COVID-19 - Pandemic in India!

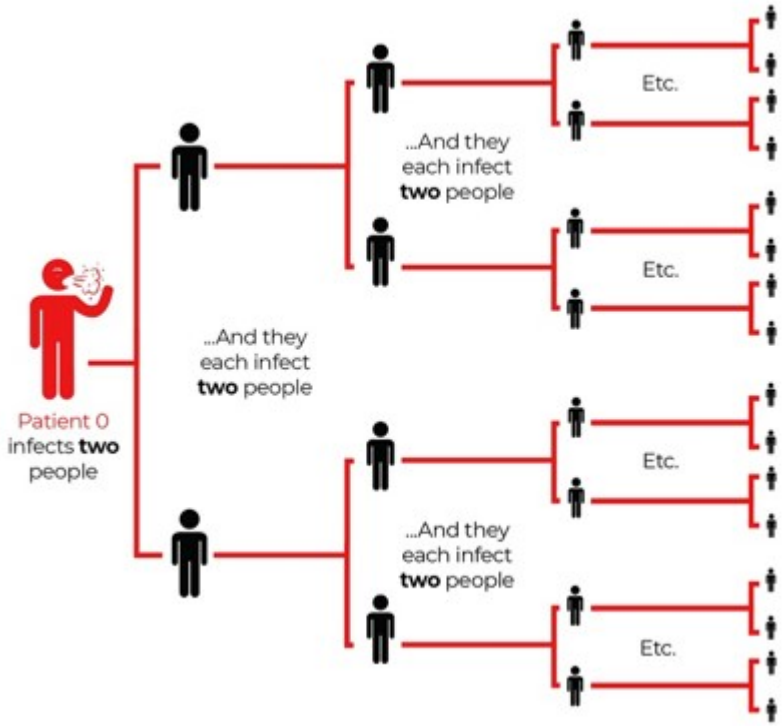
About COVID-19

The **coronavirus (COVID-19)** pandemic is the greatest global humanitarian challenge the world has faced since World War II. The pandemic virus has spread widely, and the number of cases is rising daily. The government is working to slow down its spread.

Till date it has spread across 215 countries infecting 5,491,194 people and killing 346,331 so far. In India, as many as 138,536 COVID-19 cases have been reported so far. Of these, 57,692 have recovered and 4,024 have died. COVID19

Corona Virus Explained in Simple Terms:

- Let's say Raghav got infected yesterday, but he won't know it untill next 14 days
- Raghav thinks he is healthy but he is infecting 10 persons per day
- Now these 10 persons think they are completely healthy, they travel, go out and infect 100 others
- These 100 persons think they are healthy but they have already infected 1000 persons
- No one knows who is healthy or who can infect you
- All you can do is be responsible, stay in quarentine



Problem Statement:

India has responded quickly, implementing a proactive, nationwide, lockdown, to flatten the curve and use the time to plan and resource responses adequately. As of 23rd May 2020, India has witnessed 3720 deaths from 32 States and Union Territories, with a total of 123202 confirmed cases due to COVID-19. Globally the Data Scientists are using AI and machine learning to analyze, predict, and take safety measures against COVID-19 in India.

Goal:

We need a explore the COVID situation in India and the world, and strong model that predicts how the virus could spread across India in the next 15 days.

Tasks to be performed:

- Analyze the present condition in India
- Scrape out the COVID-19 from websites
- Figure out the death rate and cure rate per 100 across the affected states
- Create different charts to visualize the following:
    - Age group distribution of affected patients
    - Total sample test done till date
    - Growth rate of COVID in top 15 states
    - Top 10 States in each health facility
    - State wise testing insight
- ICMR testing centres in each state
- Use Prophet to predict the confirmed cases in India
- Use ARIMA to predict the confirmed cases in India
- Compare the Indian COVID cases globally

Importing the required libraries

```
# importing the required libraries
import pandas as pd

# Visualisation libraries
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

```
import folium
from folium import plugins

# Manipulating the default plot size
plt.rcParams['figure.figsize'] = 10, 12

# Disable warnings
import warnings
warnings.filterwarnings('ignore')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

Part 1: Analysing the present condition in India

How it started in India?:

The first **COVID-19** case was reported on 30th January 2020 when a student arrived **Kerala** from Wuhan. Just in next 2 days, Kerela reported 2 more cases. For almost a month, no new cases were reported in India, however, on 2nd March 2020, five new cases of corona virus were reported in Kerala again and since then the cases have been rising affecting **25** states, till now *(Bihar and Manipur being the most recent)*. Here is a brief timeline of the cases in India.

COVID-19 in India - Timeline

Recent COVID-19 updates in India

- Sikkim on Saturday reported its first +ve COVID-19 case
- With over 6,500 fresh cases, the Covid in India rose to 1,25,101 on Saturday morning, with 3,720 fatalities
- West Bengal asks Railways not to send migrant trains to State till May 26 in view of Cyclone Amphan
- 196 new COVID 19 positive cases were reported in Karnataka on Saturday
- Complete lockdown in Bengaluru on Sunday.
    - Bruhat Bengaluru Mahanagara Palike (BBMP) Commissioner B.H. Anil Kumar said the conditions and restrictions on Sunday will be similar to that under coronavirus lockdown 1.0.

How is AI-ML useful in fighting the COVID-19 pandemic?

- Medical resource optimization
- Ensuring demand planning stability
- Contact tracing
- Situational awareness and critical response analysis

1.1 Scraping the datasets from the official Govt. website

```
# for date and time opeations
from datetime import datetime
# for file and folder operations
import os
# for regular expression opeations
import re
# for listing files in a folder
import glob
# for getting web contents
import requests
# for scraping web contents
from bs4 import BeautifulSoup
```

```
# get data

# link at which web data recides
link = 'https://www.mohfw.gov.in/'
# get web data
req = requests.get(link)
# parse web data
soup = BeautifulSoup(req.content, "html.parser")
```

```
# find the table
# ==============
# our target table is the last table in the page

# get the table head
# table head may contain the column names, titles, subtitles
thead = soup.find_all('thead')[-1]
# print(thead)

# get all the rows in table head
# it usually have only one row, which has the column names
head = thead.find_all('tr')
# print(head)
```

```
# get the table tbody
# it contains the contents
tbody = soup.find_all('tbody')[-1]
# print(tbody)

# get all the rows in table body
# each row is each state's entry
body = tbody.find_all('tr')
# print(body)
```

```
# get the table contents
# ======================

# container for header rows / column title
head_rows = []
# container for table body / contents
body_rows = []

# loop through the head and append each row to head
for tr in head:
    td = tr.find_all(['th', 'td'])
    row = [i.text for i in td]
    head_rows.append(row)
# print(head_rows)

# loop through the body and append each row to body
for tr in body:
    td = tr.find_all(['th', 'td'])
    row = [i.text for i in td]
    body_rows.append(row)
# print(head_rows)
```

```
# save contents in a dataframe
# ============================

# skip last 3 rows, it contains unwanted info
# head_rows contains column title
df_bs = pd.DataFrame(body_rows[:len(body_rows)-6],
                     columns=head_rows[0])

# Drop 'S. No.' column
df_bs.drop('S. No.', axis=1, inplace=True)

# there are 36 states+UT in India
df_bs.head(36)
```

	Name of State / UT	Total Confirmed cases*	Cured/Discharged/Migrated	Deaths**
0	Andaman and Nicobar Islands	33	33	0
1	Andhra Pradesh	2757	1809	56
2	Arunachal Pradesh	1	1	0
3	Assam	329	55	4
4	Bihar	2380	653	11
5	Chandigarh	225	179	3
6	Chhattisgarh	214	64	0
7	Dadar Nagar Haveli	2	0	0
8	Delhi	12910	6267	231
9	Goa	55	16	0
10	Gujarat	13664	6169	829
11	Haryana	1131	750	16
12	Himachal Pradesh	185	61	3
13	Jammu and Kashmir	1569	774	21
14	Jharkhand	350	141	4
15	Karnataka	1959	608	42
16	Kerala	795	515	4
17	Ladakh	49	43	0
18	Madhya Pradesh	6371	3267	281
19	Maharashtra	47190	13404	1577
20	Manipur	29	4	0
21	Meghalaya	14	12	1
22	Mizoram	1	1	0
23	Odisha	1269	497	7
24	Puducherry	26	10	0#
25	Punjab	2045	1870	39
26	Rajasthan	6742	3786	160
27	Sikkim	1	0	0
28	Tamil Nadu	15512	7491	103

29	Telengana	1813	1065	49
30	Tripura	189	153	0

Data Cleaning

```
# date-time information
# ====================
#saving a copy of the dataframe
df_India = df_bs.copy()
# today's date
now  = datetime.now()
# format date to month-day-year
df_India['Date'] = now.strftime("%m/%d/%Y")

# add 'Date' column to dataframe
df_India['Date'] = pd.to_datetime(df_India['Date'], format='%m/%d/%Y')

# df_India.head(36)
```

```
# remove extra characters from 'Name of State/UT' column
df_India['Name of State / UT'] = df_India['Name of State / UT'].str.replace('#', '')
df_India['Deaths**'] = df_India['Deaths**'].str.replace('#', '')
```

```
# latitude and longitude information
# ==================================

# latitude of the states
lat = {'Delhi':28.7041, 'Haryana':29.0588, 'Kerala':10.8505, 'Rajasthan':27.0238,
       'Telengana':18.1124, 'Uttar Pradesh':26.8467, 'Ladakh':34.2996, 'Tamil Nadu':11.1271,
       'Jammu and Kashmir':33.7782, 'Punjab':31.1471, 'Karnataka':15.3173, 'Maharashtra':19.7
       'Andhra Pradesh':15.9129, 'Odisha':20.9517, 'Uttarakhand':30.0668, 'West Bengal':22.98
       'Puducherry': 11.9416, 'Chandigarh': 30.7333, 'Chhattisgarh':21.2787, 'Gujarat': 22.25
       'Himachal Pradesh': 31.1048, 'Madhya Pradesh': 22.9734, 'Bihar': 25.0961, 'Manipur':24
       'Mizoram':23.1645, 'Goa': 15.2993, 'Andaman and Nicobar Islands': 11.7401, 'Assam' : 2
       'Jharkhand': 23.6102, 'Arunachal Pradesh': 28.2180, 'Tripura': 23.9408, 'Nagaland': 26
       'Meghalaya' : 25.4670, 'Dadar Nagar Haveli' : 20.1809, 'Sikkim': 27.5330}

# longitude of the states
long = {'Delhi':77.1025, 'Haryana':76.0856, 'Kerala':76.2711, 'Rajasthan':74.2179,
        'Telengana':79.0193, 'Uttar Pradesh':80.9462, 'Ladakh':78.2932, 'Tamil Nadu':78.6569,
        'Jammu and Kashmir':76.5762, 'Punjab':75.3412, 'Karnataka':75.7139, 'Maharashtra':75.
        'Andhra Pradesh':79.7400, 'Odisha':85.0985, 'Uttarakhand':79.0193, 'West Bengal':87.8
        'Puducherry': 79.8083, 'Chandigarh': 76.7794, 'Chhattisgarh':81.8661, 'Gujarat': 71.1
        'Himachal Pradesh': 77.1734, 'Madhya Pradesh': 78.6569, 'Bihar': 85.3131, 'Manipur':9
        'Mizoram':92.9376, 'Goa': 74.1240, 'Andaman and Nicobar Islands': 92.6586, 'Assam' :
        'Jharkhand': 85.2799, 'Arunachal Pradesh': 94.7278, 'Tripura': 91.9882, 'Nagaland': 9
        'Meghalaya' : 91.3662, 'Dadar Nagar Haveli' : 73.0169, 'Sikkim': 88.5122}
```

```
# add latitude column based on 'Name of State / UT' column
df_India['Latitude'] = df_India['Name of State / UT'].map(lat)

# add longitude column based on 'Name of State / UT' column
df_India['Longitude'] = df_India['Name of State / UT'].map(long)

df_India.head(36)
```

	Name of State / UT	Total Confirmed cases*	Cured/Discharged/Migrated	Deaths**	Date	Latitude	Longitu
0	Andaman and Nicobar Islands	33	33	0	2020-05-24	11.7401	92.65
1	Andhra Pradesh	2757	1809	56	2020-05-24	15.9129	79.74
2	Arunachal Pradesh	1	1	0	2020-05-24	28.2180	94.72
3	Assam	329	55	4	2020-05-24	26.2006	92.93

```
# rename columns

df_India = df_India.rename(columns={'Cured/Discharged/Migrated' :'Cured/Discharged',
                                    'Total Confirmed cases *': 'Confirmed',
                                    'Total Confirmed cases ': 'Confirmed',
                                    'Total Confirmed cases* ': 'Confirmed'})
df_India = df_India.rename(columns={'Cured/Discharged':'Cured'})
df_India = df_India.rename(columns={'Name of State / UT':'State/UnionTerritory'})
df_India = df_India.rename(columns={'Name of State / UT':'State/UnionTerritory'})

df_India = df_India.rename(columns=lambda x: re.sub('Total Confirmed cases \(Including .* for
                                                    'Total Confirmed cases',x))
df_India = df_India.rename(columns={'Deaths ( more than 70% cases due to comorbidities )':'De
                                    'Deaths**':'Deaths'})
```

2020

```
# unique state names
df_India['State/UnionTerritory'].unique()
```

```
array(['Andaman and Nicobar Islands', 'Andhra Pradesh',
       'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh',
       'Chhattisgarh', 'Dadar Nagar Haveli', 'Delhi', 'Goa', 'Gujarat',
       'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir', 'Jharkhand',
       'Karnataka', 'Kerala', 'Ladakh', 'Madhya Pradesh', 'Maharashtra',
       'Manipur', 'Meghalaya', 'Mizoram', 'Odisha', 'Puducherry',
       'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telengana',
       'Tripura', 'Uttarakhand', 'Uttar Pradesh', 'West Bengal'],
      dtype=object)
```

