Tracking Coronavirus(COVID-19) Spread in India using Python

Fetching the latest Statewise data of COVID-19 cases from the official website of Ministry of Health and visualizing it using Python libraries.

Pratik Nabriya
Mar 30 · 6 min read ★

Photo by Patrick Assalé on Unsplash

Coronavirus or COVID-19 needs no introduction. It has already been declared as a

both health perspective and an economic one. Plenty has been written about it, especially statistical reports on its exponential growth and the importance of *"flattening the curve"*.

As of now, most of us are staying and working from home to avoid the spread of corona virus. I decided to utilize the surplus time to write a Python Script that pulls the latest Statewise data of COVID-19 cases from the official website of Ministry of Health and Family Welfare, Government of India and turn it into insightful visualizations using popular Python packages like GeoPandas, Seaborn and Matplotlib.

Packages used

- Beautifulsoup — A library for pulling data out of *html* and *xml* files.
- Requests — A library for making HTTP requests in python.
- GeoPandas — A library for working with geospatial data in python.
- PrettyTable — quick and easy to represent tabular data in visually appealing ASCII tables.
- and other regular packages like Pandas, Matplotlib and Seaborn.

If you don't have any of the above mentioned packages installed on your system, please follow the installation instructions that are mentioned in the respective links.

(Note that Geopandas further depends on fiona for file access and descartes and matplotlib for plotting)

Scrape the Data

To scrape a website using Python, you need to perform these four basic steps:

1. Sending an HTTP GET request to the URL of the webpage that you want to scrape, which will respond with the HTML content. We can do this by using the *Request* library of Python.
2. Analyzing the HTML tags and their attributes, such as class, id, and other HTML tag attributes. Also, identifying your HTML tags where your content lives.
3. Fetching and parsing the data using *Beautifulsoup* library and maintain the data in some data structure such as Dictionary or List.

4. Output data in any file format such as csv, xlsx, json, etc. or use this tabulated data to make visualizations using *Seaborn/Matplotlib* libraries.

If you are new to web scraping, check this blog for the step-by-step explanation that can help you get started with web scarping using Python.

Jump into the Code

Import necessary libraries —

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from bs4 import BeautifulSoup
import geopandas as gpd
from prettytable import PrettyTable
```

Web Scraping —

```
url = 'https://www.mohfw.gov.in/'

# make a GET request to fetch the raw HTML content
web_content = requests.get(url).content

# parse the html content
soup = BeautifulSoup(web_content, "html.parser")

# remove any newlines and extra spaces from left and right
extract_contents = lambda row: [x.text.replace('\n', '') for x in
row]

# find all table rows and data cells within
stats = []
all_rows = soup.find_all('tr')
```

```
for row in all_rows:
    stat = extract_contents(row.find_all('td'))

# notice that the data that we require is now a list of length 5
    if len(stat) == 5:
        stats.append(stat)

#now convert the data into a pandas dataframe for further
processing

new_cols = ["Sr.No",
"States/UT","Confirmed","Recovered","Deceased"]
state_data = pd.DataFrame(data = stats, columns = new_cols)
state_data.head()
```

Output:

	Sr.No	States/UT	Confirmed	Recovered	Deceased
0	1	Andhra Pradesh	23	1	0
1	2	Andaman and Nicobar Islands	9	0	0
2	3	Bihar	15	0	1
3	4	Chandigarh	8	0	0
4	5	Chhattisgarh	7	0	0

Before we proceed further, notice that the scraped data columns are actually of 'string' datatype. We need to convert them into 'int' datatype.

