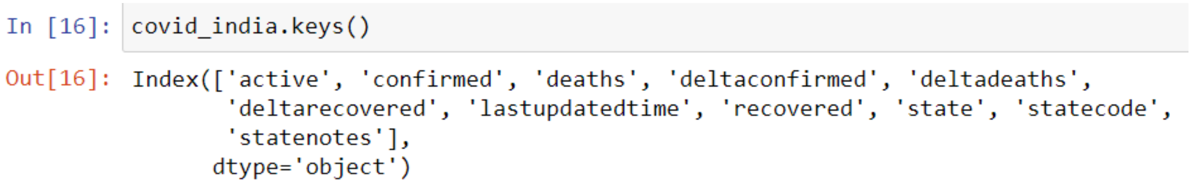
b. Checking Number of Rows and Columns in the data frame



```
In [14]: covid_india.shape
Out[14]: (38, 11)
```

Fig 6. Checking shape

c. Fetching all the Attributes of the Column

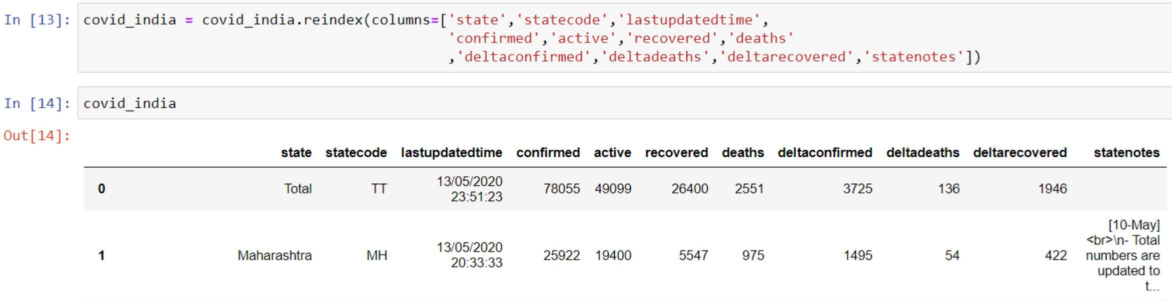


```
In [16]: covid_india.keys()

Out[16]: Index(['active', 'confirmed', 'deaths', 'deltaconfirmed', 'deltadeaths',
                'deltarecovered', 'lastupdatedtime', 'recovered', 'state', 'statecode',
                'statenotes'],
               dtype='object')
```

Fig 6. Keys of the data frame

Wrangling the Data

c. Rearranging the Columns for clear understanding



```
In [13]: covid_india = covid_india.reindex(columns=['state','statecode','lastupdatedtime',
                                            'confirmed','active','recovered','deaths'
                                            ,'deltaconfirmed','deltadeaths','deltarecovered','statenotes'])
```

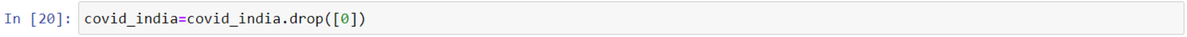
```
In [14]: covid_india
```

Out[14]:

	state	statecode	lastupdatedtime	confirmed	active	recovered	deaths	deltaconfirmed	deltadeaths	deltarecovered	statenotes
0	Total	TT	13/05/2020 23:51:23	78055	49099	26400	2551	3725	136	1946	
1	Maharashtra	MH	13/05/2020 20:33:33	25922	19400	5547	975	1495	54	422	[10-May] \n- Total numbers are updated to t...

Fig 7. Rearranged columns

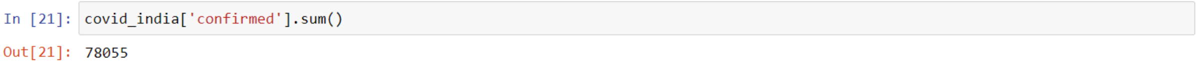
d. Dropping the 0 index because the column doesn't have any use in our data frame



```
In [20]: covid_india=covid_india.drop([0])
```

Fig 8. Dropping the column

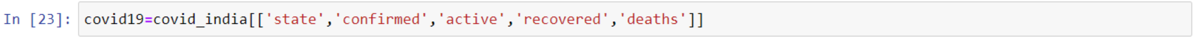
e. Sum of all the confirm cases in India



```
In [21]: covid_india['confirmed'].sum()

Out[21]: 78055
```

Fig 8.Adding all tha values

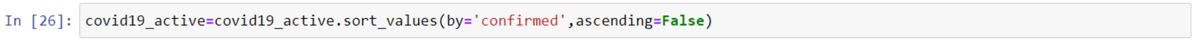
f. Making a sperate data frame for STATE, CONFIRM, ACTIVE & DEATHS cases in India



```
In [23]: covid19=covid_india[['state','confirmed','active','recovered','deaths']]
```

Fig 9.Making new data frame

g. Sorting the values in descending order for future data visualization

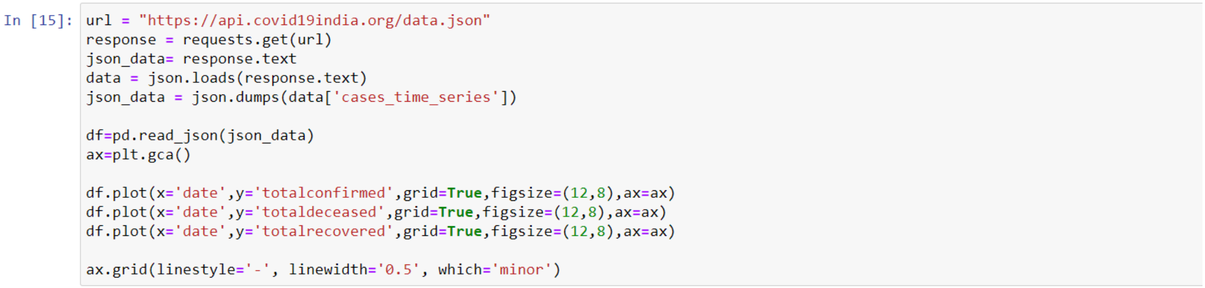


```
In [26]: covid19_active=covid19_active.sort_values(by='confirmed',ascending=False)
```