2020

```
# number of missing values
df_India.isna().sum()
```

```
State/UnionTerritory    0
Confirmed               0
Cured                   0
Deaths                  0
```

```
Date                    0
Latitude                0
Longitude               0
dtype: int64
```

```
# number of unique values
df_India.nunique()
```

```
State/UnionTerritory    34
Confirmed               32
Cured                   32
Deaths                  21
Date                     1
Latitude                34
Longitude               30
dtype: int64
```

Saving data

```
# saving data
# ===========

# file names as year-month-day.csv format
file_name = now.strftime("%Y_%m_%d")+' - COVID-19_India.csv'

# location for saving the file
file_loc = '/content/'

# save file as a scv file
df_India.to_csv(file_loc + file_name, index=False)

# df_India.head(36)
```

```
# fix datatype
df_India['Date'] = pd.to_datetime(df_India['Date'])
```

```
# rename state/UT names
df_India['State/UnionTerritory'].replace('Chattisgarh', 'Chhattisgarh', inplace=True)
df_India['State/UnionTerritory'].replace('Pondicherry', 'Puducherry', inplace=True)
```

Final dataframe

```
df_India.head(36)
```

	level_0	index	State/UnionTerritory	Confirmed	Cured	Deaths	Date	Latitude	Lon
0	0	33	Kerala	795	515	4	2020-05-24	10.8505	
1	1	32	Rajasthan	6742	3786	160	2020-05-24	27.0238	
2	2	31	Madhya Pradesh	6371	3267	281	2020-05-24	22.9734	
3	3	30	Uttar Pradesh	6017	3406	155	2020-05-24	26.8467	
4	4	29	Goa	55	16	0	2020-05-24	15.2993	
5	5	28	Ladakh	49	43	0	2020-05-24	34.2996	
6	6	27	Maharashtra	47190	13404	1577	2020-05-24	19.7515	
7	7	26	Jharkhand	350	141	4	2020-05-24	23.6102	
8	8	25	West Bengal	3459	1281	269	2020-05-24	22.9868	
9	9	24	Andaman and Nicobar Islands	33	33	0	2020-05-24	11.7401	
10	10	23	Assam	329	55	4	2020-05-24	26.2006	
11	11	22	Manipur	29	4	0	2020-05-24	24.6637	
12	12	21	Andhra Pradesh	2757	1809	56	2020-05-24	15.9129	
13	13	20	Puducherry	26	10	0	2020-05-24	11.9416	
14	14	19	Uttarakhand	244	56	2	2020-05-24	30.0668	
15	15	18	Bihar	2380	653	11	2020-05-24	25.0961	
16	16	17	Chandigarh	225	179	3	2020-05-24	30.7333	
17	17	16	Chhattisgarh	214	64	0	2020-05-24	21.2787	
18	18	15	Punjab	2045	1870	39	2020-05-24	31.1471	
19	19	14	Dadar Nagar Haveli	2	0	0	2020-	20.1809	

								05-24	
20	20	13	Karnataka	1959	608	42	2020-05-24	15.3173	7
21	21	12	Tripura	189	153	0	2020-05-24	23.9408	9

```
# complete data info
df_India.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34 entries, 0 to 33
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   level_0               34 non-null     int64         
 1   index                 34 non-null     int64         
 2   State/UnionTerritory  34 non-null     object        
 3   Confirmed             34 non-null     object        
 4   Cured                 34 non-null     object        
 5   Deaths                34 non-null     object        
 6   Date                  34 non-null     datetime64[ns]
 7   Latitude              34 non-null     float64       
 8   Longitude             34 non-null     float64       
dtypes: datetime64[ns](1), float64(2), int64(2), object(4)
memory usage: 2.7+ KB
```

Save as .csv file

```
# saving data
# ===========

# file names as year-month-day.csv format
file_name = now.strftime("%Y_%m_%d")+' - COVID-19_India_preprocessed.csv'

# location for saving the file
file_loc = '/content/'

# save file as a scv file
df_India.to_csv(file_loc + file_name, index=False)
```

```
#Learn how to read a .csv file by creating a dataframe using pandas
# Reading the datasets
df= pd.read_csv('/content/2020_05_24 - COVID-19_India_preprocessed.csv')
df_india = df.copy()
df
```

	level_0	index	State/UnionTerritory	Confirmed	Cured	Deaths	Date	Latitude	Lon
0	0	33	Kerala	795	515	4	2020-05-24	10.8505	
1	1	32	Rajasthan	6742	3786	160	2020-05-24	27.0238	
2	2	31	Madhya Pradesh	6371	3267	281	2020-05-24	22.9734	
3	3	30	Uttar Pradesh	6017	3406	155	2020-05-24	26.8467	
4	4	29	Goa	55	16	0	2020-05-24	15.2993	
5	5	28	Ladakh	49	43	0	2020-05-24	34.2996	
6	6	27	Maharashtra	47190	13404	1577	2020-05-24	19.7515	
7	7	26	Jharkhand	350	141	4	2020-05-24	23.6102	
8	8	25	West Bengal	3459	1281	269	2020-05-24	22.9868	
9	9	24	Andaman and Nicobar Islands	33	33	0	2020-05-24	11.7401	
10	10	23	Assam	329	55	4	2020-05-24	26.2006	
11	11	22	Manipur	29	4	0	2020-05-24	24.6637	
12	12	21	Andhra Pradesh	2757	1809	56	2020-05-24	15.9129	
13	13	20	Puducherry	26	10	0	2020-05-24	11.9416	
14	14	19	Uttarakhand	244	56	2	2020-05-24	30.0668	
15	15	18	Bihar	2380	653	11	2020-05-24	25.0961	
16	16	17	Chandigarh	225	179	3	2020-05-24	30.7333	
17	17	16	Chhattisgarh	214	64	0	2020-05-24	21.2787	
18	18	15	Punjab	2045	1870	39	2020-05-24	31.1471	
19	19	14	Dadar Nagar Haveli	2	0	0	2020-	20.1809	

									05-24	
20	20	13		Karnataka	1959	608	42	2020-05-24	15.3173	7

```
from google.colab import drive
drive.mount('/content/drive')
```

1.2 Analysing COVID19 Cases in India

```
total_cases = df['Confirmed'].sum()
print('Total number of confirmed COVID 2019 cases across India till date (23rd May, 2020):',
```

```
Total number of confirmed COVID 2019 cases across India till date (23rd May, 2020): 1295
```

```
#Learn how to highlight your dataframe
df_temp = df.drop(['Latitude', 'Longitude', 'Date', 'index', 'level_0'], axis = 1) #Removing
df_temp.style.background_gradient(cmap='Reds')
```

	State/UnionTerritory	Confirmed	Cured	Deaths
0	Kerala	795	515	4
1	Rajasthan	6742	3786	160
2	Madhya Pradesh	6371	3267	281
3	Uttar Pradesh	6017	3406	155
4	Goa	55	16	0
5	Ladakh	49	43	0
6	Maharashtra	47190	13404	1577
7	Jharkhand	359	141	4

```
today = now.strftime("%Y_%m_%d")
total_cured = df['Cured'].sum()
print("Total people who were cured as of "+today+" are: ", total_cured)
total_cases = df['Confirmed'].sum()
print("Total people who were detected COVID+ve as of "+today+" are: ", total_cases)
total_death = df['Deaths'].sum()
print("Total people who died due to COVID19 as of "+today+" are: ",total_death)
total_active = total_cases-total_cured-total_death
print("Total active COVID19 cases as of "+today+" are: ",total_active)
```

```
Total people who were cured as of 2020_05_24 are:  54441
Total people who were detected COVID+ve as of 2020_05_24 are:  129530
Total people who died due to COVID19 as of 2020_05_24 are:  3867
Total active COVID19 cases as of 2020_05_24 are:  71222
```

21	Tripura	189	153	0

```
#Total Active  is the Total cases - (Number of death + Cured)
df['Total Active'] = df['Confirmed'] - (df['Deaths'] + df1['Cured'])
total_active = df['Total Active'].sum()
print('Total number of active COVID 2019 cases across India:', total_active)
Tot_Cases = df.groupby('State/UnionTerritory')['Total Active'].sum().sort_values(ascending=Fa
Tot_Cases.style.background_gradient(cmap='Reds')
```

```
Total number of active COVID 2019 cases across India: 71222
```

State/UnionTerritory	Total Active
Maharashtra	32209
Tamil Nadu	7918
Gujarat	6666
Delhi	6412
Madhya Pradesh	2823
Rajasthan	2796
Uttar Pradesh	2456
West Bengal	1909
Bihar	1716
Karnataka	1309
Andhra Pradesh	892
Jammu and Kashmir	774
Odisha	765
Telengana	699
Haryana	365
Kerala	276
Assam	270
Jharkhand	205

```
import numpy as np
state_cases = df_india.groupby('State/UnionTerritory')['Confirmed','Deaths','Cured'].max().re

#state_cases = state_cases.astype({'Deaths': 'int'})
state_cases['Active'] = state_cases['Confirmed'] - (state_cases['Deaths']+state_cases['Cured'
state_cases["Death Rate (per 100)"] = np.round(100*state_cases["Deaths"]/state_cases["Confirm
state_cases["Cure Rate (per 100)"] = np.round(100*state_cases["Cured"]/state_cases["Confirmed
state_cases.sort_values('Confirmed', ascending= False).fillna(0).style.background_gradient(cm
                        .background_gradient(cmap='Blues',subset=["Deaths"])\
                        .background_gradient(cmap='Blues',subset=["Cured"])\
                        .background_gradient(cmap='Blues',subset=["Active"])\
                        .background_gradient(cmap='Blues',subset=["Death Rate (per 100)"])\
                        .background_gradient(cmap='Blues',subset=["Cure Rate (per 100)"])
```

	State/UnionTerritory	Confirmed	Deaths	Cured	Active	Death Rate (per 100)	Cure Rate (per 100)
19	Maharashtra	47190	1577	13404	32209	3.340000	28.400000
28	Tamil Nadu	15512	103	7491	7918	0.660000	48.290000
10	Gujarat	13664	829	6169	6666	6.070000	45.150000
8	Delhi	12910	231	6267	6412	1.790000	48.540000
26	Rajasthan	6742	160	3786	2796	2.370000	56.160000
18	Madhya Pradesh	6371	281	3267	2823	4.410000	51.280000
31	Uttar Pradesh	6017	155	3406	2456	2.580000	56.610000
33	West Bengal	3459	269	1281	1909	7.780000	37.030000
1	Andhra Pradesh	2757	56	1809	892	2.030000	65.610000
4	Bihar	2380	11	653	1716	0.460000	27.440000
25	Punjab	2045	39	1870	136	1.910000	91.440000
15	Karnataka	1959	42	608	1309	2.140000	31.040000
29	Telengana	1813	49	1065	699	2.700000	58.740000
13	Jammu and Kashmir	1569	21	774	774	1.340000	49.330000
23	Odisha	1269	7	497	765	0.550000	39.160000
11	Haryana	1131	16	750	365	1.410000	66.310000
16	Kerala	795	4	515	276	0.500000	64.780000
14	Jharkhand	350	4	141	205	1.140000	40.290000
3	Assam	329	4	55	270	1.220000	16.720000
32	Uttarakhand	244	2	56	186	0.820000	22.950000
5	Chandigarh	225	3	179	43	1.330000	79.560000
6	Chhattisgarh	214	0	64	150	0.000000	29.910000

Visualization Inference:

- Almost +1,611 cases of COVID-19 has been reported today (23rd May) taking total cases to 123202.
- The cases have been confirmed across 32 states and union territories.
- Out of 123202 cases, 51784 people have been cured, discharged or migrated.
- Maharashtra, Tamilnaidu, Gujrat and Delhi are worsely affected states with maximum number of confirmed cases
- Till 23rd of May 3720 people have died in India

27	Sikkim	1	0	0	1	0.000000	0.000000

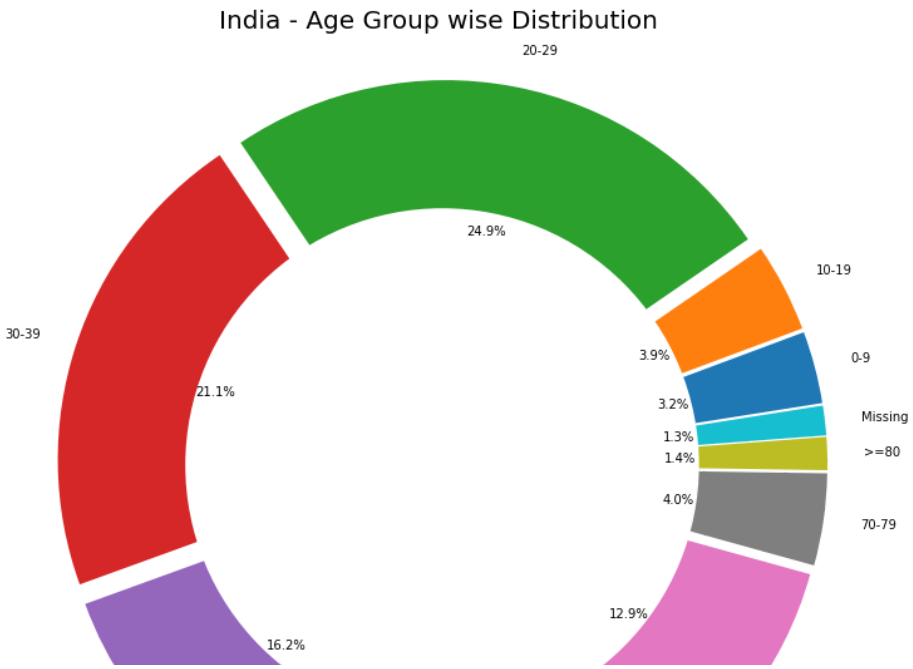
Finding more detail COVID Insights in India

```
age_details = pd.read_csv('/content/AgeGroupDetails.csv')
india_covid_19 = pd.read_csv('/content/covid_19_india.csv')
hospital_beds = pd.read_csv('/content/HospitalBedsIndia.csv')
individual_details = pd.read_csv('/content/IndividualDetails.csv')
ICMR_details = pd.read_csv('/content/ICMRTestingDetails.csv')
ICMR_labs = pd.read_csv('/content/ICMRTestingLabs.csv')
state_testing = pd.read_csv('/content/StatewiseTestingDetails.csv')
population = pd.read_csv('/content/population_india_census2011.csv')
```

```
india_covid_19['Date'] = pd.to_datetime(india_covid_19['Date'],dayfirst = True)
state_testing['Date'] = pd.to_datetime(state_testing['Date'])
ICMR_details['DateTime'] = pd.to_datetime(ICMR_details['DateTime'],dayfirst = True)
ICMR_details = ICMR_details.dropna(subset=['TotalSamplesTested', 'TotalPositiveCases'])
```

```
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/css
recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/
latest_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/c
```

```
labels = list(age_details['AgeGroup'])
sizes = list(age_details['TotalCases'])

explode = []

for i in labels:
    explode.append(0.05)

plt.figure(figsize= (15,10))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode =explode)
centre_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('India - Age Group wise Distribution',fontsize = 20)
plt.axis('equal')
plt.tight_layout()
```