```
state_data['Confirmed'] = state_data['Confirmed'].map(int)
state_data['Recovered'] = state_data['Recovered'].map(int)
state_data['Deceased'] = state_data['Deceased'].map(int)
```

You may also choose to present the data using PrettyTable

```
table = PrettyTable()
table.field_names = (new_cols)

for i in stats:
    table.add_row(i)

table.add_row(["","Total",
               sum(state data['Confirmed']),
```

```
print(table)
```

Output:

Sr.No	States/UT	Confirmed	Recovered	Deceased
1	Andhra Pradesh	23	1	0
2	Andaman and Nicobar Islands	9	0	0
3	Bihar	15	0	1
4	Chandigarh	8	0	0
5	Chhattisgarh	7	0	0
6	Delhi	87	6	2
7	Goa	5	0	0
8	Gujarat	69	1	6
9	Haryana	36	18	0
10	Himachal Pradesh	3	0	1
11	Jammu and Kashmir	48	2	2
12	Karnataka	83	5	3
13	Kerala	202	19	1
14	Ladakh	13	3	0
15	Madhya Pradesh	47	0	3
16	Maharashtra	198	25	8
17	Manipur	1	0	0
18	Mizoram	1	0	0
19	Odisha	3	0	0
20	Puducherry	1	0	0
21	Punjab	38	1	1
22	Rajasthan	59	3	0
23	Tamil Nadu	67	4	1
24	Telengana	71	1	1
25	Uttarakhand	7	2	0
26	Uttar Pradesh	82	11	0
27	West Bengal	22	0	2
	Total	1205	102	32

(This is the latest available data at the time of writing this article)

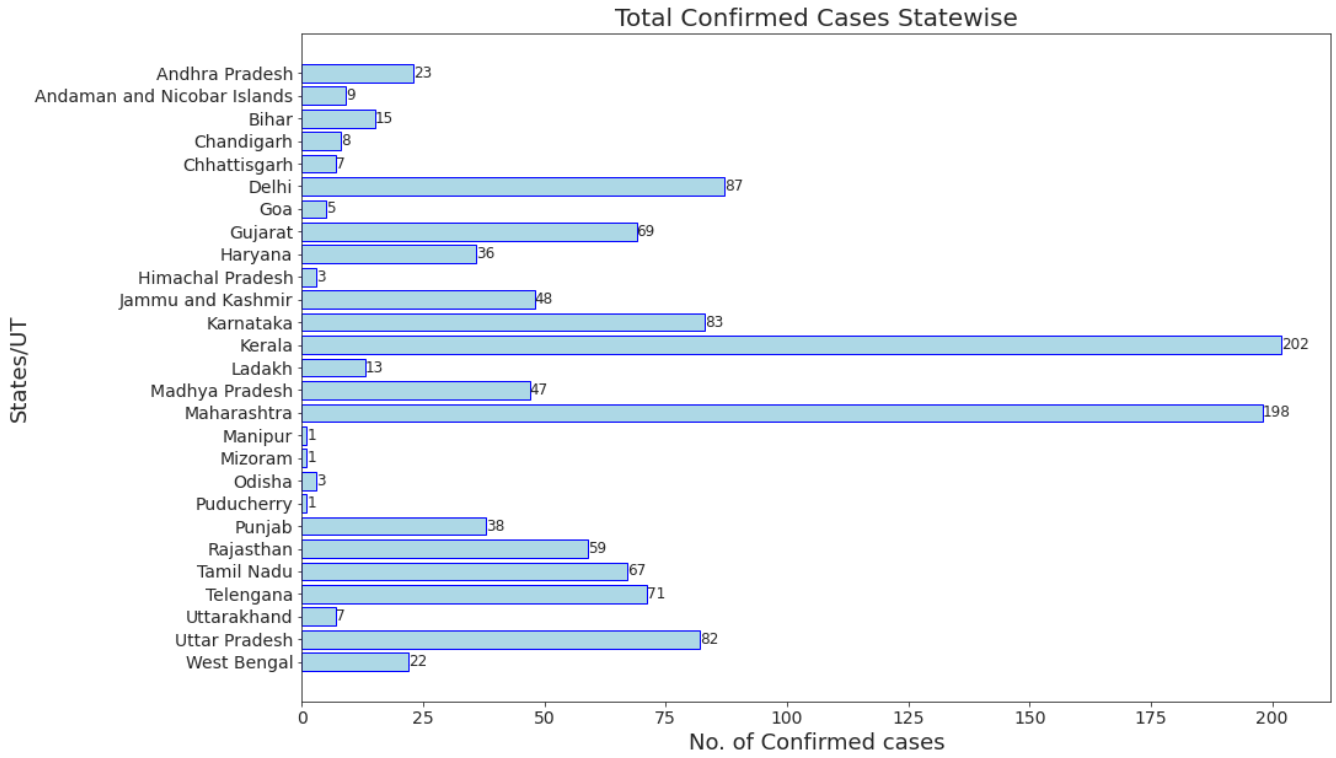
Bar Chart —Statewise total confirmed cases

Plotting horizontal barplot to show the statewise total confirmed cases —

