CSE506 Intro. to Data Mining

Assignment 1

Arjun Lakhera 2018133
Daksh Thapar 2018137

We have made a function compDates(s,sd,ed) in our file that compares string date from the JSON column and compares it with starting and ending dates, and returns True if it exists between them and False otherwise.

We have also made a function covidDF(json_file_path) that takes in path of the JSON and normalizes it to create a dataframe that we can use for all our questions.

Q1. Data Manipulation

1. We made 3 counter variables, 1 for each category and iterated over the dataframe row- wise and checked whether date fir that row lies between starting date and end date and checked the status column for that row and according to this value, we added the value of the attribute “tt” into our counter values. We did calculate sum over the entire row initially but after careful observations, we saw in the JSON that “tt” contains the sum for all the states and we can just access value at that index to find cases for that day for that category.

   confirmed_count: 4110211 recovered_count: 3177666 deceased_count: 70094

2. Performed the exact same steps as in Part 1, now we just add those values to the counter for every row that correspond to the column name “dl” (Delhi).

   confirmed_count: 188193 recovered_count: 163785 deceased_count: 4538

3. Performed the exact same steps as in Part, now we just add those values to the counter for every row that correspond to the column name “dl” (Delhi) and “mh” (Maharashtra).

   confirmed_count: 1072055 recovered_count: 800359 deceased_count: 30813

4. For each of the 3 categories, I have created a separate dictionary, and its keys being the state names or column names. We converted our dataframe to float values, and this means now our string numbers will be float numbers and our characters strings will be “nan” valued. We iterate over every row and check category, and then every column for that row and check its column name and for that specific name (or key in the dictionary) we add that float value. We repeat this for every category and every row.

   After we have iterated over the dataframe and added details to our dictionary, we remove the keys “status” “date” and “un” from our dictionaries. Status and Date are strings that have nan values after float conversion and “un” is a column for unassigned states so we felt it is not relevant to our question, and so has been clarified by the TAs in the announcements.

After we obtain these 3 dictionaries, we sort them based on their cases values in decreasing order and print the states and their respective counts for the maximum case value (if multiple states have the same maximum value, we have taken care of that scenario too).

```
Confirmed
Highest affected State is:  mh
Highest affected State count is:  883862
Recovered
Highest affected State is:  mh
Highest affected State count is:  636574
Deceased
Highest affected State is:  mh
Highest affected State count is:  26275
```

5. Performed the exact same steps as in Part 4, we now sort our final dictionaries in increasing order and print all those states having their cases values equal to the minimum value.

```
Confirmed
Lowest affected State is:  dd ld
Lowest affected State count is:  0
Recovered
Lowest affected State is:  dd ld
Lowest affected State count is:  0
Deceased
Lowest affected State is:  dd ld mz
Lowest affected State count is:  0
```

Note- *On Google Classroom people have been clarifying whether to keep or remove union territory values, but according to the state map list provided in that announcement-*

https://github.com/covid19india/blog/blob/7cf01da3d5de7b860ae3bfc0ec3638f2e72863a6/_posts/2020-06-15-hornbill.md

this clearly shows the heading for all rows as "States" and this website classifies all the columns as states even if they are union territories. Also in the questions, "state Delhi" has been mentioned even though it's a union territory, so for this dataset and problem and the given state map list, we have assumed all these entities as states

6. We iterated over every row keeping in mind the date is valid and also the category (status), and checked if state is Delhi and compared the cases values with a predefined max value= 0, we keep on updating spike value and the corresponding date when the number of cases for that day for that category are greater than the max value variable.

```
Confirmed
Day:  23-Jun-20
Count:  3947
Recovered
Day:  20-Jun-20
```

```
Count:  7725
Deceased
Day:  16-Jun-20
Count:  437
```

7. Performed the exact same steps as in Part 4, and now that we have our 3 dictionaries, for every state in those 3 dictionaries, we find Active cases values (Confirmed-Recovered-Deceased) and print values for all states separately.