Fig 10.Sorting

Data Visualization

a. This is a Line chart here we are comparing Total confirmed , Total Deceased & Total Recovered with each other on a time stamp here x axis indicates date & y axis indicates the count of the Line chart.



```
In [15]: url = "https://api.covid19india.org/data.json"
         response = requests.get(url)
         json_data= response.text
         data = json.loads(response.text)
         json_data = json.dumps(data['cases_time_series'])

         df=pd.read_json(json_data)
         ax=plt.gca()

         df.plot(x='date',y='totalconfirmed',grid=True,figsize=(12,8),ax=ax)
         df.plot(x='date',y='totaldeceased',grid=True,figsize=(12,8),ax=ax)
         df.plot(x='date',y='totalrecovered',grid=True,figsize=(12,8),ax=ax)

         ax.grid(linestyle='-', linewidth='0.5', which='minor')
```

Fig 11.a. Input for Line chart

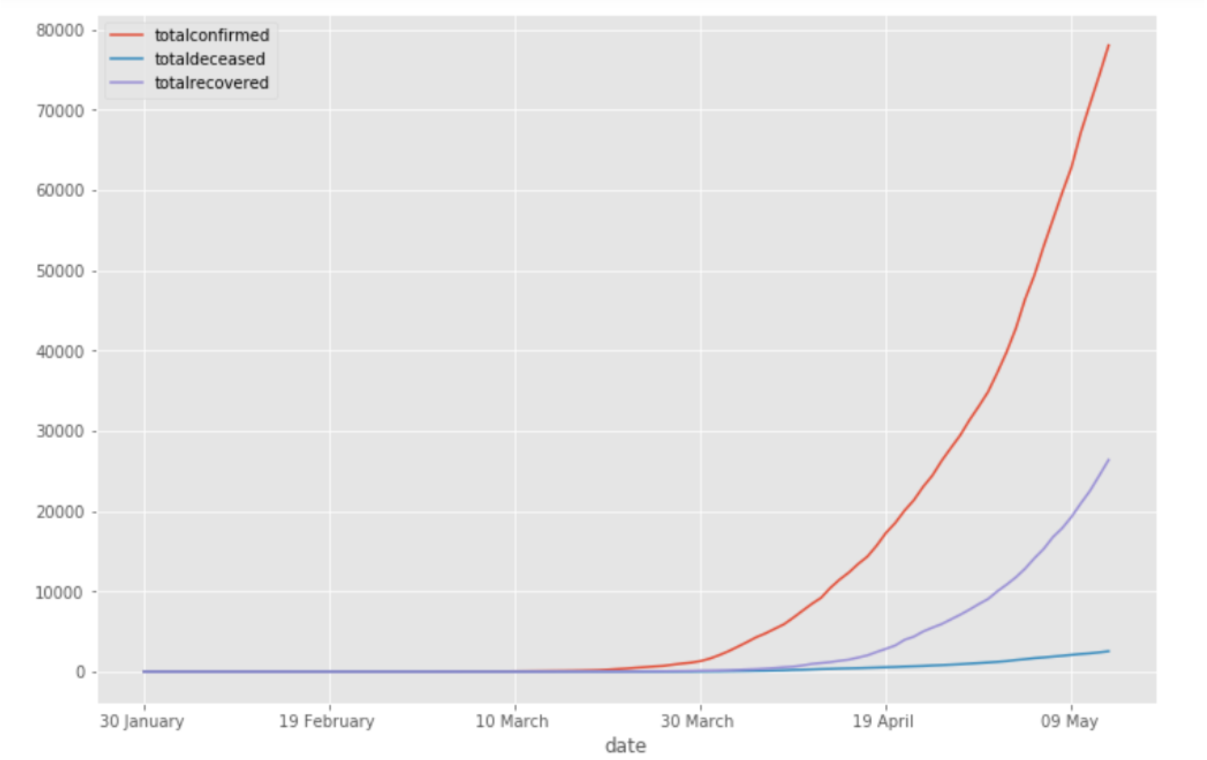
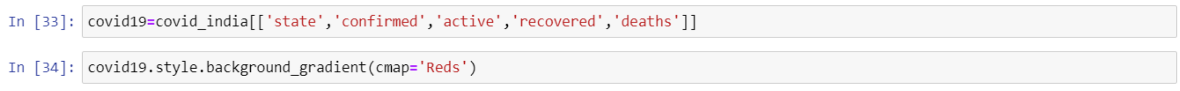