```
sns.set_style("ticks")
plt.figure(figsize = (15,10))

plt.barh(state_data["States/UT"],
state_data["Confirmed"].map(int),align = 'center', color =
'lightblue', edgecolor = 'blue')

plt.xlabel('No. of Confirmed cases', fontsize = 18)
```

```
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.title('Total Confirmed Cases Statewise', fontsize = 18 )

for index, value in enumerate(state_data["Confirmed"]):
    plt.text(value, index, str(value), fontsize = 12)

plt.show()
```

Output:

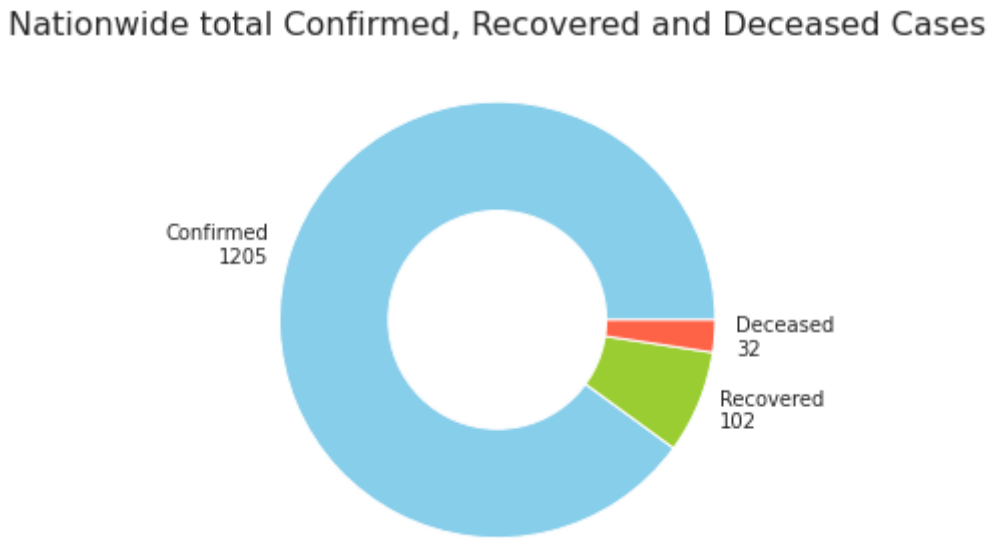


Donut Chart — Nationwide total Confirmed, Recovered and Deceased cases

```
group_size = [sum(state_data['Confirmed']),
              sum(state_data['Recovered']),
              sum(state_data['Deceased'])]

group_labels = ['Confirmed\n' +
str(sum(state_data['Confirmed'])),
                'Recovered\n' +
str(sum(state_data['Recovered'])),
                'Deceased\n' + str(sum(state_data['Deceased']))]

custom_colors = ['skyblue','yellowgreen','tomato']
```

```
custom_colors)
central_circle = plt.Circle((0,0), 0.5, color = 'white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size = 12)
plt.title('Nationwide total Confirmed, Recovered and Deceased
Cases', fontsize = 20)

plt.show()
```

Output:



Chloropleth map of the total Confirmed Cases

The shape files used in this article to plot the India map with state boundaries can be downloaded from here.