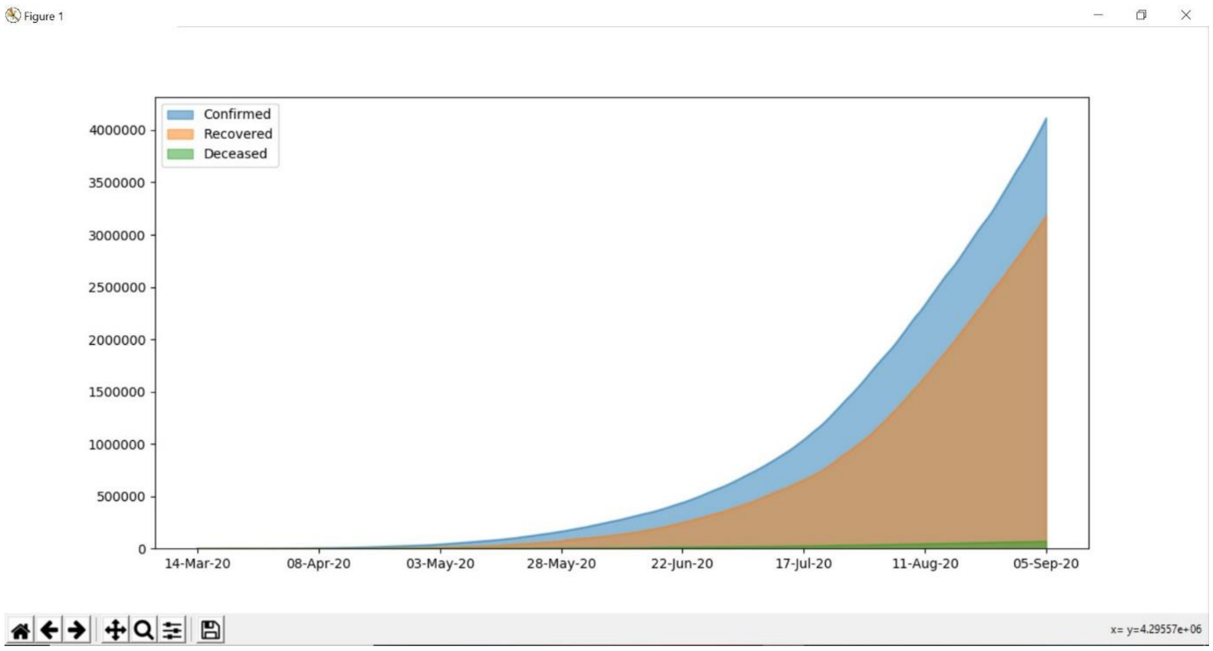
```
Active Cases on 5th September 2020, state wise-
an -  343
ap -  100880
ar -  1525
as -  28404
br -  16735
ch -  2143
ct -  22320
dd -  0
dl -  19870
dn -  301
ga -  4945
gj -  16266
hp -  2023
hr -  14912
jh -  14980
jk -  9547
ka -  100224
kl -  21867
la -  834
ld -  0
mh -  221013
ml -  1374
mn -  1872
mp -  15687
mz -  349
nl -  701
or -  25856
pb -  15870
py -  5163
rj -  14996
sk -  561
tg -  32405
tn -  51580
tr -  5905
up -  59963
ut -  7649
wb -  23390
```

Q2. Plotting

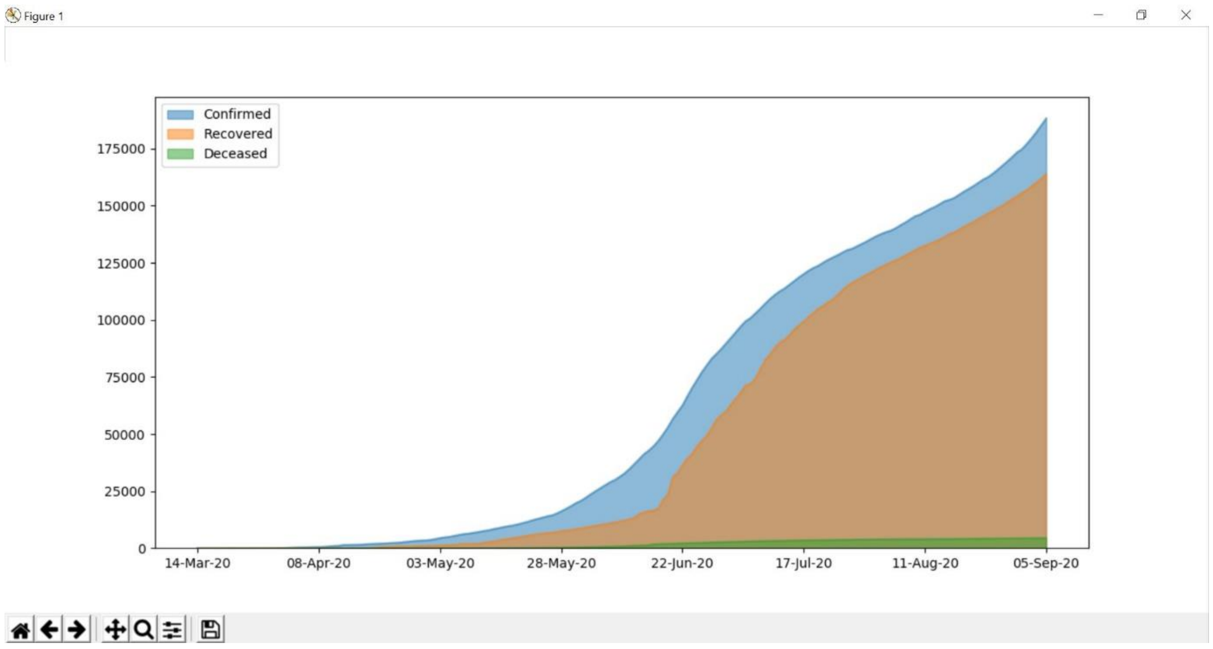
We have plotted cumulative plots as has been suggested by Tas on classroom and mail, and for better analysis we have provided both kind of plots, **cumulative** and **per- day plots**