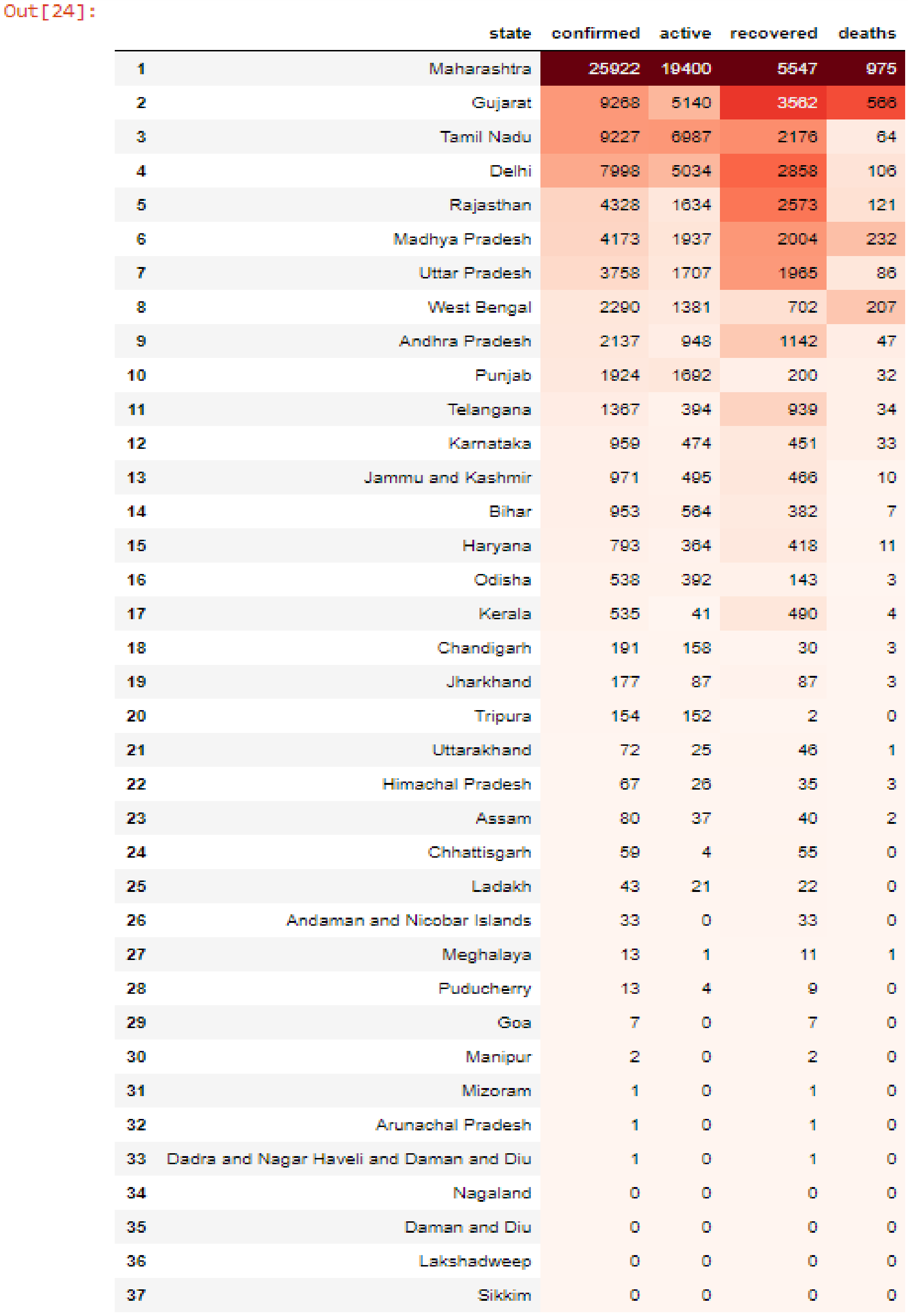
Fig 11.b. Output

b. Here we have created a new dataframe from our old Dataset which was Covid_India. In this data set we are combining only selected and later on we are making a heatmap out of this data set. Heatmap shows the color intensity towards the higher values as compared to low affected so we could conclude that the more red dense area is highly affected.



```
In [33]: covid19=covid_india[['state','confirmed','active','recovered','deaths']]

In [34]: covid19.style.background_gradient(cmap='Reds')
```

Fig 12.a. Creating new Dataset and making Heatmap out of it

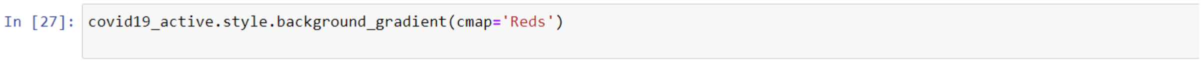


Out[24]:

	state	confirmed	active	recovered	deaths
1	Maharashtra	25922	19400	5547	975
2	Gujarat	9268	5140	3562	566
3	Tamil Nadu	9227	6987	2176	64
4	Delhi	7998	5034	2858	106
5	Rajasthan	4328	1634	2573	121
6	Madhya Pradesh	4173	1937	2004	232
7	Uttar Pradesh	3758	1707	1965	86
8	West Bengal	2290	1381	702	207
9	Andhra Pradesh	2137	948	1142	47
10	Punjab	1924	1692	200	32
11	Telangana	1367	394	939	34
12	Karnataka	959	474	451	33
13	Jammu and Kashmir	971	495	466	10
14	Bihar	953	564	382	7
15	Haryana	793	364	418	11
16	Odisha	538	392	143	3
17	Kerala	535	41	490	4
18	Chandigarh	191	158	30	3
19	Jharkhand	177	87	87	3
20	Tripura	154	152	2	0
21	Uttarakhand	72	25	46	1
22	Himachal Pradesh	67	26	35	3
23	Assam	80	37	40	2
24	Chhattisgarh	59	4	55	0
25	Ladakh	43	21	22	0
26	Andaman and Nicobar Islands	33	0	33	0
27	Meghalaya	13	1	11	1
28	Puducherry	13	4	9	0
29	Goa	7	0	7	0
30	Manipur	2	0	2	0
31	Mizoram	1	0	1	0
32	Arunachal Pradesh	1	0	1	0
33	Dadra and Nagar Haveli and Daman and Diu	1	0	1	0
34	Nagaland	0	0	0	0
35	Daman and Diu	0	0	0	0
36	Lakshadweep	0	0	0	0
37	Sikkim	0	0	0	0