```
# reading the shape file of map of India in GeoDataFrame

map_data = gpd.read_file('Indian_States.shp')
map_data.rename(columns = {'st_nm':'States/UT'}, inplace = True)
map_data.head()
```

Output:

	States/UT	geometry
0	Andaman & Nicobar Island	MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2...

2	Assam	MULTIPOLYGON (((89.74323 26.30362, 89.74290 26...
3	Bihar	MULTIPOLYGON (((84.50720 24.26323, 84.50355 24...
4	Chandigarh	POLYGON ((76.84147 30.75996, 76.83599 30.73623...

I noticed that the names of some of the States and Union Territories(UT) and in the shape file were not consistent with the state names on the government website. So, I modified the State/UT names in the GeoDataFrame to match with the ones on our State dataframe.

Correcting spellings of states in *map_data* dataframe —

```
map_data['States/UT'] =
map_data['States/UT'].str.replace('&','and')
map_data['States/UT'].replace('Arunanchal Pradesh',
                              'Arunachal Pradesh', inplace =
True)
map_data['States/UT'].replace('Telangana',
                              'Telengana', inplace = True)
map_data['States/UT'].replace('NCT of Delhi',
                              'Delhi', inplace = True)
map_data['States/UT'].replace('Andaman and Nicobar Island',
                              'Andaman and Nicobar Islands',
                               inplace = True)
```

Merge the two dataframes *state_data* and *map_data* on States/UT names —

```
merged_data = pd.merge(map_data, state_data,
                       how = 'left', on = 'States/UT')