Cumulative

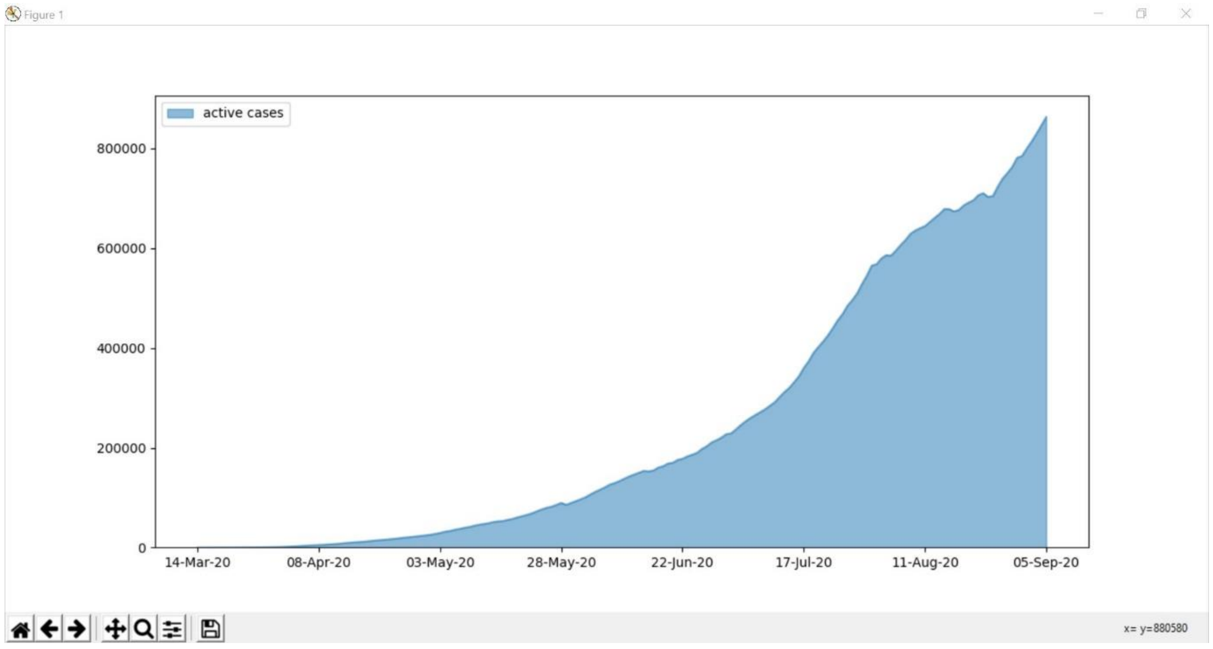
1. Area trend line for total number of "Confirmed", "Recovered" and "Deceased" from start date to end date



2. Area trend line for total number of "Confirmed", "Recovered" and "Deceased" from start date to end date for state Delhi