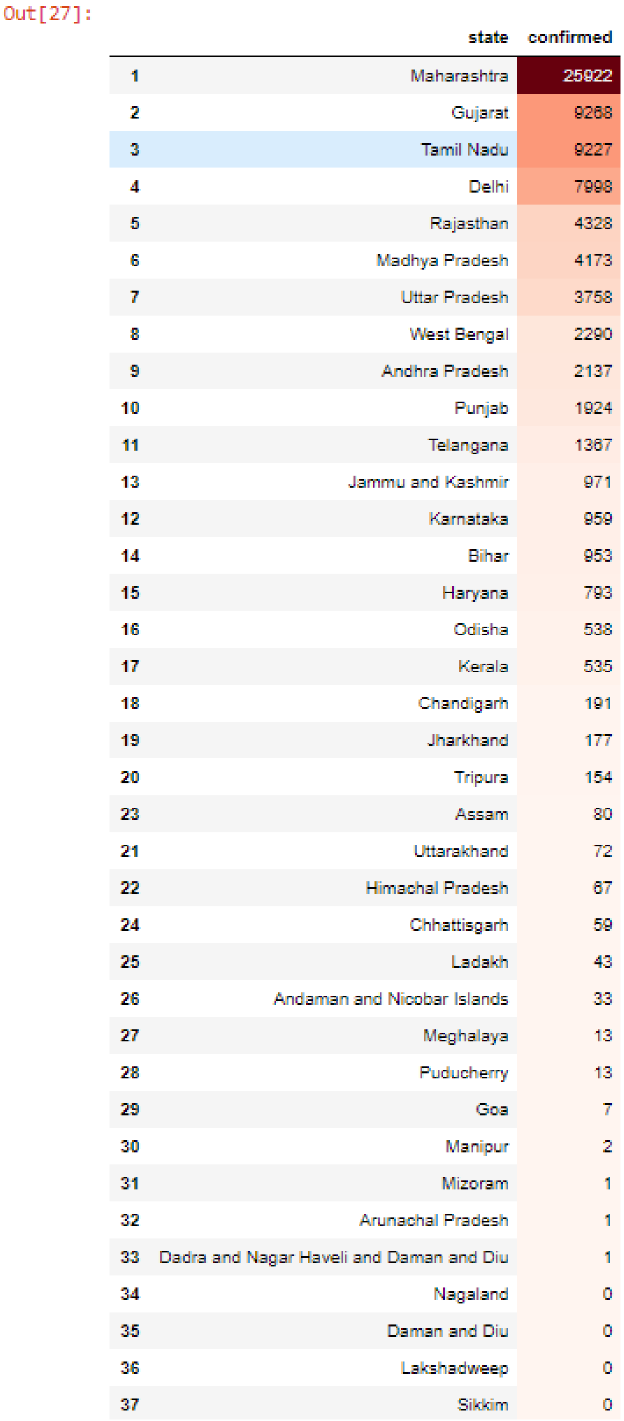
Fig 12.b. Output for gradient

c. Applying Gradient for a single column



```
In [27]: covid19_active.style.background_gradient(cmap='Reds')
```

Fig13. a. Getting Input for the column gradient



Out[27]:

	state	confirmed
1	Maharashtra	25922
2	Gujarat	9268
3	Tamil Nadu	9227
4	Delhi	7998
5	Rajasthan	4328
6	Madhya Pradesh	4173
7	Uttar Pradesh	3758
8	West Bengal	2290
9	Andhra Pradesh	2137
10	Punjab	1924
11	Telangana	1367
13	Jammu and Kashmir	971
12	Karnataka	959
14	Bihar	953
15	Haryana	793
16	Odisha	538
17	Kerala	535
18	Chandigarh	191
19	Jharkhand	177
20	Tripura	154
23	Assam	80
21	Uttarakhand	72
22	Himachal Pradesh	67
24	Chhattisgarh	59
25	Ladakh	43
26	Andaman and Nicobar Islands	33
27	Meghalaya	13
28	Puducherry	13
29	Goa	7
30	Manipur	2
31	Mizoram	1
32	Arunachal Pradesh	1
33	Dadra and Nagar Haveli and Daman and Diu	1
34	Nagaland	0
35	Daman and Diu	0
36	Lakshadweep	0
37	Sikkim	0

Fig 13.b. Output for Confirmed Cases

d. Bar graph and scatted plot for the Confirm cases on y axis and states on x axis which help us to explain the Confirm cases situation in more appropriate manner.

```
In [35]: covid19.iplot(kind='bar',x='state',y='confirmed')
```

