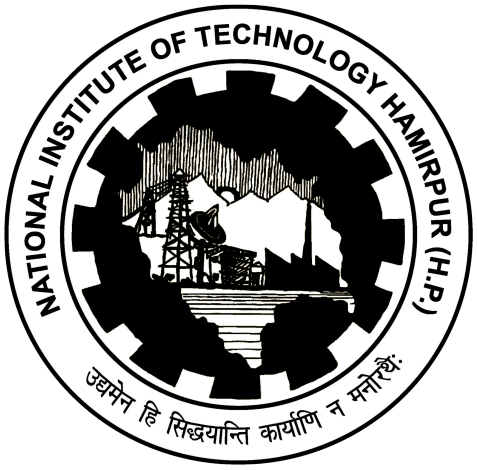

National Institute of Technology, Hamirpur

Department of Computer Science and Engineering

Covid-19 Tracker

10 July, 2020

Author:

Kashish Srivastav (185014)
Dipesh Chopra (185015)
Akash Rana (185034)

Supervisor:

Dr. Chandra Shekhar

The Abstract

> "A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance."

Ever since the first cases of COVID-19 appeared, it was unclear exactly how many people were exposed to the virus. Fortunately, India has dramatically increased our testing capacity since then. But we're still playing catch-up. And in the face of conflicting reports of the number of confirmed cases, our team built a dashboard to show the most accurate data we could find, in the cleanest way possible. When our team was deciding which stack to use, we knew that our primary goal of cleaning and displaying healthcare data would best be accomplished if we wrote our application in Python. Apart from using Python we used CSS to style our dashboard.

Installation

The dashboard has been deployed using Heroku platform as a service (PaaS) and can be found at https://covid-19-co.herokuapp.com/

If you want to deploy the app on your local machine, follow the steps below.

- The source code can be found at

```
git clone https://github.com/cannibalcheeseburger/
    covid-19-tracker.git
cd covid-19-tracker
```

- Requirements

```
python -m pip install -r requirements.txt
```

- Running Program

```
python app.py
```

- Open a browser and connect to the host server. Usually, the server is hosted at port: 8050. Enter the following address in the address bar. http://127.0.0.1:8050/

Usage

The dashboard looks like the image below.

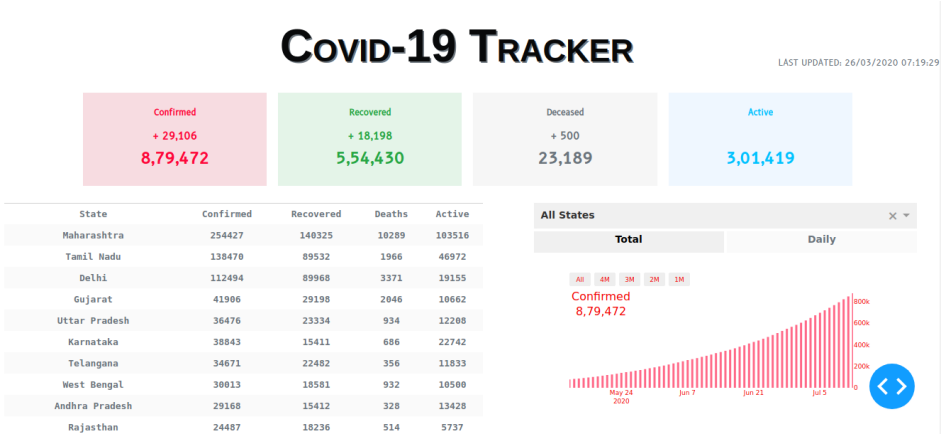

Figure 1: Dashboard.

At the top of the dashboard we can find the total number of Confirmed, Recovered, Deceased and Active cases. Daily increase in the number of cases is also displayed.

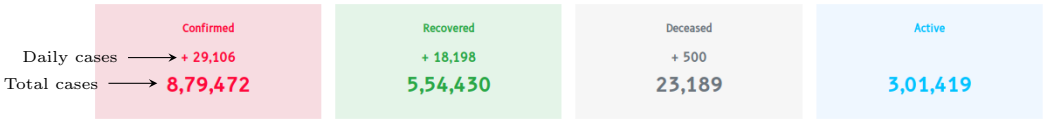

Figure 2: Top bar

The table to the left summarize the total number of cases state-wise. We can also view the individual time-series graphs of confirmed, recovered, deceased and active cases by selecting states from the given dropdown. It also has two different tabs to show the graphs on Total/Daily basis.

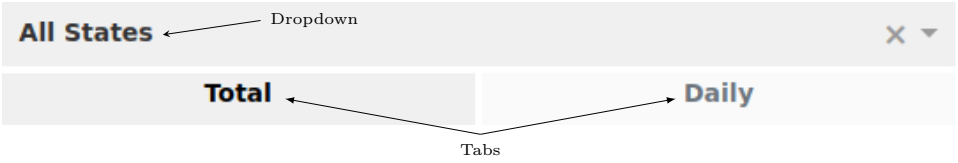

Figure 3: Dropdown and Tabs

Using the graphs is very easy and simplified. Initially, date axis begin from the start of previous month, which can be toggled by using the buttons available with each graph.