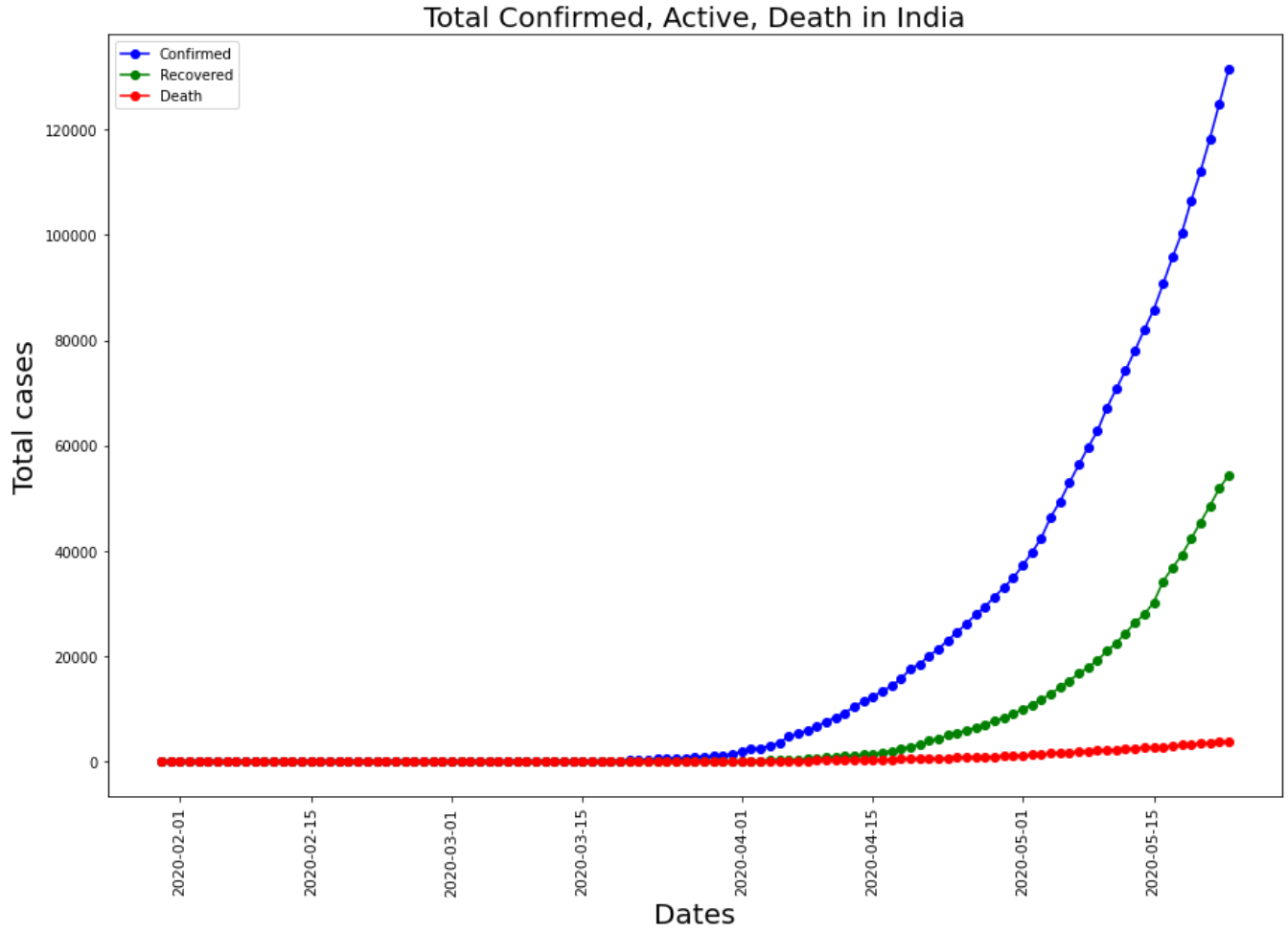
We could see that the age group <40 is the most affected which is against the trend which says elderly people are more at risk of being affected. Only 17% of people >60 are affected.

```
dates = list(confirmed_df.columns[4:])
dates = list(pd.to_datetime(dates))
dates_india = dates[8:]
```

```
tes = list(pd.to_datetime(dates))
dates_india = dates[8:]
df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

k = df1[df1['Country/Region']=='India'].loc[:,'1/30/20':]
india_confirmed = k.values.tolist()[0]

k = df2[df2['Country/Region']=='India'].loc[:,'1/30/20':]
india_deaths = k.values.tolist()[0]

k = df3[df3['Country/Region']=='India'].loc[:,'1/30/20':]
india_recovered = k.values.tolist()[0]

plt.figure(figsize= (15,10))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Total Confirmed, Active, Death in India" , fontsize = 20)

ax1 = plt.plot_date(y= india_confirmed,x= dates_india,label = 'Confirmed',linestyle ='-',colo
ax2 = plt.plot_date(y= india_recovered,x= dates_india,label = 'Recovered',linestyle ='-',colo
```

```
ax3 = plt.plot_date(y= india_deaths,x= dates_india,label = 'Death',linestyle ='-',color = 'r'
plt.legend()
```

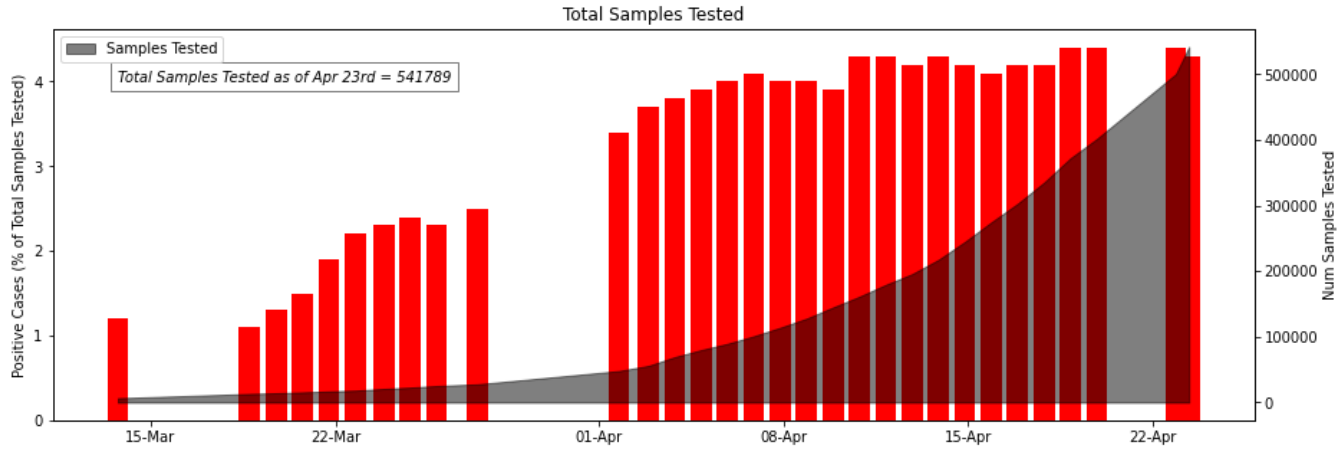
```
<matplotlib.legend.Legend at 0x7f93e19d16a0>
```



Total Samples Tested

```
import matplotlib.dates as mdates
ICMR_details['Percent_positive'] = round((ICMR_details['TotalPositiveCases']/ICMR_details['To

fig, ax1 = plt.subplots(figsize= (15,5))
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
ax1.set_ylabel('Positive Cases (% of Total Samples Tested)')
ax1.bar(ICMR_details['DateTime'] , ICMR_details['Percent_positive'], color="red",label = 'Per
```

```
ax1.text(ICMR_details[ 'DateTime' ][0],4,  'Total Samples Tested as of Apr 23rd = 541789 ', style
        bbox={'facecolor': 'white' ,'alpha': 0.5, 'pad': 5})

ax2 = ax1.twinx()
ax2.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
ax2.set_ylabel('Num Samples Tested')
ax2.fill_between(ICMR_details['DateTime'],ICMR_details['TotalSamplesTested'],color = 'black',

plt.legend(loc="upper left")
plt.title('Total Samples Tested')
plt.show()
```



Testing LIVE Status

```
import json
# get response from the web page
response = requests.get('https://api.covid19india.org/state_test_data.json')

# get contents from the response
content = response.content

# parse the json file
parsed = json.loads(content)

# keys
parsed.keys()
```

```
dict_keys(['states_tested_data'])
```

```
# save data in a dataframe
tested = pd.DataFrame(parsed['states_tested_data'])
```

```
# first few rows
tested.tail()
```

	coronaenquirycalls	cumulativepeopleinquarantine	negative	numcallsstatehelpline
1346				
1347				
1348				
1349				
1350				

```
# fix datatype
tested['updatedon'] = pd.to_datetime(tested['updatedon'])
```

```
# save file as a scv file
tested.to_csv('updated_tests_latest_state_level.csv', index=False)
```

```
state_test_cases = tested.groupby(['updatedon','state'])['totaltested','populationncp2019proj
```

```
state_test_cases.head(36)
```

	updatedon	state	totaltested	populationncp2019projection	testpositivityrate
0	01/04/2020	Delhi	2621	19814000	0.00%
1	01/04/2020	Kerala	7965	35125000	3.33%
2	01/04/2020	West Bengal	659	96906000	5.61%
3	01/05/2020	Andaman and Nicobar Islands	3754	397000	0.88%
4	01/05/2020	Andhra Pradesh	102460	52221000	1.43%
5	01/05/2020	Arunachal Pradesh	724	1504000	0.28%
6	01/05/2020	Assam		34293000	
7	01/05/2020	Bihar	24118	119520000	1.93%
8	01/05/2020	Chandigarh	1252	1179000	7.03%
9	01/05/2020	Chhattisgarh	18039	28724000	0.24%
10	01/05/2020	Delhi		19814000	
11	01/05/2020	Goa	2181	1540000	0.32%
12	01/05/2020	Gujarat	68774	67936000	6.86%
13	01/05/2020	Haryana	30191	28672000	1.18%
14	01/05/2020	Himachal Pradesh	6472	7300000	0.62%
15	01/05/2020	Jammu and Kashmir	21695	13203000	2.95%
16	01/05/2020	Jharkhand	11771	37403000	0.96%
17	01/05/2020	Karnataka	64898	65798000	0.91%
18	01/05/2020	Kerala	27150	35125000	1.83%
19	01/05/2020	Ladakh	2430	293000	0.91%
20	01/05/2020	Madhya Pradesh	44116	82232000	6.15%
21	01/05/2020	Maharashtra	144159	122153000	7.28%
22	01/05/2020	Manipur	461	3103000	0.43%
23	01/05/2020	Mizoram	180	1192000	0.56%
24	01/05/2020	Nagaland	664	2150000	0.00%
25	01/05/2020	Odisha	34133	43671000	0.44%
26	01/05/2020	Puducherry	2698	1504000	0.33%

20	01/05/2020	Puducherry	2698	1504000	0.55%

```
state_test_cases = tested.groupby('state')['totaltested','populationncp2019projection','testp
state_test_cases['testpositivityrate'] = state_test_cases['testpositivityrate'].str.replace('
```

```
state_test_cases = state_test_cases.apply(pd.to_numeric)
```

30	01/05/2020	Tripura	4828	3992000	0.06%

```
state_test_cases.nunique()
```

```
totaltested                          35
populationncp2019projection          34
testpositivityrate                   32
testsperpositivecase                 23
testsperthousand                     34
totalpeoplecurrentlyinquarantine     19
dtype: int64
```

35	02/04/2020	Assam	962	34293000	1.66%

```
state_test_cases.sort_values('totaltested', ascending= False).style.background_gradient(cmap=
                      .background_gradient(cmap='Blues',subset=["populationncp2019projectio
                      .background_gradient(cmap='Blues',subset=["testpositivityrate"])\
                      .background_gradient(cmap='Blues',subset=["testsperpositivecase"])\
                      .background_gradient(cmap='Blues',subset=["testsperthousand"])\
                      .background_gradient(cmap='Blues',subset=["totalpeoplecurrentlyinquar
```