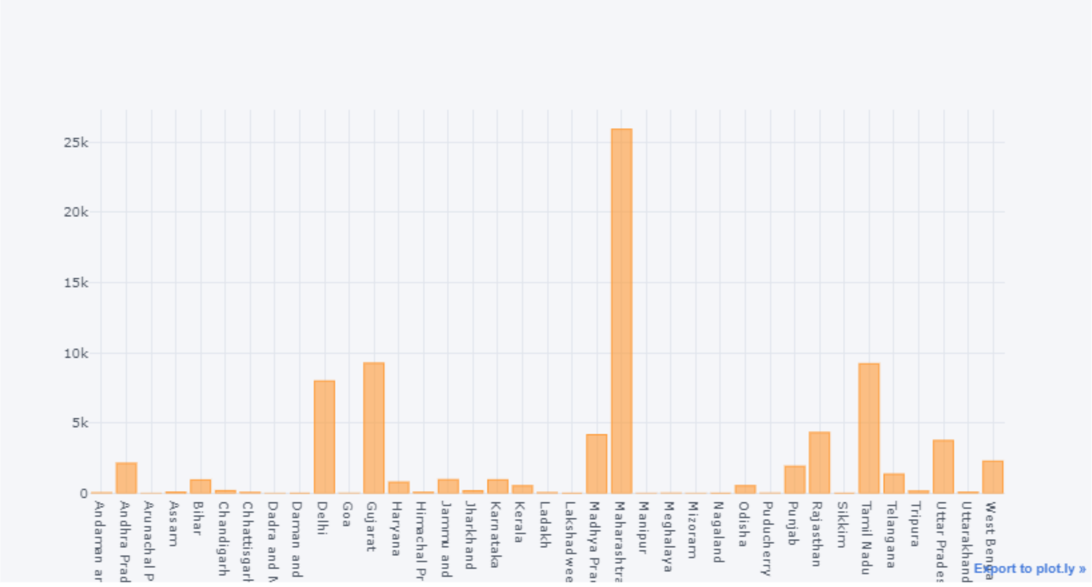


Fig 14.a. Bar Graph

```
In [29]: covid19.iplot(kind='scatter',x='state',y='confirmed',mode='markers+lines',title="Confirmed Cases in INDIA"
                       ,xTitle='Name of the states',yTitle='Total cases',colors='red',size=15)
```

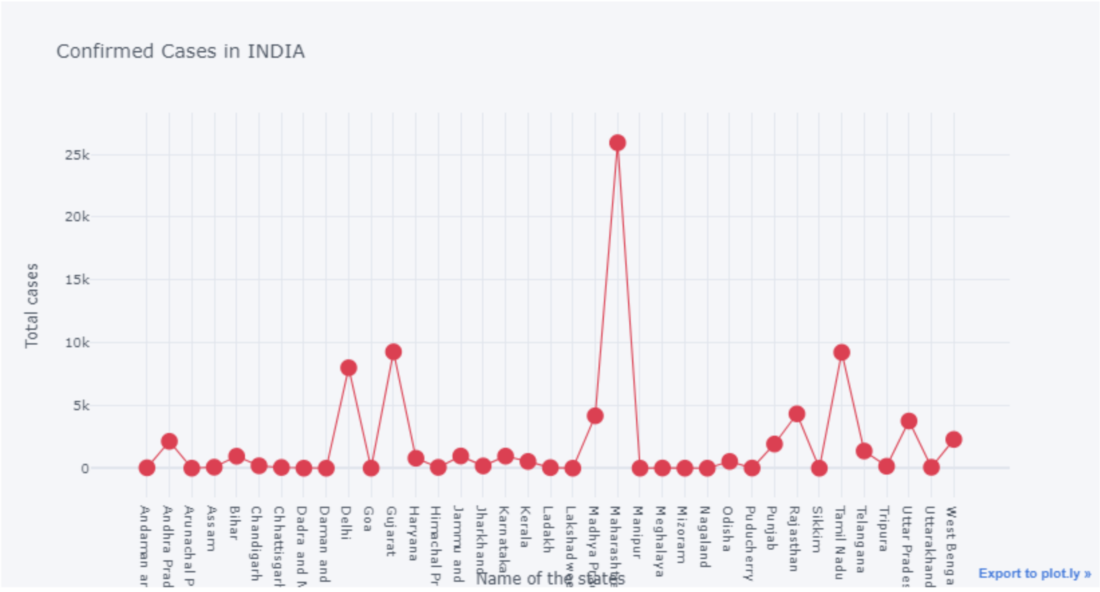
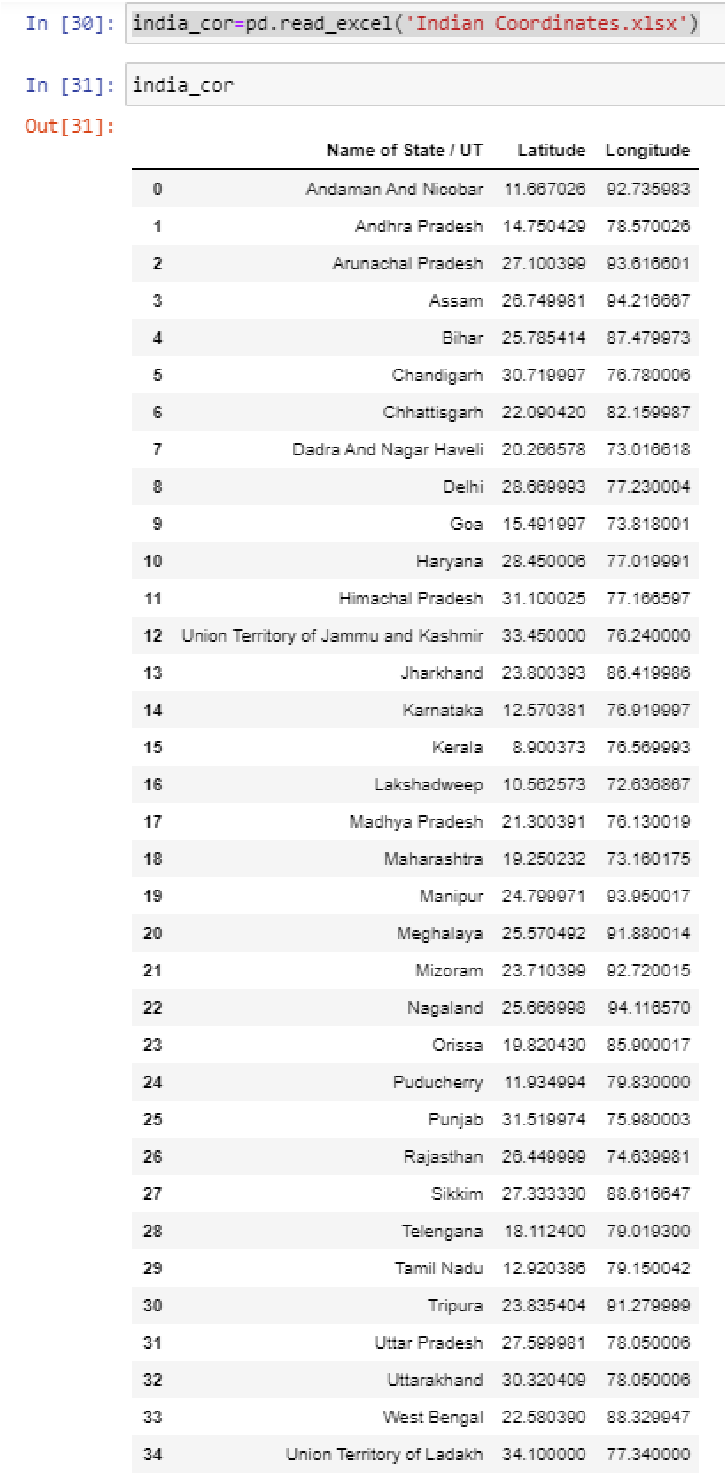