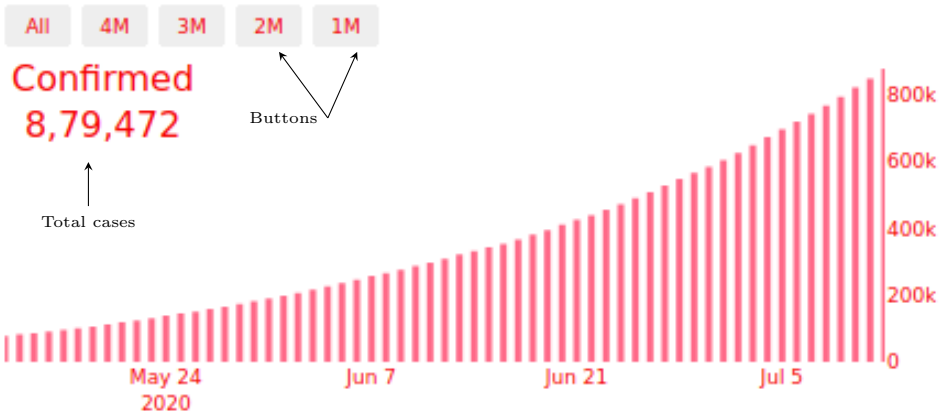

Figure 4: Graph for Total Confirmed Cases

Figure 5, shows the spread trends of the virus in the country.

- Confirmed per Million
- Recovery Rate
- Active Percentage
- Mortality Rate

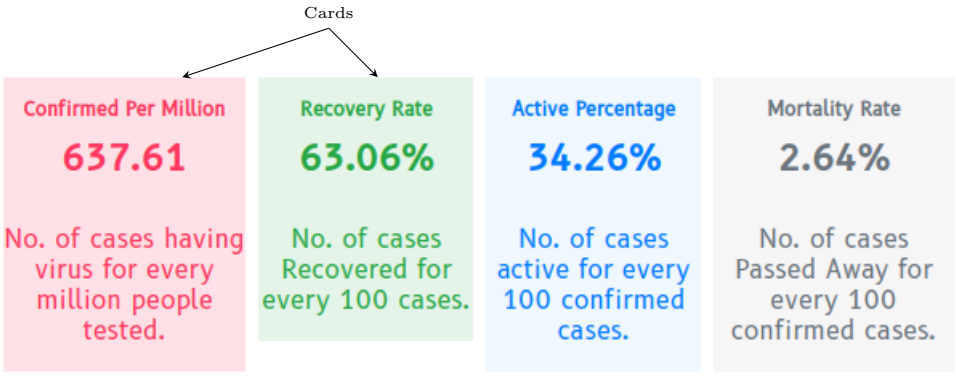

Figure 5: Spread Trends

The scatter plot below the trends corner and state-wise data table shows the relation between Deaths and Confirmed cases per state.

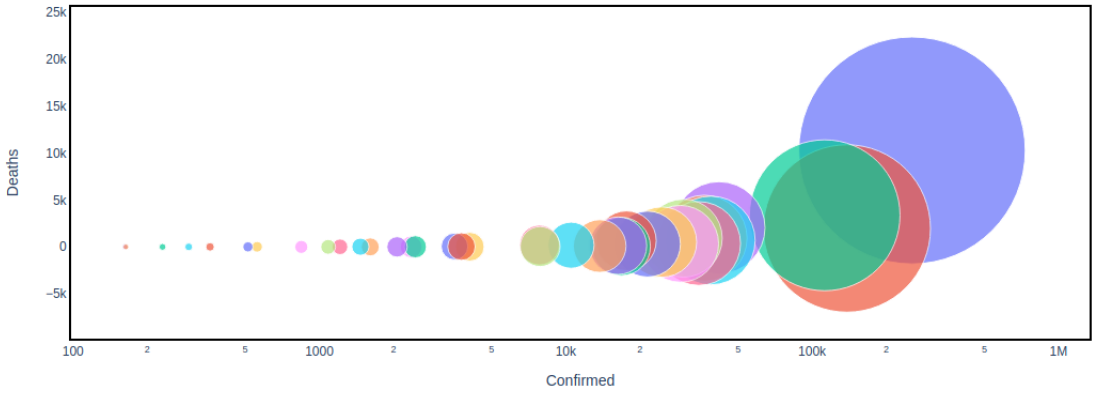

Figure 6: Relation between deaths and confirmed cases.

Results

Great flexibility comes with convoluted layers and a complex syntax. It can often be daunting with tedious coding to make a publishable plot using matplotlib. In other words, matplotlib should not be our only data visualisation tool, we need always be equipped with alternatives in our toolbox to handle evolving requirements. When it comes with requirements for efficiency, interaction, and web-based visualisation, 'plotly' and 'dash' are must-learn tools. The beauty about these tools is that we do not need to master JavaScript, CSS, and HTML for making interactive web visualisation as both tools handle that for us. They connect us with JavaScript (e.g. React.js, Plotly.js, and d3.js) to make graphs responsive and also beautiful. What we need to provide is just simple codes in pure python environment.

Since it is a web-based application, it is inevitable to deal with HTML structures for the layout. Instead of writing HTML ourselves, dash provides us with the dash-html-components library to deal with it. We basically just need to compose the layout using Python structures.

Conclusion

We knew that Plotly Dash and some of its libraries (Dash Core, HTML, and Bootstrap Components) could offer the tools we need and save us a ton of development time. And last but not least, Python3 and dash led to flexibility and a scalable backend for our hosted and cached data.

References

All of the API used to for retrieving datasets can pe found at :
https://api.covid19india.org/csv/

- https://api.covid19india.org/csv/latest/state_wise.csv
- https://api.covid19india.org/csv/latest/state_wise_daily.csv
- https://api.covid19india.org/csv/latest/state_wise_daily.csv
- https://api.covid19india.org/csv/latest/case_time_series.csv