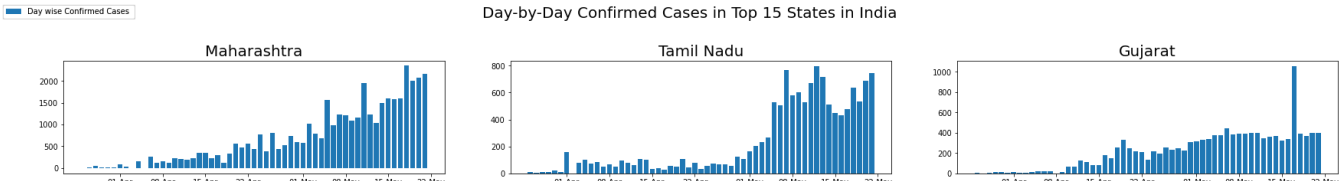
state	totaltested	populationncp2019projection	testpositivityrate	testsperpositivecase	testsp
Madhya Pradesh	99677	82232000	9.770000	21.000000	1.5300
Uttar Pradesh	98300	224979000	5.290000	41.000000	0.8500
Karnataka	98081	65798000	2.790000	97.000000	2.6500
Rajasthan	97790	77264000	2.880000	49.000000	3.5700
Haryana	97006	28672000	5.770000	87.000000	3.0700
Jammu and Kashmir	96826	13203000	6.990000	74.000000	8.1100
Maharashtra	95210	122153000	9.070000	26.000000	2.6200
Andhra Pradesh	94558	52221000	7.330000	94.000000	5.2800

Day-by-Day Confirmed Cases in Top 15 States in India

Uttarakhand	9915	11141000	2.480000	99.000000	1.3900

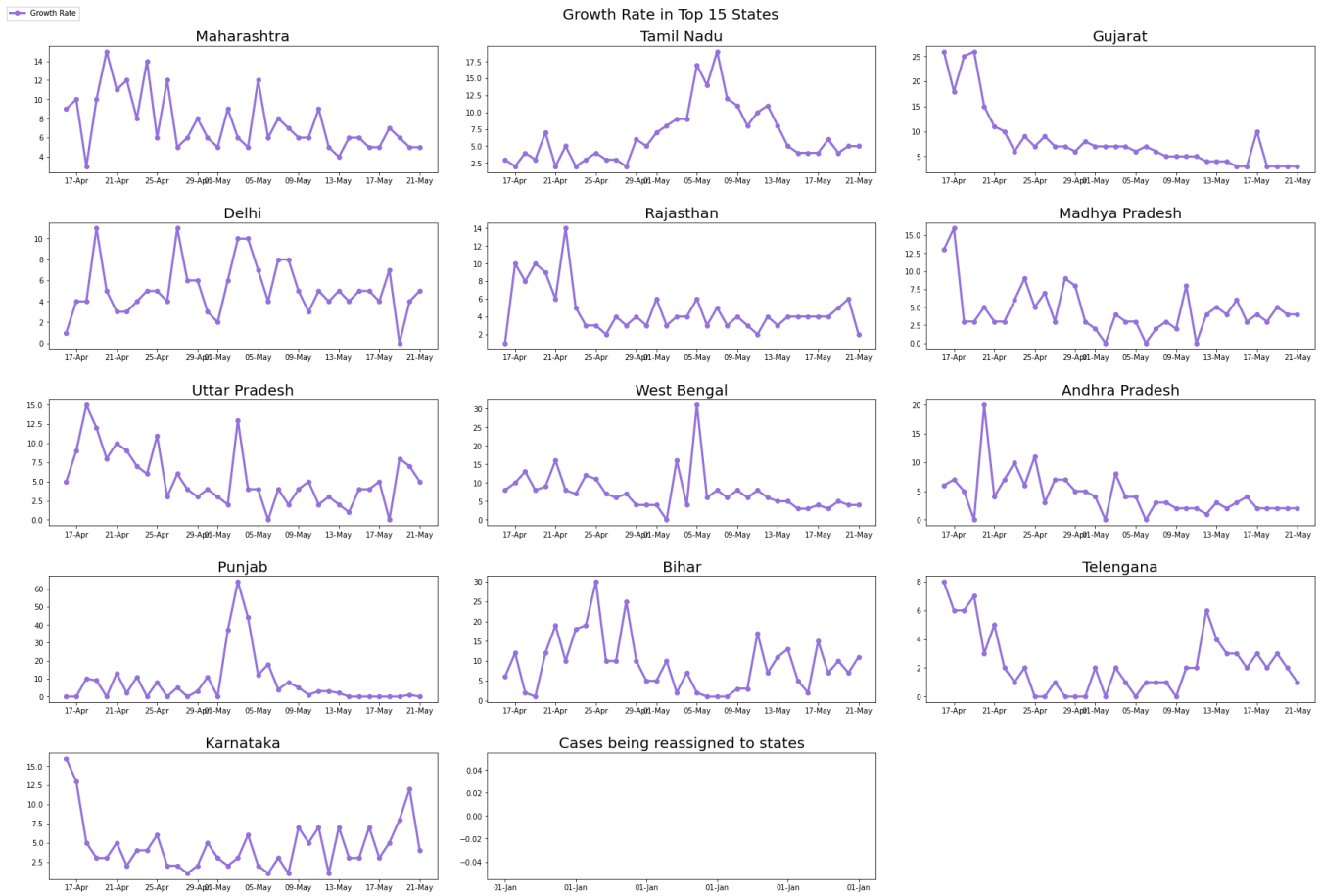
```
all_state = list(df_India['State/UnionTerritory'].unique())

latest = india_covid_19[india_covid_19['Date'] > '24-03-20']
state_cases = latest.groupby('State/UnionTerritory')['Confirmed','Deaths','Cured'].max().rese
latest['Active'] = latest['Confirmed'] - (latest['Deaths']- latest['Cured'])
state_cases = state_cases.sort_values('Confirmed', ascending= False).fillna(0)
states =list(state_cases['State/UnionTerritory'][0:15])

states_confirmed = {}
states_deaths = {}
states_recovered = {}
states_dates = {}

for state in states:
    df = latest[latest['State/UnionTerritory'] == state].reset_index()
    k = []
    l = []
    m = []
    n = []
    for i in range(1,len(df)):
        k.append(df['Confirmed'][i]-df['Confirmed'][i-1])
        l.append(df['Deaths'][i]-df['Deaths'][i-1])
        m.append(df['Cured'][i]-df['Cured'][i-1])
        n.append(df['Active'][i]-df['Active'][i-1])
    states_confirmed[state] = k
    states_deaths[state] = l
    states_recovered[state] = m
    states_active[state] = n
    date = list(df['Date'])
    states_dates[state] = date[1:]

fig = plt.figure(figsize= (25,17))
plt.suptitle('Day-by-Day Confirmed Cases in Top 15 States in India',fontsize = 20,y=1.0)
```

```
k=0
for i in range(1,15):
    ax = fig.add_subplot(5,3,i)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
    ax.bar(states_dates[states[k]],states_confirmed[states[k]],label = 'Day wise Confirmed Ca
    plt.title(states[k],fontsize = 20)
    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper left')
    k=k+1
plt.tight_layout(pad=5.0)
```



Growth Rate in top 15 States in India

```
def calc_growthRate(values):
    k = []
    for i in range(1,len(values)):
        summ = 0
        for j in range(i):
            summ = summ + values[j]
        rate = (values[i]/summ)*100
        k.append(int(rate))
    return k

fig = plt.figure(figsize= (25,17))
plt.suptitle('Growth Rate in Top 15 States',fontsize = 20,y=1.0)
k=0
for i in range(1,15):
    ax = fig.add_subplot(5,3,i)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
    #ax.bar(states_dates[states[k]],states_confirmed[states[k]],label = 'Day wise Confirmed C
    growth_rate = calc_growthRate(states_confirmed[states[k]])
    ax.plot_date(states_dates[states[k]][21:],growth_rate[20:],color = '#9370db',label = 'Gro
    plt.title(states[k],fontsize = 20)
    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper left')
    k=k+1
plt.tight_layout(pad=3.0)
```



Though being highly populated the relative confimred cases of India is low compared to other countries. This could be because of two reasons:

- 67 days lockdown imposed by prime minister Narendra Modi in several stages (Source : Health Ministry)
- Low testing rate (Source: news18)

Exploring different types of hospital beds available in India during lockdown

```
cols_object = list(hospital_beds.columns[2:8])

for cols in cols_object:
    hospital_beds[cols] = hospital_beds[cols].astype(int,errors = 'ignore')

hospital_beds = hospital_beds.drop('Sno',axis=1)
```

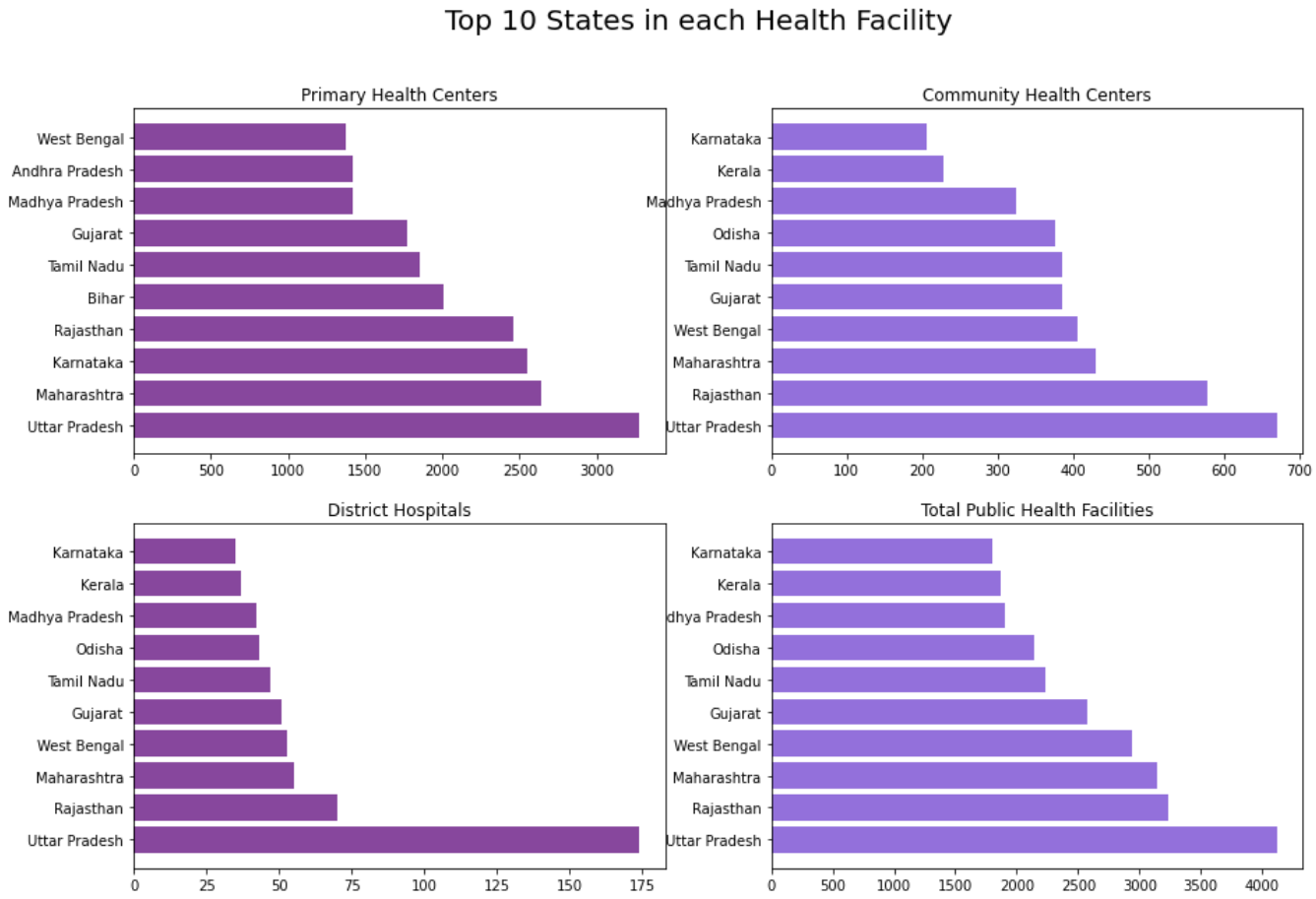
```
hospital_beds.head(36)
```

	State/UT	NumPrimaryHealthCenters_HMIS	NumCommunityHealthCenters_HMIS	NumSubDi
0	Andaman & Nicobar Islands	27	4	
1	Andhra Pradesh	1417	198	
2	Arunachal Pradesh	122	62	
3	Assam	1007	166	
4	Bihar	2007	63	
5	Chandigarh	40	2	
6	Chhattisgarh	813	166	
7	Dadra & Nagar Haveli	9	2	
8	Daman & Diu	4	2	
9	Delhi	534	25	
10	Goa	31	4	
11	Gujarat	1770	385	
12	Haryana	500	131	
13	Himachal Pradesh	516	79	
14	Jammu & Kashmir	702	87	
15	Jharkhand	343	179	
16	Karnataka	2547	207	
17	Kerala	933	229	
18	Lakshadweep	4	3	
19	Madhya Pradesh	1420	324	
20	Maharashtra	2638	430	
21	Manipur	87	17	
22	Meghalaya	138	29	
23	Mizoram	65	10	
24	Nagaland	134	21	
25	Odisha	1360	377	

26	Puducherry	40	4
27	Punjab	521	146
28	Rajasthan	2463	579
29	Sikkim	25	2
30	Tamil Nadu	1854	385
31	Telangana	788	82
32	Tripura	114	22
33	Uttar Pradesh	3277	671
34	Uttarakhand	275	69
35	West Bengal	1374	406