Fig 14.b. Scatter Plot

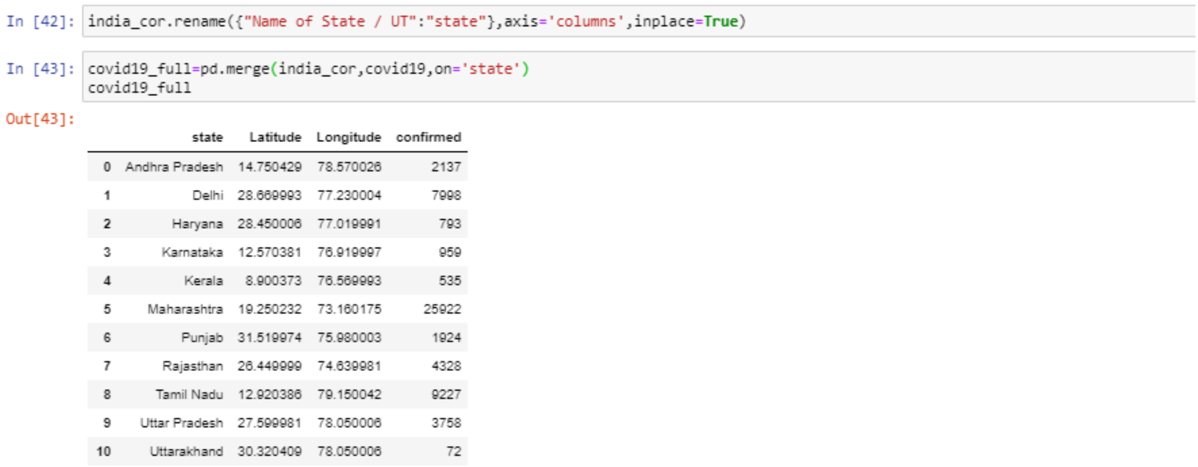
e. Calling India Map coordinates CSV file to combine our previous dataset with India coordinated and make a new dataset for new visualization technique.



```
In [30]: india_cor=pd.read_excel('Indian Coordinates.xlsx')

In [31]: india_cor
Out[31]:
```

	Name of State / UT	Latitude	Longitude
0	Andaman And Nicobar	11.667026	92.735983
1	Andhra Pradesh	14.750429	78.570026
2	Arunachal Pradesh	27.100399	93.616601
3	Assam	26.749981	94.216667
4	Bihar	25.785414	87.479973
5	Chandigarh	30.719997	76.780006
6	Chhattisgarh	22.090420	82.159987
7	Dadra And Nagar Haveli	20.266578	73.016618
8	Delhi	28.669993	77.230004
9	Goa	15.491997	73.818001
10	Haryana	28.450006	77.019991
11	Himachal Pradesh	31.100025	77.166597
12	Union Territory of Jammu and Kashmir	33.450000	76.240000
13	Jharkhand	23.800393	86.419986
14	Karnataka	12.570381	76.919997
15	Kerala	8.900373	76.569993
16	Lakshadweep	10.562573	72.636867
17	Madhya Pradesh	21.300391	76.130019
18	Maharashtra	19.250232	73.160175
19	Manipur	24.799971	93.950017
20	Meghalaya	25.570492	91.880014
21	Mizoram	23.710399	92.720015
22	Nagaland	25.666998	94.116570
23	Orissa	19.820430	85.900017
24	Puducherry	11.934994	79.830000
25	Punjab	31.519974	75.980003
26	Rajasthan	26.449999	74.639981
27	Sikkim	27.333330	88.616647
28	Telengana	18.112400	79.019300
29	Tamil Nadu	12.920386	79.150042
30	Tripura	23.835404	91.279999
31	Uttar Pradesh	27.599981	78.050006
32	Uttarakhand	30.320409	78.050006
33	West Bengal	22.580390	88.329947
34	Union Territory of Ladakh	34.100000	77.340000

Fig 15.a. Getting a new Dataset

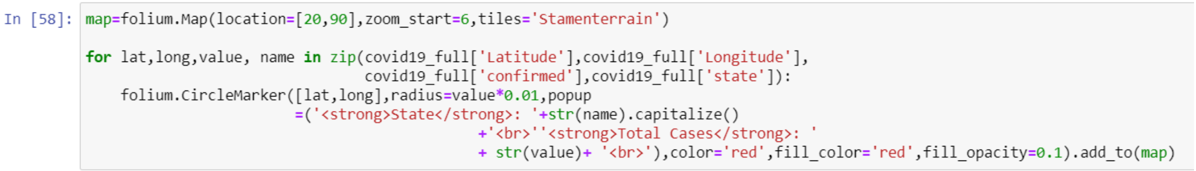


```
In [42]: india_cor.rename({"Name of State / UT":"state"},axis='columns',inplace=True)

In [43]: covid19_full=pd.merge(india_cor,covid19,on='state')
         covid19_full

Out[43]:
```

	state	Latitude	Longitude	confirmed
0	Andhra Pradesh	14.750429	78.570026	2137
1	Delhi	28.669993	77.230004	7998
2	Haryana	28.450006	77.019991	793
3	Karnataka	12.570381	76.919997	959
4	Kerala	8.900373	76.569993	535
5	Maharashtra	19.250232	73.160175	25922
6	Punjab	31.519974	75.980003	1924
7	Rajasthan	26.449999	74.639981	4328
8	Tamil Nadu	12.920386	79.150042	9227
9	Uttar Pradesh	27.599981	78.050006	3758
10	Uttarakhand	30.320409	78.050006	72