merged_data.fillna(0, inplace = True)
merged_data.drop('Sr.No', axis = 1, inplace = True)
merged_data.head()
```

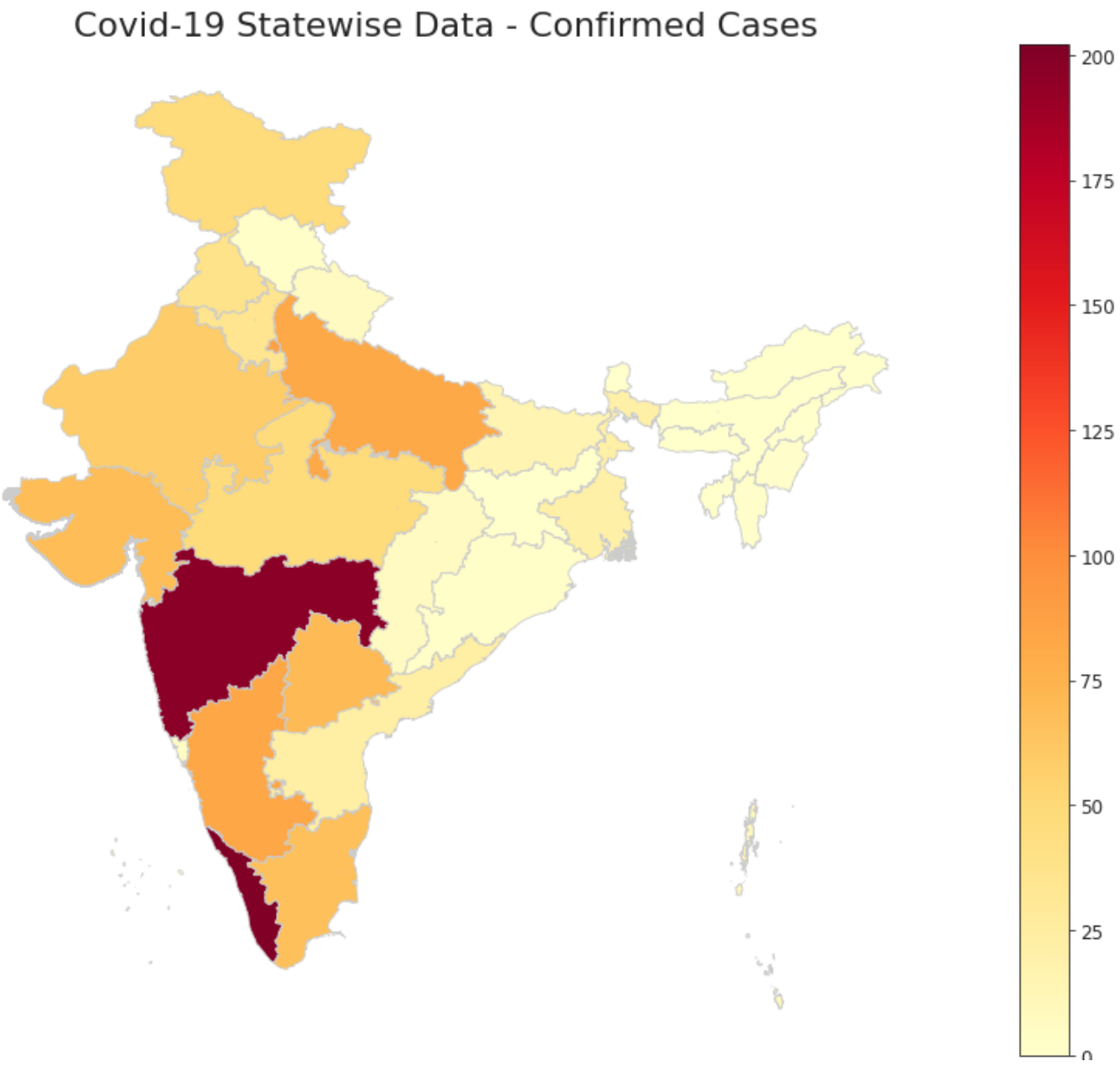
Output:

	States/UT	geometry	Confirmed	Recovered	Deceased
0	Andaman and Nicobar Islands	MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2...	9.0	0.0	0.0
1	Arunachal Pradesh	POLYGON ((96.16261 29.38078, 96.16860 29.37432...	0.0	0.0	0.0
2	Assam	MULTIPOLYGON (((89.74323 26.30362, 89.74290 26...	0.0	0.0	0.0

Display the Statewise data on the map of India —

```
fig, ax = plt.subplots(1, figsize=(20, 12))
ax.axis('off')

ax.set_title('Covid-19 Statewise Data – Confirmed Cases',
             fontdict =  {'fontsize': '25', 'fontweight' : '3'})