Exploring top 10 States in each health facilities

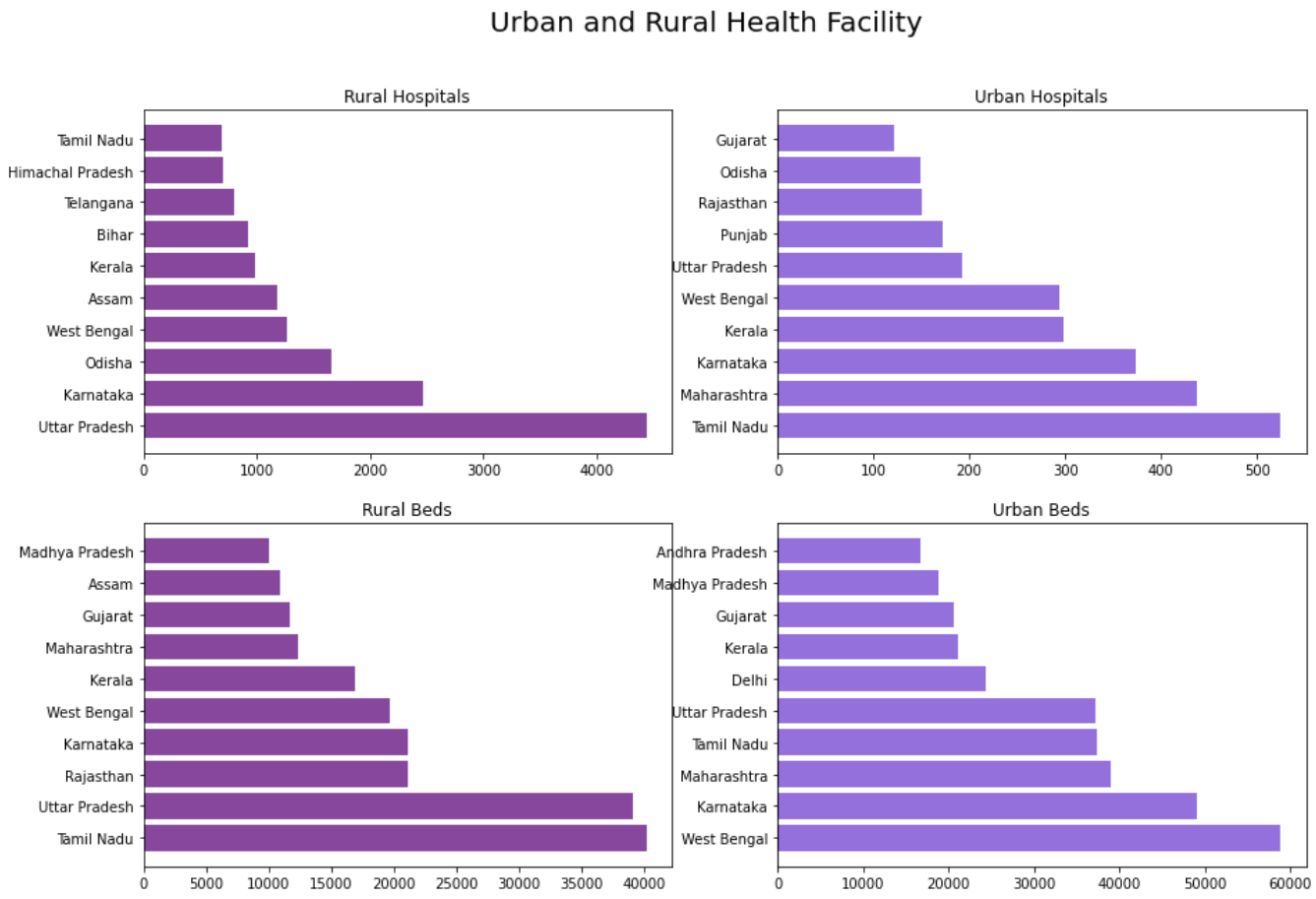
```
top_10_primary = hospital_beds.nlargest(10,'NumPrimaryHealthCenters_HMIS')
top_10_community = hospital_beds.nlargest(10,'NumCommunityHealthCenters_HMIS')
top_10_district_hospitals = hospital_beds.nlargest(10,'NumDistrictHospitals_HMIS')
top_10_public_facility = hospital_beds.nlargest(10,'TotalPublicHealthFacilities_HMIS')
top_10_public_beds = hospital_beds.nlargest(10,'NumPublicBeds_HMIS')

plt.figure(figsize=(15,10))
plt.suptitle('Top 10 States in each Health Facility',fontsize=20)
plt.subplot(221)
plt.title('Primary Health Centers')
plt.barh(top_10_primary['State/UT'],top_10_primary['NumPrimaryHealthCenters_HMIS'],color ='#8

plt.subplot(222)
plt.title('Community Health Centers')
plt.barh(top_10_community['State/UT'],top_10_community['NumCommunityHealthCenters_HMIS'],colo

plt.subplot(224)
plt.title('Total Public Health Facilities')
plt.barh(top_10_community['State/UT'],top_10_public_facility['TotalPublicHealthFacilities_HMI

plt.subplot(223)
plt.title('District Hospitals')
plt.barh(top_10_community['State/UT'],top_10_district_hospitals['NumDistrictHospitals_HMIS'],
```



Exploring Urban and Rural Healthcare Facility

```
top_rural_hos = hospital_beds.nlargest(10,'NumRuralHospitals_NHP18')
top_rural_beds = hospital_beds.nlargest(10,'NumRuralBeds_NHP18')
top_urban_hos = hospital_beds.nlargest(10,'NumUrbanHospitals_NHP18')
top_urban_beds = hospital_beds.nlargest(10,'NumUrbanBeds_NHP18')

plt.figure(figsize=(15,10))
plt.suptitle('Urban and Rural Health Facility',fontsize=20)
plt.subplot(221)
plt.title('Rural Hospitals')
plt.barh(top_rural_hos['State/UT'],top_rural_hos['NumRuralHospitals_NHP18'],color = '#87479d'

plt.subplot(222)
plt.title('Urban Hospitals')
plt.barh(top_urban_hos['State/UT'],top_urban_hos['NumUrbanHospitals_NHP18'],color = '#9370db'

plt.subplot(223)
plt.title('Rural Beds')
plt.barh(top_rural_beds['State/UT'],top_rural_beds['NumRuralBeds_NHP18'],color = '#87479d');
```

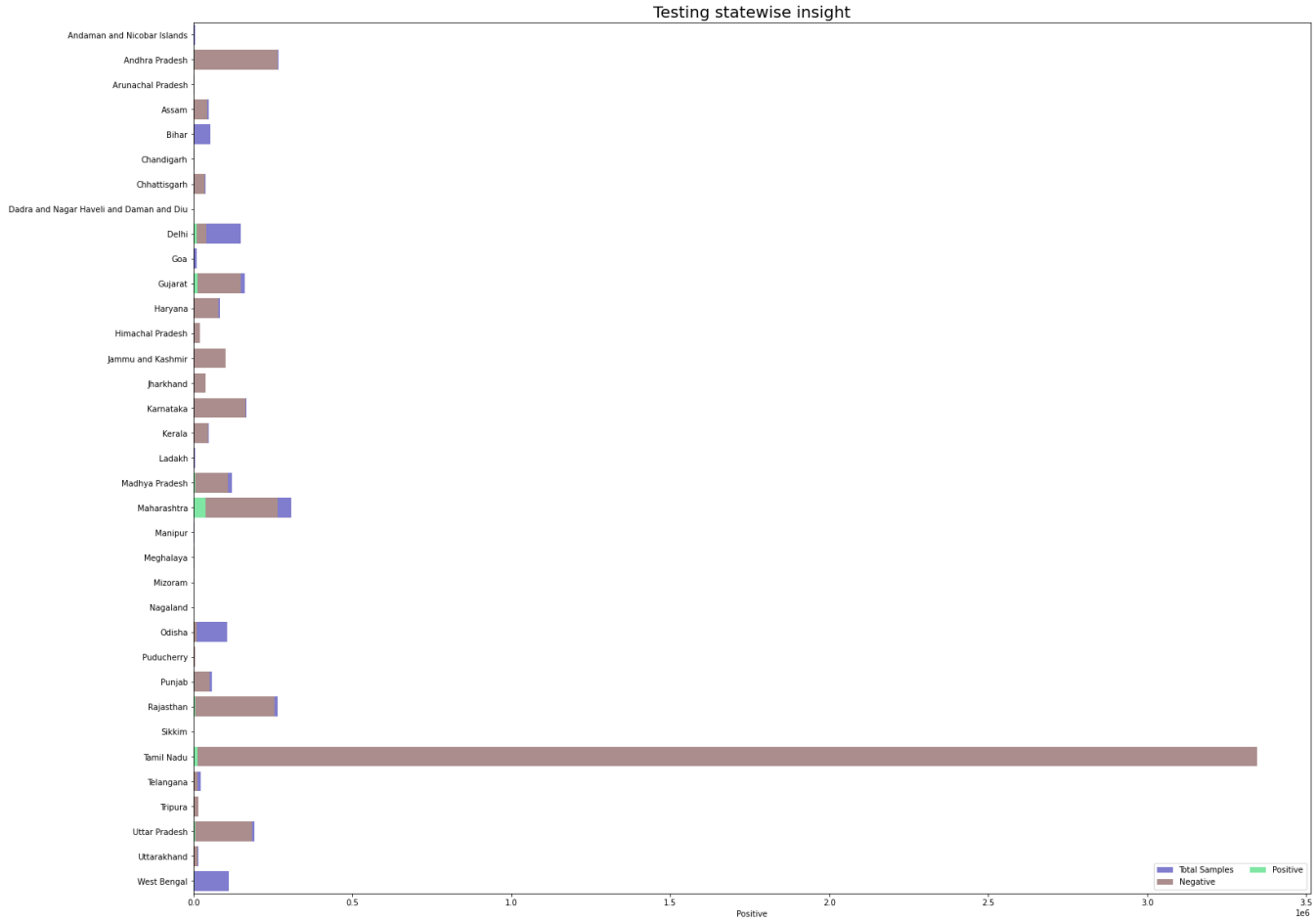
```
plt.subplot(224)
plt.title('Urban Beds')
plt.barh(top_urban_beds['State/UT'],top_urban_beds['NumUrbanBeds_NHP18'],color = '#9370db');
```



Exploring Statewise Testing Insights

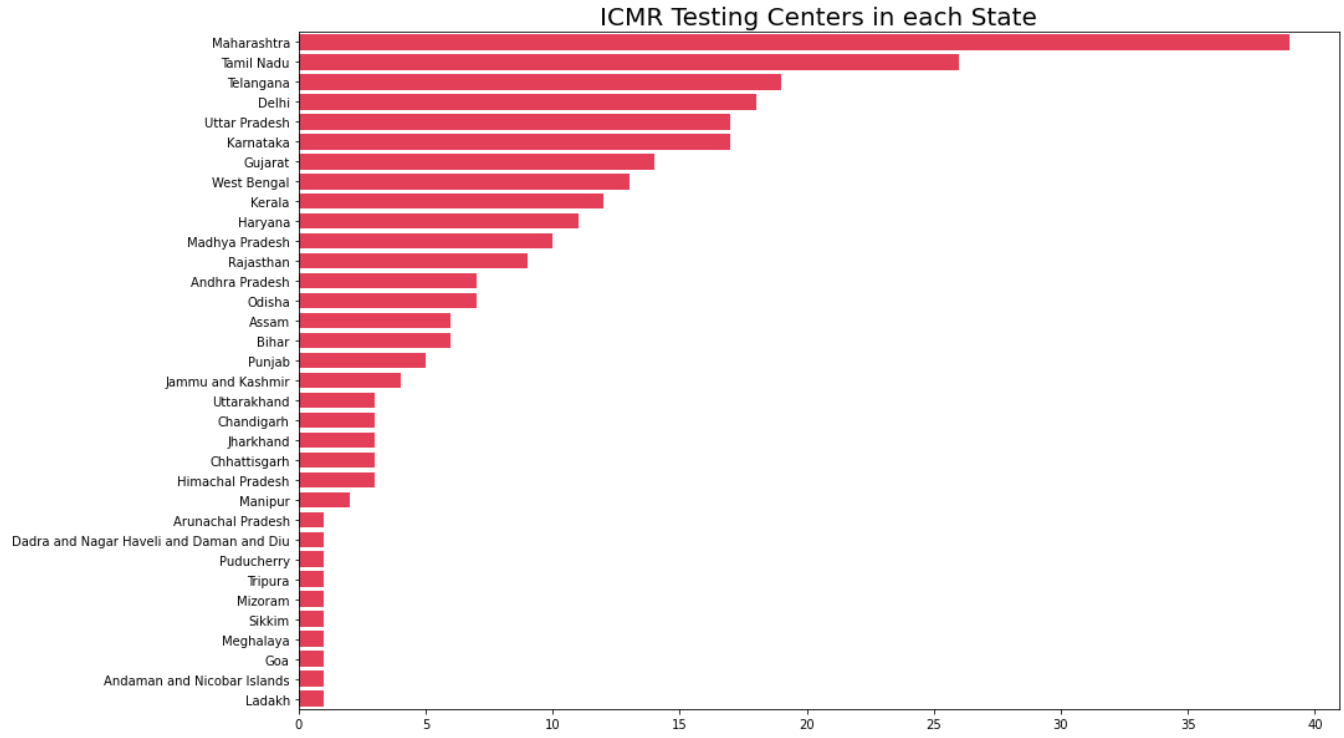
```
state_test = pd.pivot_table(state_testing, values=['TotalSamples','Negative','Positive'], ind
state_names = list(state_test.index)
state_test['State'] = state_names

plt.figure(figsize=(25,20))
sns.set_color_codes("pastel")
sns.barplot(x="TotalSamples", y= state_names, data=state_test,label="Total Samples", color =
sns.barplot(x='Negative', y=state_names, data=state_test,label='Negative', color= '#af8887')
```

```
sns.barplot(x='Positive', y=state_names, data=state_test,label='Positive', color='#6ff79d')
plt.title('Testing statewise insight',fontsize = 20)
plt.legend(ncol=2, loc="lower right", frameon=True);
```



Number of ICMR Testing Centres in each state

```
values = list(ICMR_labs['state'].value_counts())
names = list(ICMR_labs['state'].value_counts().index)

plt.figure(figsize=(15,10))
sns.set_color_codes("pastel")
plt.title('ICMR Testing Centers in each State', fontsize = 20)
sns.barplot(x= values, y= names,color = '#ff2345');
```



Let's Start with the predictions

```
train = pd.read_csv('/content/train.csv')
test = pd.read_csv('/content/test.csv')
train['Date'] = pd.to_datetime(train['Date'])
```

```
test['Date'] = pd.to_datetime(test['Date'])
```

Prophet

Prophet is open source software released by Facebook's Core Data Science team. It is available for download on CRAN and PyPI.