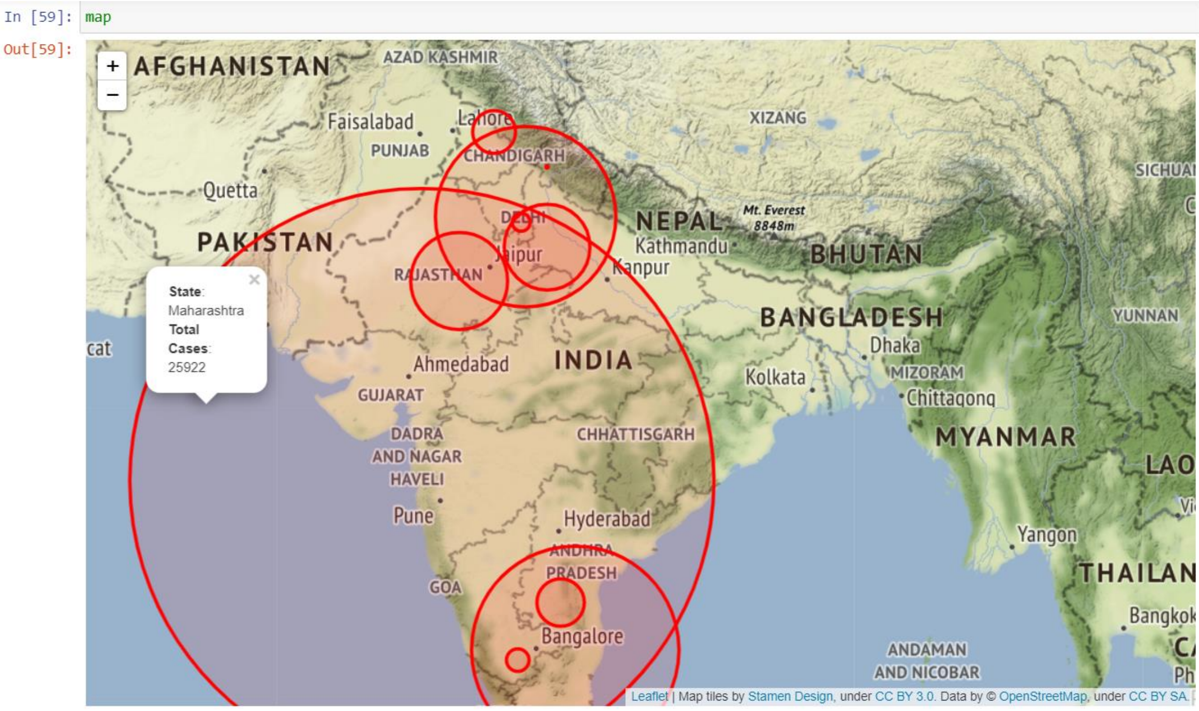
Fig 15.b. merging Two Columns

f. Making an Indian map to spot the Covid affected areas on the 3D map



```
In [58]: map=folium.Map(location=[20,90],zoom_start=6,tiles='Stamenterrain')

         for lat,long,value, name in zip(covid19_full['Latitude'],covid19_full['Longitude'],
                                         covid19_full['confirmed'],covid19_full['state']):
             folium.CircleMarker([lat,long],radius=value*0.01,popup
                                 =('<strong>State</strong>: '+str(name).capitalize()
                                                     +'<br>''<strong>Total Cases</strong>: '
                                                     + str(value)+ '<br>'),color='red',fill_color='red',fill_opacity=0.1).add_to(map)
```

Fig 16.a generating input



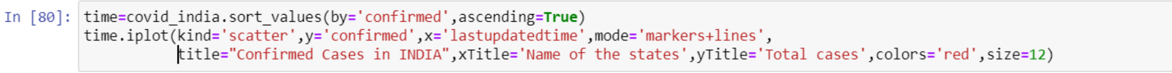
```
In [59]: map
Out[59]:
```

State: Maharashtra Total Cases: 25922

Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under CC BY SA.

Fig 16.b Output On MAP

g. Generating a time series through scatter plot to show the time stamp taking x axis as dates of the cases are confirmed and y axis as Confirmed cases in India



```
In [80]: time=covid_india.sort_values(by='confirmed',ascending=True)
         time.iplot(kind='scatter',y='confirmed',x='lastupdatedtime',mode='markers+lines',
                    title="Confirmed Cases in INDIA",xTitle='Name of the states',yTitle='Total cases',colors='red',size=12)
```