merged_data.plot(column = 'Confirmed', cmap='YlOrRd',
                 linewidth=0.8, ax=ax, edgecolor='0.8',
                 legend = True)
plt.show()
```

Output:



Complete Python code

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import requests
from bs4 import BeautifulSoup
import geopandas as gpd
from prettytable import PrettyTable

# offical ministry of health website
url = 'https://www.mohfw.gov.in/'

# make a GET request to fetch the raw HTML content
web_content = requests.get(url).content

# parse the html content
soup = BeautifulSoup(web_content, "html.parser")

# remove any newlines and extra spaces from left and right
extract_contents = lambda row: [x.text.replace('\n', '') for x in row]

stats = [] # initialize stats
all_rows = soup.find_all('tr') # find all table rows

for row in all_rows:
    stat = extract_contents(row.find_all('td')) # find all data cells
    # notice that the data that we require is now a list of length 5
    if len(stat) == 5:
        stats.append(stat)

# now convert the data into a pandas dataframe for further processing
new_cols = ["Sr.No", "States/UT","Confirmed","Recovered","Deceased"]
state_data = pd.DataFrame(data = stats, columns = new_cols)

# converting the 'string' data to 'int'
state_data['Confirmed'] = state_data['Confirmed'].map(int)
state_data['Recovered'] = state_data['Recovered'].map(int)
state_data['Deceased']  = state_data['Deceased'].map(int)

# pretty table representation
table = PrettyTable()
table.field_names = (new_cols)
for i in stats:
    table.add_row(i)
```

```
                sum(state_data['Recovered']),
                sum(state_data['Deceased'])])
print(table)

# barplot to show total confirmed cases Statewise
sns.set_style("ticks")
plt.figure(figsize = (15,10))
plt.barh(state_data["States/UT"], state_data["Confirmed"].map(int),
         align = 'center', color = 'lightblue', edgecolor = 'blue')
plt.xlabel('No. of Confirmed cases', fontsize = 18)
plt.ylabel('States/UT', fontsize = 18)
plt.gca().invert_yaxis() # this is to maintain the order in which the states appear
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.title('Total Confirmed Cases Statewise', fontsize = 20)

for index, value in enumerate(state_data["Confirmed"]):
    plt.text(value, index, str(value), fontsize = 12, verticalalignment = 'center')
plt.show()

# donut chart representing nationwide total confirmed, cured and deceased cases
group_size = [sum(state_data['Confirmed']),
              sum(state_data['Recovered']),
              sum(state_data['Deceased'])]

group_labels = ['Confirmed\n' + str(sum(state_data['Confirmed'])),
                'Recovered\n' + str(sum(state_data['Recovered'])),
                'Deceased\n'  + str(sum(state_data['Deceased']))]
custom_colors = ['skyblue','yellowgreen','tomato']

plt.figure(figsize = (5,5))
plt.pie(group_size, labels = group_labels, colors = custom_colors)
central_circle = plt.Circle((0,0), 0.5, color = 'white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size = 12)
plt.title('Nationwide total Confirmed, Recovered and Deceased Cases', fontsize = 16)
plt.show()

# read the state wise shapefile of India in a GeoDataFrame and preview it
map_data = gpd.read_file('Indian_States.shp')
map_data.rename(columns = {'st_nm':'States/UT'}, inplace = True)
map_data.head()

# correct the name of states in the map dataframe
```

```
map_data['States/UT'].replace('Telangana', 'Telengana', inplace = True)
map_data['States/UT'].replace('NCT of Delhi', 'Delhi', inplace = True)

# merge both the dataframes - state_data and map_data
merged_data = pd.merge(map_data, state_data, how = 'left', on = 'States/UT')
merged_data.fillna(0, inplace = True)
merged_data.drop('Sr.No', axis = 1, inplace = True)
merged_data.head()

# create figure and axes for Matplotlib and set the title
fig, ax = plt.subplots(1, figsize=(20, 12))
ax.axis('off')
ax.set_title('Covid-19 Statewise Data - Confirmed Cases', fontdict = {'fontsize': '25',
# plot the figure
merged_data.plot(column = 'Confirmed', cmap='YlOrRd', linewidth=0.8, ax=ax, edgecolor='0
plt.show()
```

End notes

Covid-19 hasn't yet hit India in a widespread way. The missing pieces of data are making it impossible to predict how the outbreak will unfold in the coming months. As I conclude this article, I pray for the safety and well-being of everyone in India and around the world.

. . .

Note from the editors: *Towards Data Science is a Medium publication primarily based on the study of data science and machine learning. We are not health professionals or epidemiologists, and the opinions of this article should not be interpreted as professional advice. To learn more about the coronavirus pandemic, you can click here.*

Covid 19 Data Visualization Web Scraping India Python

> ## Stay up to date on coronavirus (Covid-19)
>
> Follow the Medium Coronavirus Blog or sign up for the newsletter to read expert-backed coronavirus stories from Medium and across the web, such as:
>
> - Social distancing has made all of us helpers.

- Could blood thinners be helpful for Covid-19?