We use Prophet, a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Why Prophet?

- **Accurate and fast:** Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. Facebook finds it to perform better than any other approach in the majority of cases. It fit models in Stan so that you get forecasts in just a few seconds.
- **Fully automatic:** Get a reasonable forecast on messy data with no manual effort. Prophet is robust to outliers, missing data, and dramatic changes in your time series.
- **Tunable forecasts:** The Prophet procedure includes many possibilities for users to tweak and adjust forecasts. You can use human-interpretable parameters to improve your forecast by adding your domain knowledge
- **Available in R or Python:** Facebook has implemented the Prophet procedure in R and Python. Both of them share the same underlying Stan code for fitting. You can use whatever language you're comfortable with to get forecasts.

References

- https://facebook.github.io/prophet/
- https://facebook.github.io/prophet/docs/
- https://github.com/facebook/prophet
- https://facebook.github.io/prophet/docs/quick_start.html

```
!pip install Prophet
```

```
Collecting Prophet
  Downloading https://files.pythonhosted.org/packages/1b/1c/61c969e76119a8257220869a6b3
Requirement already satisfied: pytz>=2014.9 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: pandas>=0.15.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.8.0 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.6/dist-packages (
```

```
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-p
Building wheels for collected packages: Prophet
  Building wheel for Prophet (setup.py) ... done
  Created wheel for Prophet: filename=prophet-0.1.1-cp36-none-any.whl size=12170 sha256=
  Stored in directory: /root/.cache/pip/wheels/77/3e/f3/1c536bf1f871f818686e7fbf31cab18c
Successfully built Prophet
Installing collected packages: Prophet
Successfully installed Prophet-0.1.1
```

```
from fbprophet import Prophet
from fbprophet.plot import plot_plotly, add_changepoints_to_plot

k = df1[df1['Country/Region']=='India'].loc[:,'1/22/20':]
india_confirmed = k.values.tolist()[0]
data = pd.DataFrame(columns = ['ds','y'])
data['ds'] = dates
data['y'] = india_confirmed
```

The input to Prophet is always a dataframe with two columns: **ds** and **y**. The **ds (datestamp)** column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

```
confirmed.columns = ['ds','y']
#confirmed['ds'] = confirmed['ds'].dt.date
confirmed['ds'] = pd.to_datetime(confirmed['ds'])
```

4.1 Forecasting Confirmed NCOVID-19 Cases Worldwide with Prophet (Base model)

Generating a week ahead forecast of confirmed cases of NCOVID-19 using Prophet, with 95% prediction interval by creating a base model with no tweaking of seasonality-related parameters and additional regressors.

```
prop = Prophet(interval_width=0.95)
prop.fit(data)
future = prop.make_future_dataframe(periods=15)
future.tail(15)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to c
```

	ds
123	2020-05-24
124	2020-05-25
125	2020-05-26
126	2020-05-27
127	2020-05-28
128	2020-05-29
129	2020-05-30
130	2020-05-31
131	2020-06-01
132	2020-06-02
133	2020-06-03
134	2020-06-04

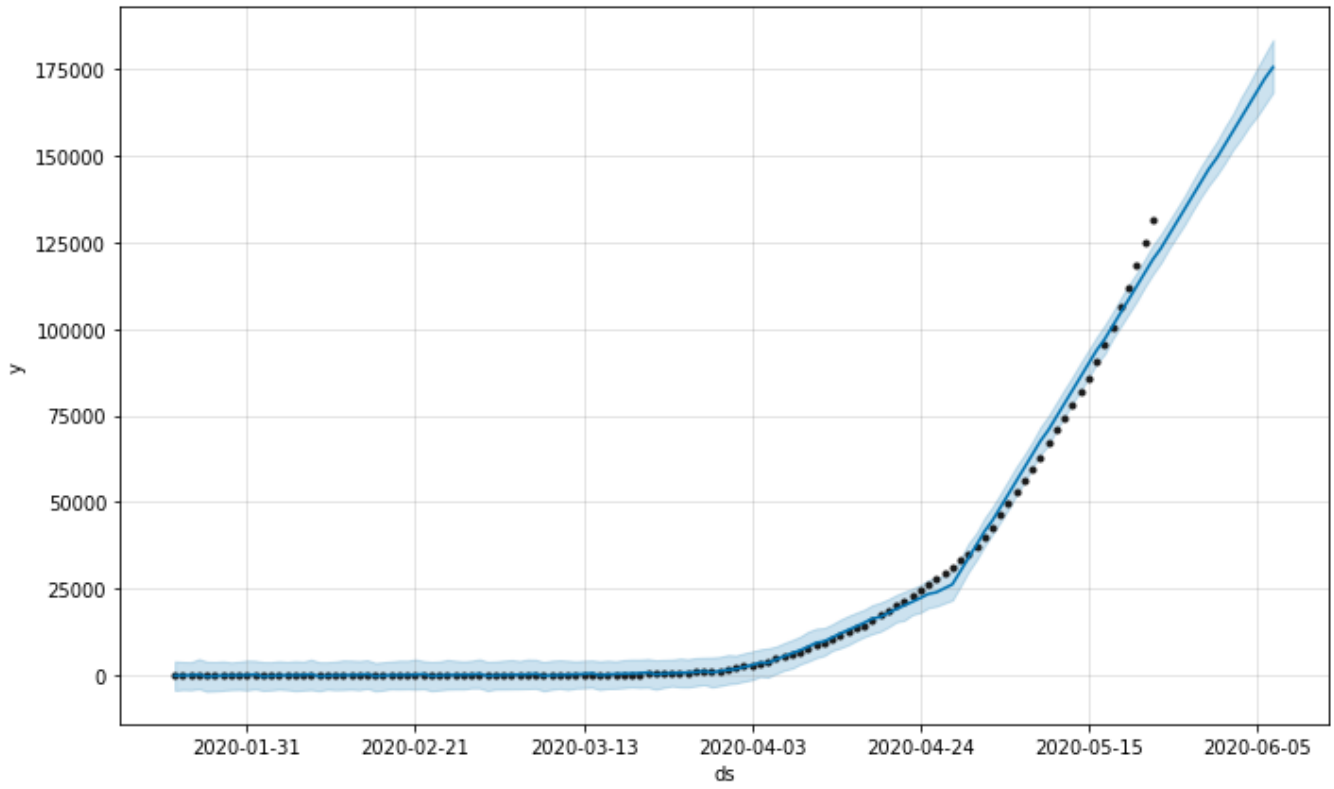
The **predict** method will assign each row in future a predicted value which it names **yhat**. If you pass in historical dates, it will provide an in-sample fit. The **forecast object** here is a new dataframe that includes a column yhat with the forecast, as well as columns for components and uncertainty intervals.

```
#predicting the future with date, and upper and lower limit of y value
forecast = prop.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

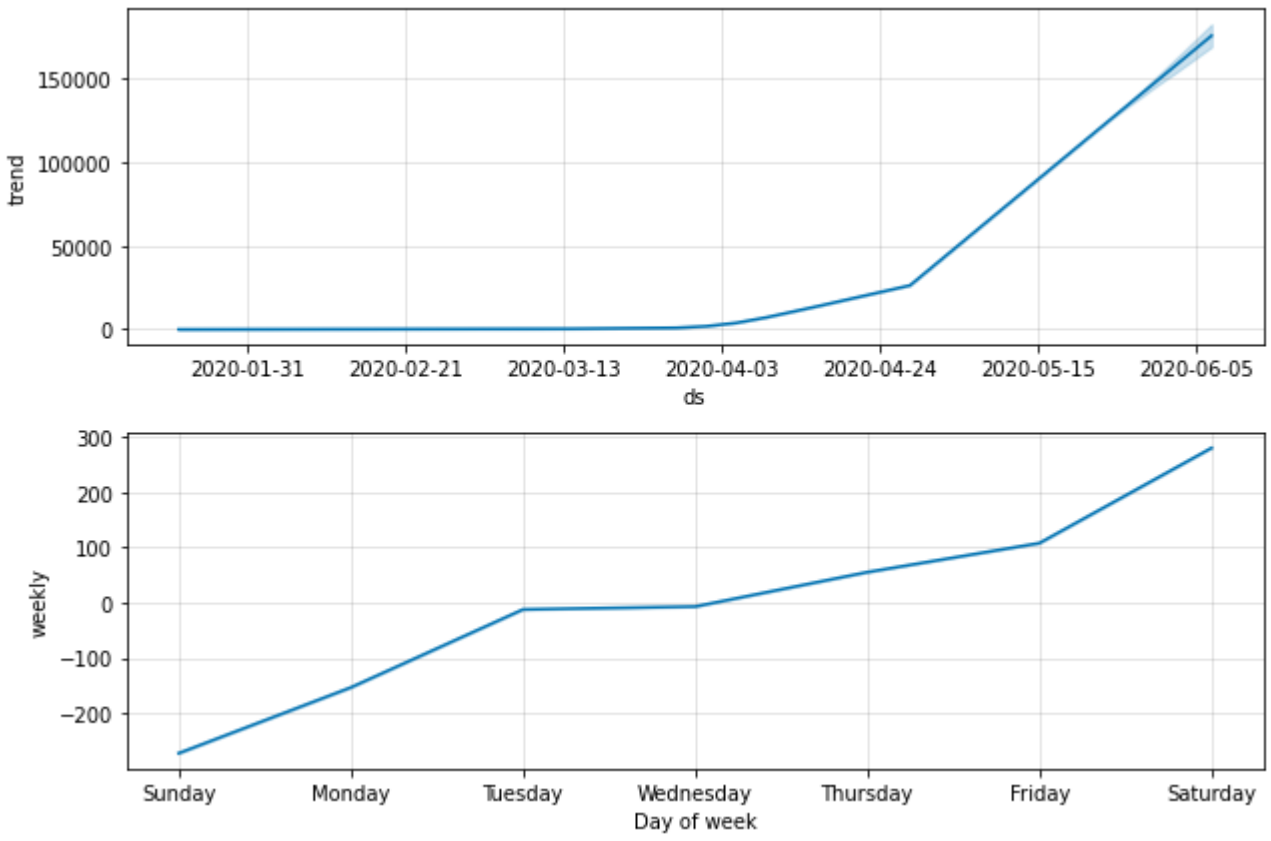
	ds	yhat	yhat_lower	yhat_upper
133	2020-06-03	160907.626979	154956.600188	166912.732577
134	2020-06-04	164711.645426	158501.310683	170909.673351
135	2020-06-05	168505.959102	161449.108222	175271.167154
136	2020-06-06	172419.829233	164934.966298	179445.394979
137	2020-06-07	175609.868008	168394.216054	183564.223453

You can plot the forecast by calling the Prophet.plot method and passing in your forecast dataframe.

```
confirmed_forecast_plot = prop.plot(forecast)
```

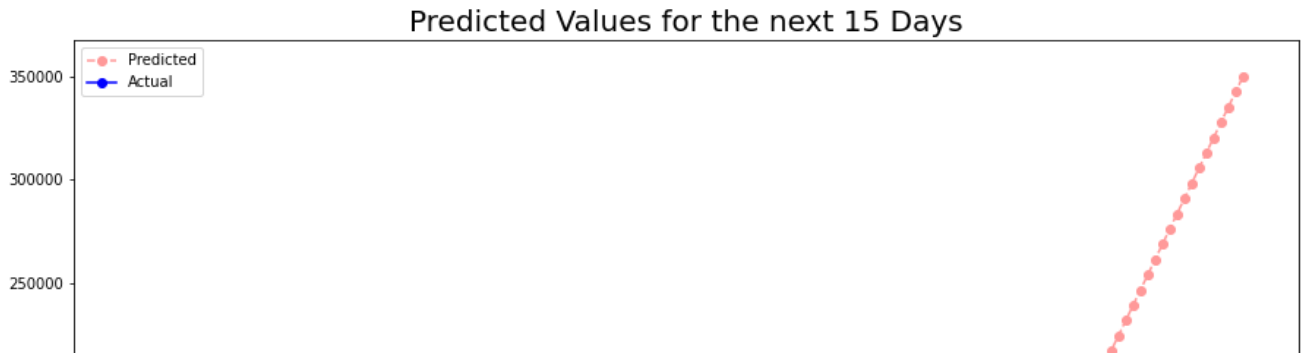



```
confirmed_forecast_plot =prop.plot_components(forecast)
```



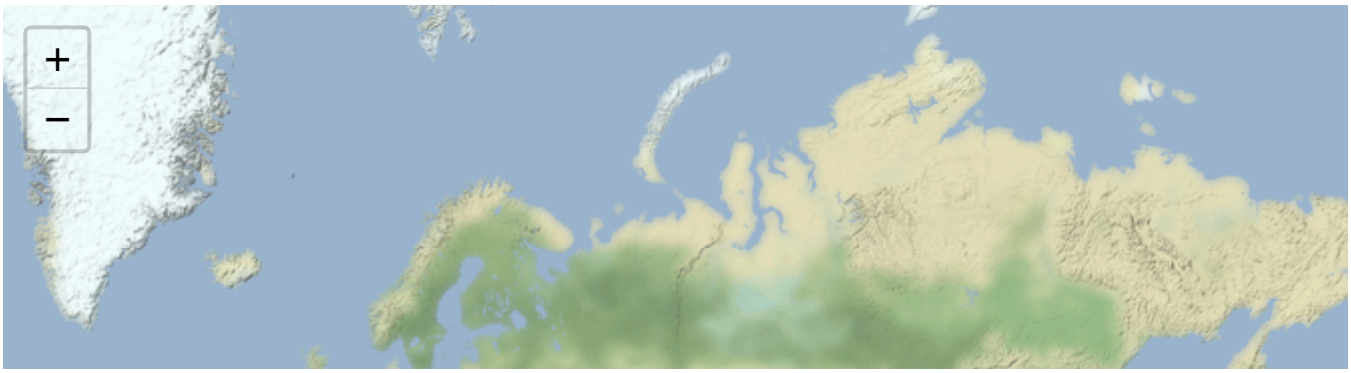
ARIMA Model