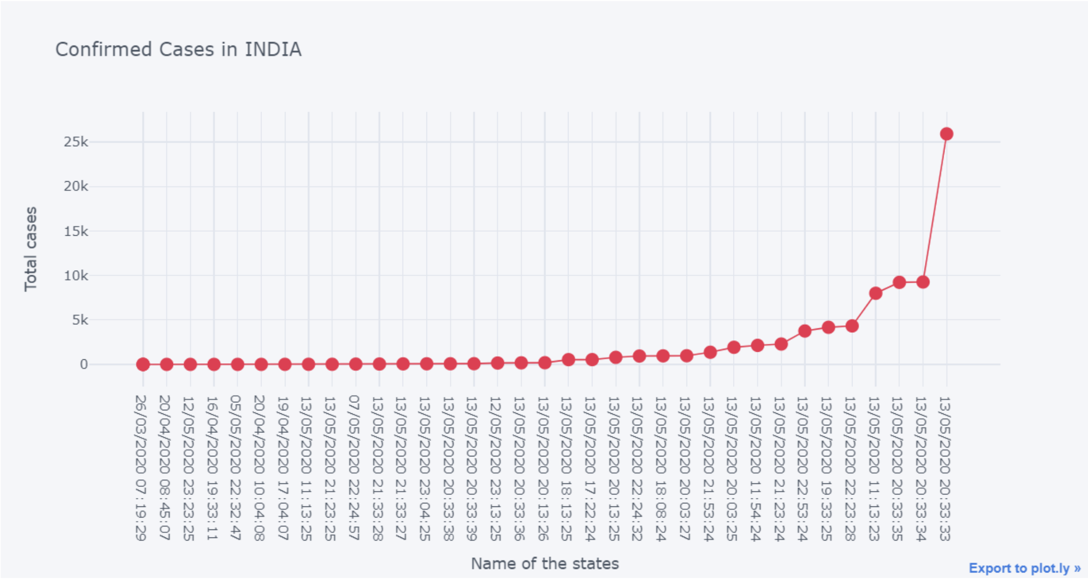
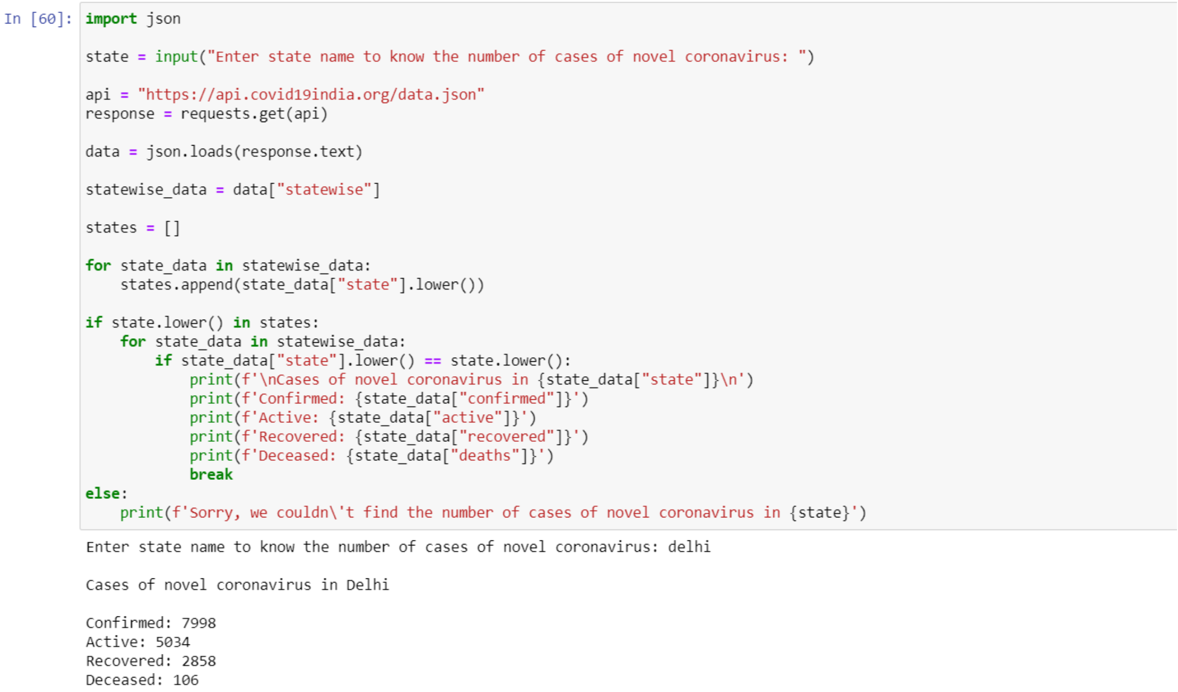


Fig 17 Scatter plot

Function

- This function help us to find the details of state which is been entered by the user as a input.



```
In [60]: import json

         state = input("Enter state name to know the number of cases of novel coronavirus: ")

         api = "https://api.covid19india.org/data.json"
         response = requests.get(api)

         data = json.loads(response.text)

         statewise_data = data["statewise"]

         states = []

         for state_data in statewise_data:
             states.append(state_data["state"].lower())

         if state.lower() in states:
             for state_data in statewise_data:
                 if state_data["state"].lower() == state.lower():
                     print(f'\nCases of novel coronavirus in {state_data["state"]}\n')
                     print(f'Confirmed: {state_data["confirmed"]}')
                     print(f'Active: {state_data["active"]}')
                     print(f'Recovered: {state_data["recovered"]}')
                     print(f'Deceased: {state_data["deaths"]}')
                     break
         else:
             print(f'Sorry, we couldn\'t find the number of cases of novel coronavirus in {state}')
```

```
Enter state name to know the number of cases of novel coronavirus: delhi

Cases of novel coronavirus in Delhi

Confirmed: 7998
Active: 5034
Recovered: 2858
Deceased: 106
```

Fig 18 detail of the state function

Future goals

- This project will be on air as a live merging all the values into the database and make it user responsive.
- Further comparison between countries will be done for more in depth understanding
- No dedicated module will be formed because this module done have a higher accuracy as such it is a live data so predication may change as per the fluctuation in data can be observed
- It will be an open project for other developers and an API key will be generated for the above project
- More method for data visualization will be observed in near future

Conclusion

- This Live visualization techniques have more functionality and usage as it appears in above snippets and this conclude with more comprehensible Intelligible is done and which may carry forward in near future.