```
from statsmodels.tsa.arima_model import ARIMA

from datetime import timedelta

arima = ARIMA(data['y'], order=(5, 1, 0))
arima = arima.fit(trend='c', full_output=True, disp=True)
forecast = arima.forecast(steps= 30)
pred = list(forecast[0])

start_date = data['ds'].max()
prediction_dates = []
for i in range(30):
    date = start_date + timedelta(days=1)
    prediction_dates.append(date)
    start_date = date
plt.figure(figsize= (15,10))
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Predicted Values for the next 15 Days" , fontsize = 20)

plt.plot_date(y= pred,x= prediction_dates,linestyle ='dashed',color = '#ff9999',label = 'Pred
plt.plot_date(y=data['y'],x=data['ds'],linestyle = '-',color = 'blue',label = 'Actual');
plt.legend();
```

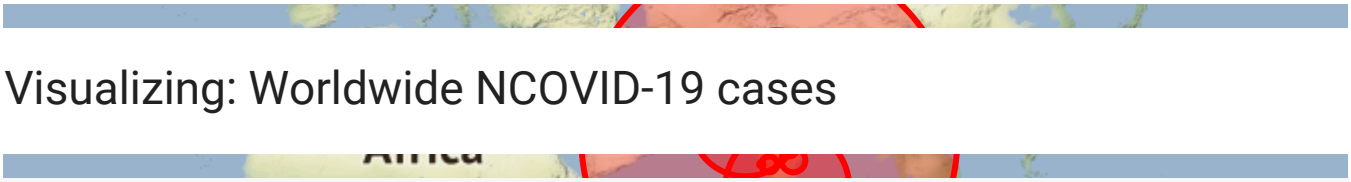


1.4 Visualising the spread geographically


```
# Learn how to use folium to create a zoomable map
map = folium.Map(location=[20, 70], zoom_start=4,tiles='Stamenterrain')

for lat, lon, value, name in zip(df_india['Latitude'], df_india['Longitude'], df_india['Confi
    folium.CircleMarker([lat, lon], radius=value*0.002, popup = ('<strong>State</strong>: ' +
map
```

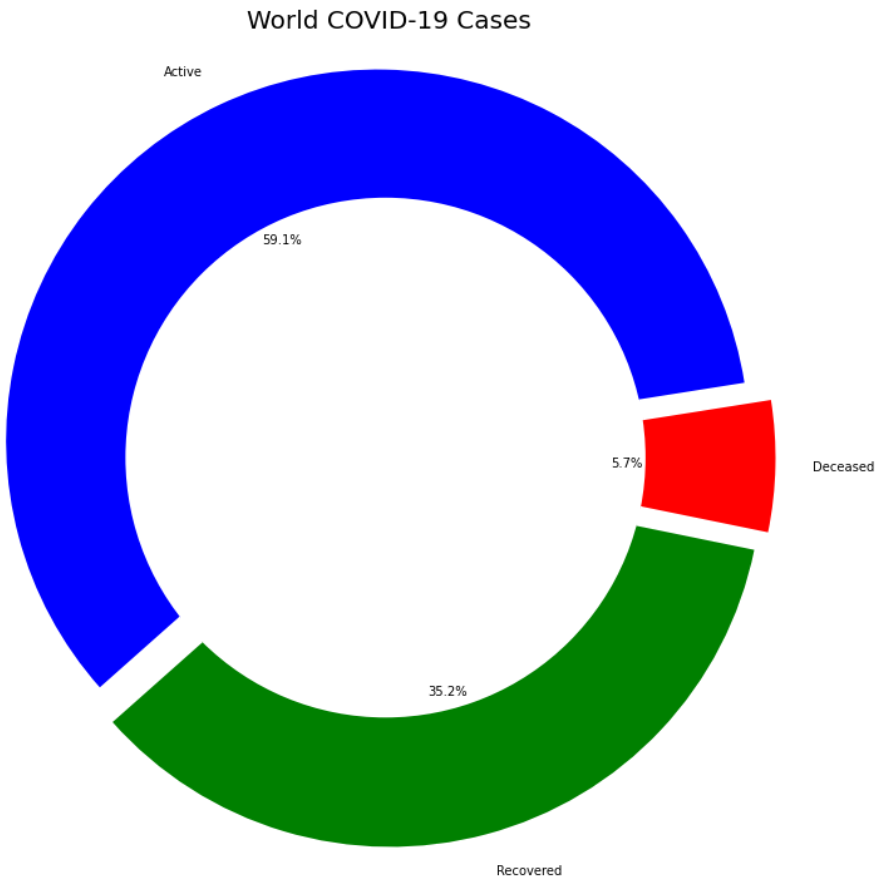


Part 3: Exploring World wide data



3.1 Visualizing: Worldwide NCOVID-19 cases


```
world_confirmed = confirmed_df[confirmed_df.columns[-1:]].sum()
world_recovered = recovered_df[recovered_df.columns[-1:]].sum()
world_deaths = deaths_df[deaths_df.columns[-1:]].sum()
world_active = world_confirmed - (world_recovered - world_deaths)

labels = ['Active','Recovered','Deceased']
sizes = [world_active,world_recovered,world_deaths]
color= ['blue','green','red']
explode = []

for i in labels:
    explode.append(0.05)

plt.figure(figsize= (15,10))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode = explode,colors = col
centre_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('World COVID-19 Cases',fontsize = 20)
plt.axis('equal')
plt.tight_layout()
```



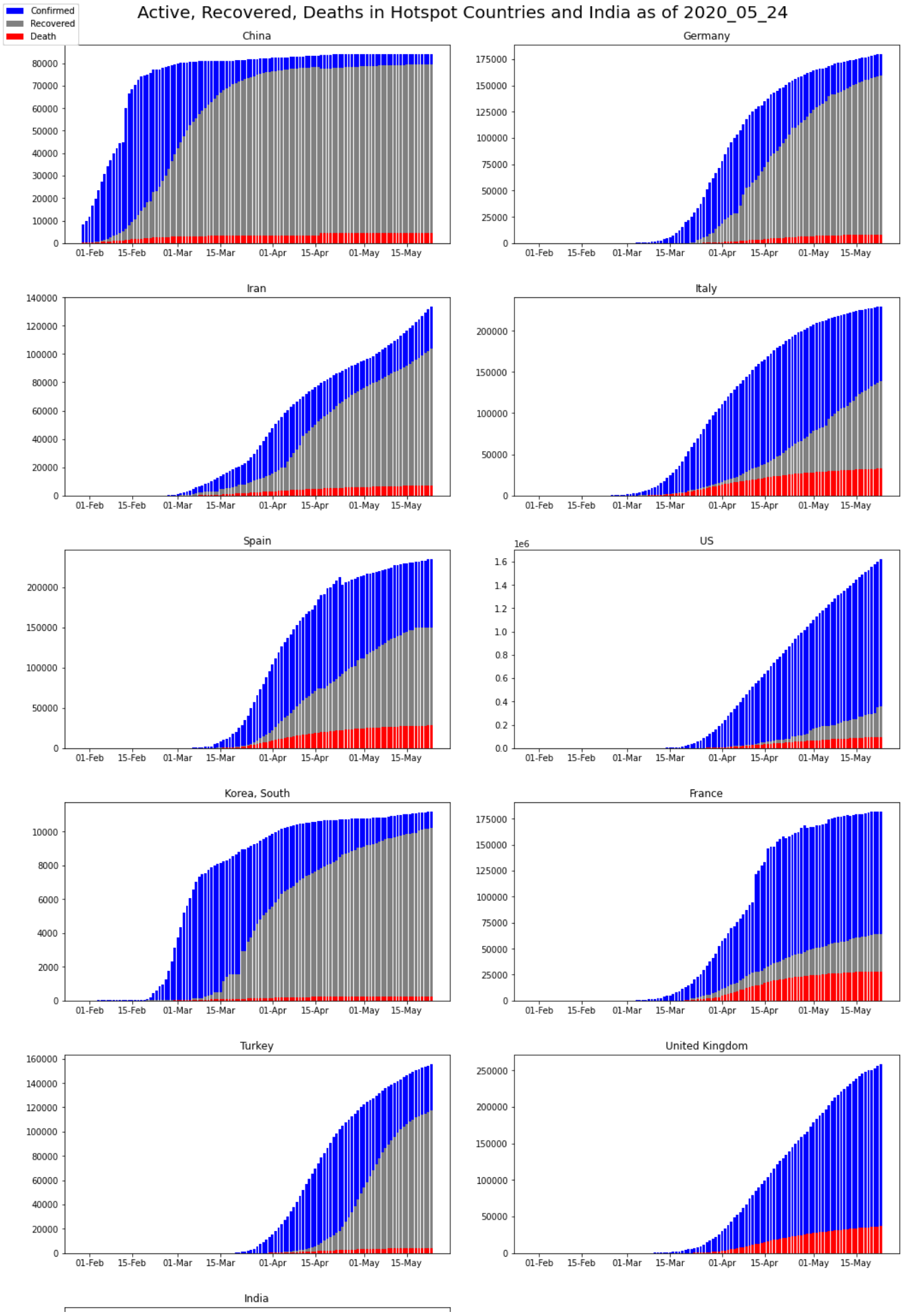

```
dates
```

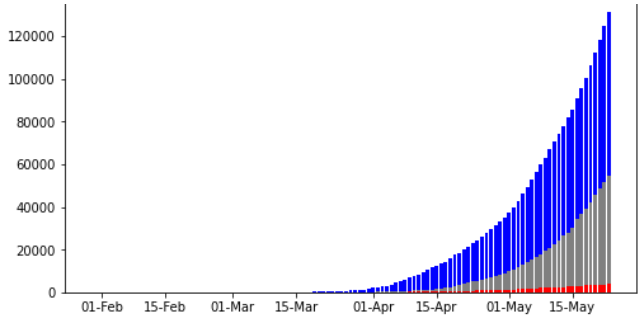
```
[Timestamp('2020-01-22 00:00:00'),
 Timestamp('2020-01-23 00:00:00'),
 Timestamp('2020-01-24 00:00:00'),
 Timestamp('2020-01-25 00:00:00'),
 Timestamp('2020-01-26 00:00:00'),
 Timestamp('2020-01-27 00:00:00'),
 Timestamp('2020-01-28 00:00:00'),
 Timestamp('2020-01-29 00:00:00'),
 Timestamp('2020-01-30 00:00:00'),
 Timestamp('2020-01-31 00:00:00'),
 Timestamp('2020-02-01 00:00:00'),
 Timestamp('2020-02-02 00:00:00'),
 Timestamp('2020-02-03 00:00:00'),
 Timestamp('2020-02-04 00:00:00'),
 Timestamp('2020-02-05 00:00:00'),
 Timestamp('2020-02-06 00:00:00'),
 Timestamp('2020-02-07 00:00:00'),
 Timestamp('2020-02-08 00:00:00'),
 Timestamp('2020-02-09 00:00:00'),
 Timestamp('2020-02-10 00:00:00'),
 Timestamp('2020-02-11 00:00:00'),
 Timestamp('2020-02-12 00:00:00'),
```

```
        Timestamp('2020-02-13 00:00:00'),
        Timestamp('2020-02-14 00:00:00'),
        Timestamp('2020-02-15 00:00:00'),
        Timestamp('2020-02-16 00:00:00'),
        Timestamp('2020-02-17 00:00:00'),
        Timestamp('2020-02-18 00:00:00'),
        Timestamp('2020-02-19 00:00:00'),
        Timestamp('2020-02-20 00:00:00'),
        Timestamp('2020-02-21 00:00:00'),
        Timestamp('2020-02-22 00:00:00'),
        Timestamp('2020-02-23 00:00:00'),
        Timestamp('2020-02-24 00:00:00'),
        Timestamp('2020-02-25 00:00:00'),
        Timestamp('2020-02-26 00:00:00'),
        Timestamp('2020-02-27 00:00:00'),
        Timestamp('2020-02-28 00:00:00'),
        Timestamp('2020-02-29 00:00:00'),
        Timestamp('2020-03-01 00:00:00'),
        Timestamp('2020-03-02 00:00:00'),
        Timestamp('2020-03-03 00:00:00'),
        Timestamp('2020-03-04 00:00:00'),
        Timestamp('2020-03-05 00:00:00'),
        Timestamp('2020-03-06 00:00:00'),
        Timestamp('2020-03-07 00:00:00'),
        Timestamp('2020-03-08 00:00:00'),
        Timestamp('2020-03-09 00:00:00'),
        Timestamp('2020-03-10 00:00:00'),
        Timestamp('2020-03-11 00:00:00'),
        Timestamp('2020-03-12 00:00:00'),
        Timestamp('2020-03-13 00:00:00'),
        Timestamp('2020-03-14 00:00:00'),
        Timestamp('2020-03-15 00:00:00'),
        Timestamp('2020-03-16 00:00:00'),
        Timestamp('2020-03-17 00:00:00'),
        Timestamp('2020-03-18 00:00:00'),
        Timestamp('2020-03-19 00:00:00'),
        Timestamp('2020-03-20 00:00:00'),
```

```
hotspots = ['China','Germany','Iran','Italy','Spain','US','Korea, South','France','Turkey','U
dates = list(confirmed_df.columns[4:])
dates = list(pd.to_datetime(dates))
dates_india = dates[8:]

df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

global_confirmed = {}
global_deaths = {}
global_recovered = {}
global_active= {}

for country in hotspots:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed[country] = k.values.tolist()[0]
```

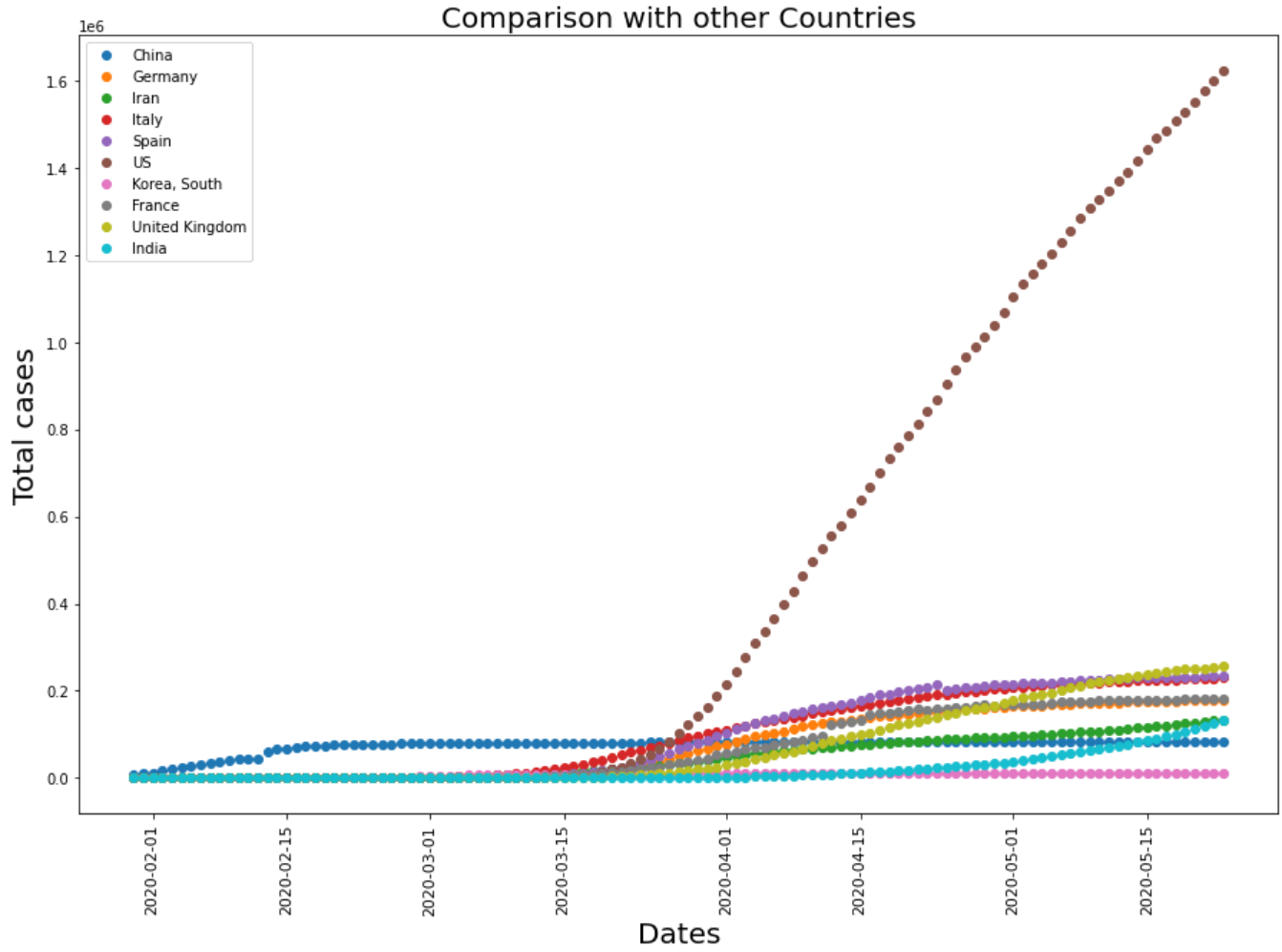
```
    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths[country] = k.values.tolist()[0]

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_recovered[country] = k.values.tolist()[0]

# for country in hotspots:
#     k = list(map(int.__sub__, global_confirmed[country], global_deaths[country]))
#     global_active[country] = list(map(int.__sub__, k, global_recovered[country]))

fig = plt.figure(figsize= (15,25))
plt.suptitle('Active, Recovered, Deaths in Hotspot Countries and India as of '+ today,fontsiz
#plt.legend()
k=0
for i in range(1,12):
    ax = fig.add_subplot(6,2,i)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
    # ax.bar(dates_india,global_active[hotspots[k]],color = 'green',alpha = 0.6,label = 'Acti
    ax.bar(dates_india,global_confirmed[hotspots[k]],color='blue',label = 'Confirmed');
    ax.bar(dates_india,global_recovered[hotspots[k]],color='grey',label = 'Recovered');
    ax.bar(dates_india,global_deaths[hotspots[k]],color='red',label = 'Death');
    plt.title(hotspots[k])
    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper left')
    k=k+1

plt.tight_layout(pad=3.0)
```


Active, Recovered, Deaths in Hotspot Countries and India as of 2020_05_24
Confirmed
Recovered
Death
China
Germany
Iran
Italy
Spain
US
1e6
Korea, South
France
Turkey
United Kingdom
India
01-Feb
15-Feb
01-Mar
15-Mar
01-Apr
15-Apr
01-May
15-May

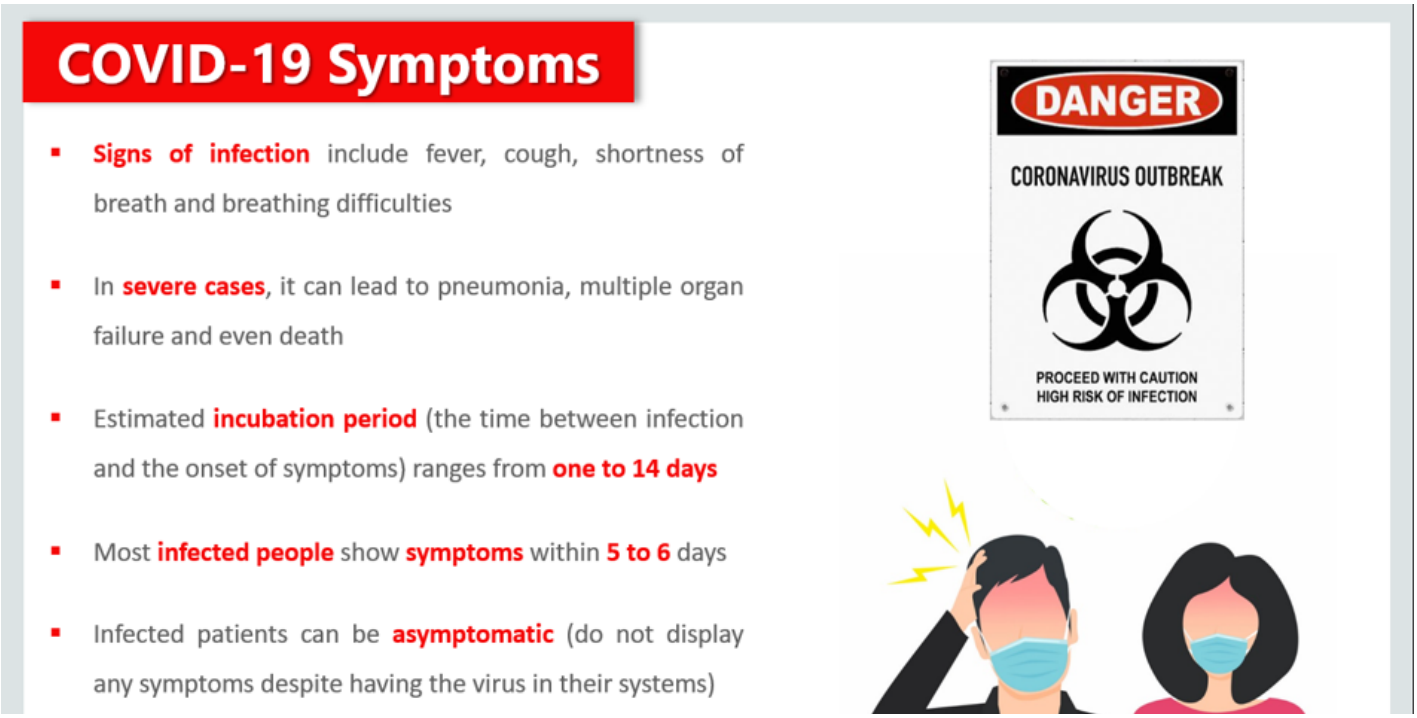



```
countries = ['China','Germany','Iran','Italy','Spain','US','Korea, South','France','United Ki

global_confirmed = []
global_recovered = []
global_deaths = []

for country in countries:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed.append(k.values.tolist()[0])

    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])
```

```
plt.figure(figsize= (15,10))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Comparison with other Countries" , fontsize = 20)

for i in range(len(countries)):
    plt.plot_date(y= global_confirmed[i],x= dates_india,label = countries[i])
plt.legend();
```



COVID-19 Symptoms



Data Source:

- https://www.mohfw.gov.in/
- https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset
- https://www.worldometers.info/coronavirus/#countries
- https://api.covid19india.org/

DIY: Assignment

The latest data can also be extracted from the available APIs and reading the json. Below are the API list that have been provided by crowd sourced. Extract and use these data to find meaningful insights.

- National time series, statewise stats and test counts
- State-district-wise State-district-wise V2
- Travel history
- Raw data
- States Daily changes
- Statewise Tested Numbers

Extracting data from Hirokuapp

```
api = pd.read_json('https://corona-virus-stats.herokuapp.com/api/v1/cases/countries-search')
```

```
json_data = api['data']['rows']
```

```
data = json_normalize(json_data)
data
```

	country	country_abbreviation	total_cases	new_cases	total_deaths	new_deaths	tot
0	World		4,525,103	3,077	303,351	269	
1	USA	US	1,457,593	0	86,912	0	
2	Spain	ES	272,646	0	27,321	0	
3	Russia	RU	252,245	0	2,305	0	
4	UK	GB	233,151	0	33,614	0	
5	Italy	IT	223,096	0	31,368	0	
6	Brazil	BR	203,165	247	13,999	6	
7	France	FR	178,870	0	27,425	0	
8	Germany	DE	174,975	0	7,928	0	
9	Turkey	TR	144,749	0	4,007	0	

Collecting Data for Statewise Insights

[ ] ↳ *15 cells hidden*

Collecting Some more Statewise Data

```
# get response from the web page
response = requests.get('https://api.covid19india.org/state_test_data.json')

# get contents from the response
content = response.content

# parse the json file
parsed = json.loads(content)

# keys
parsed.keys()
```

```
# get response from the web page
response = requests.get('https://api.covid19india.org/state_test_data.json')

# get contents from the response
content = response.content

# parse the json file
parsed = json.loads(content)

# keys
parsed.keys()
```

```
# save data in a dataframe
th = pd.DataFrame(parsed['states_tested_data'])

# first few rows
th
```

```
th.columns
```

```
# save to csv`
th.to_csv('tests_latest_state_level.csv', index=False)
```

```
# to get web contents
import requests
# to parse json contents
import json
# to parse csv files
import csv
```

```

```

Zones

```
# get response from the web page
response = requests.get('https://api.covid19india.org/zones.json')

# get contents from the response
content = response.content

# parse the json file
parsed = json.loads(content)

# keys
parsed.keys()
```

```
dict_keys(['zones'])
```

```
zo = pd.DataFrame(parsed['zones'])
zo.head()
```

	district	districtcode	lastupdated	source	st
0	Nicobars	AN_Nicobars	01/05/2020	https://www.facebook.com/airnewsalerts/photos/...	Anda Nic Isla
1	North and Middle Andaman	AN_North and Middle Andaman	01/05/2020	https://www.facebook.com/airnewsalerts/photos/...	Anda Nic Isla

```
# save to csv`
zo.to_csv('zones.csv', index=False)
```

National level daily

```
response = requests.get('https://api.covid19india.org/data.json')
content = response.content
parsed = json.loads(content)
parsed.keys()
```

```
dict_keys(['cases_time_series', 'statewise', 'tested'])
```

```
national = pd.DataFrame(parsed['cases_time_series'])
national.head()
```

	dailyconfirmed	dailydeceased	dailyrecovered	date	totalconfirmed	totaldecease
0	1	0	0	30 January	1	
1	0	0	0	31 January	1	
2	0	0	0	01 February	1	
3	1	0	0	02	2	

```
national.columns
```

```
Index(['dailyconfirmed', 'dailydeceased', 'dailyrecovered', 'date',
       'totalconfirmed', 'totaldeceased', 'totalrecovered'],
```

```
      dtype='object')
```

```
national = national[['date', 'totalconfirmed', 'totaldeceased', 'totalrecovered',
                     'dailyconfirmed', 'dailydeceased', 'dailyrecovered']]
national.head()
```

	date	totalconfirmed	totaldeceased	totalrecovered	dailyconfirmed	dailydecease
0	30 January	1	0	0	1	
1	31 January	1	0	0	0	
2	01 February	1	0	0	0	
3	02	2	0	0	1	

```
# save to csv`
national.to_csv('nation_level_daily.csv', index=False)
```

National level latest

```
state_level = pd.DataFrame(parsed['statewise'])
state_level.head()
```

	active	confirmed	deaths	deltaconfirmed	deltadeaths	deltarecovered	lastupdatedt
0	72621	130110	3823	5316	97	1825	23/05/2(19:57
1	32209	47190	1577	2608	60	821	23/05/2(19:57
2	7917	15512	104	759	5	363	23/05/2(18:32
3	6591	13273	802	0	0	0	22/05/2(20:51
4	6412	12910	231	591	23	370	23/05/2(14:56

```
state_level.columns
```

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