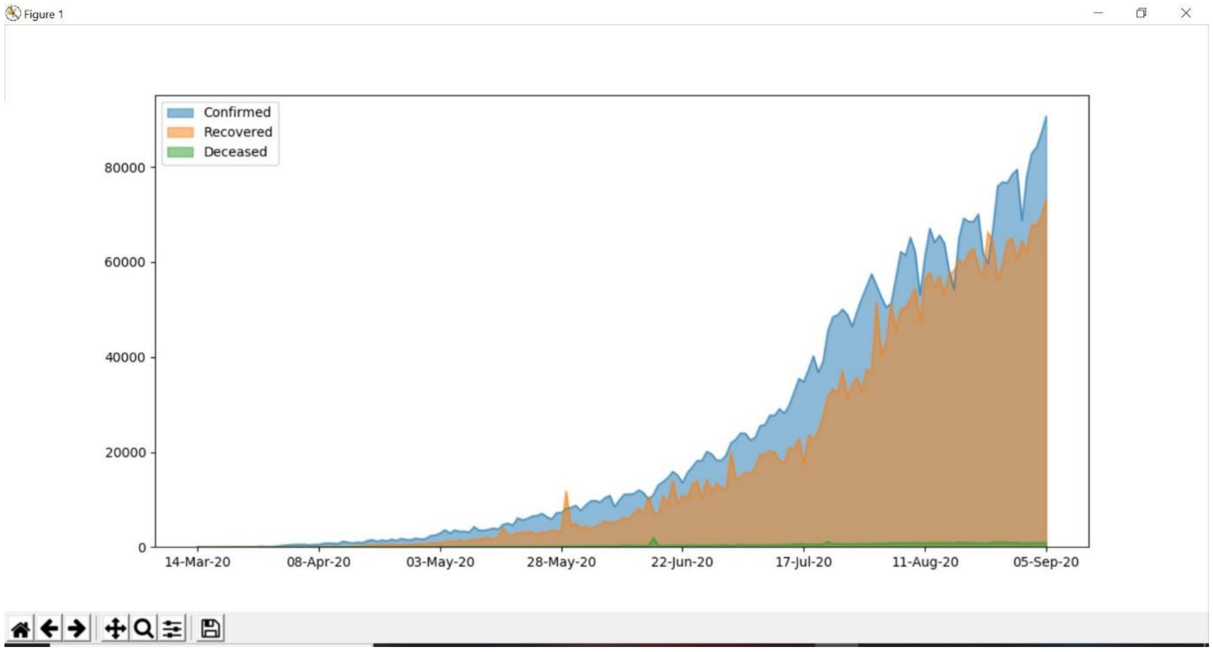


3. Area trend line for active cases of “Confirmed”, “Recovered” and “Deceased” from start date to end date

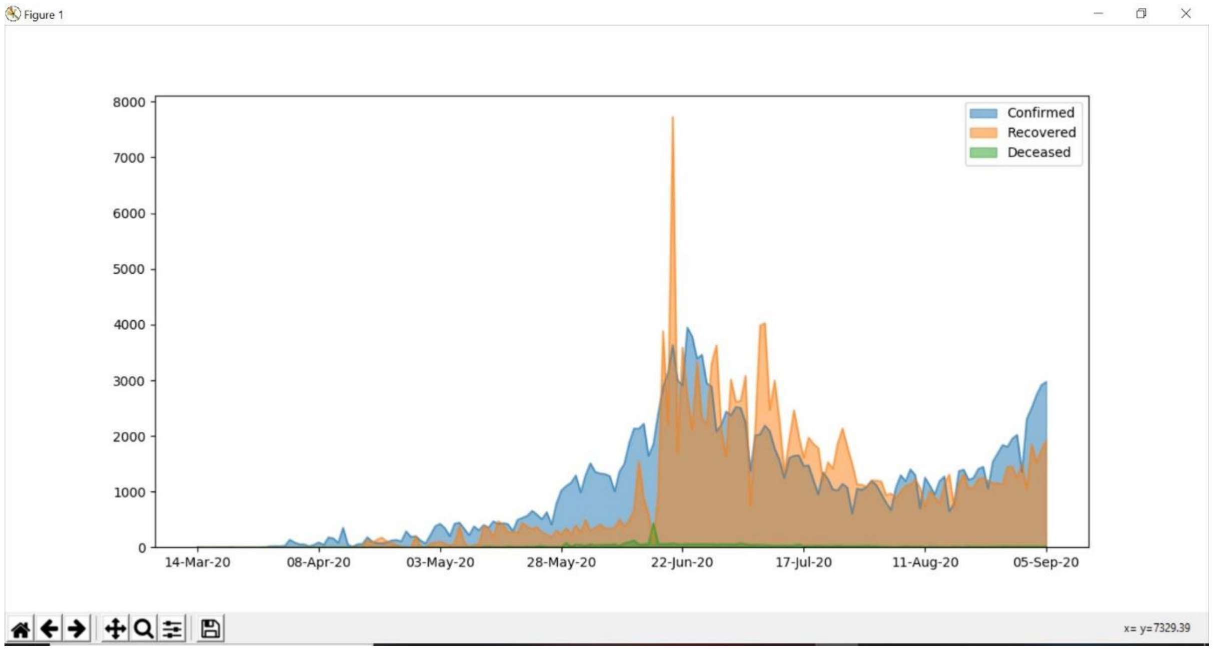


Per Day Plots

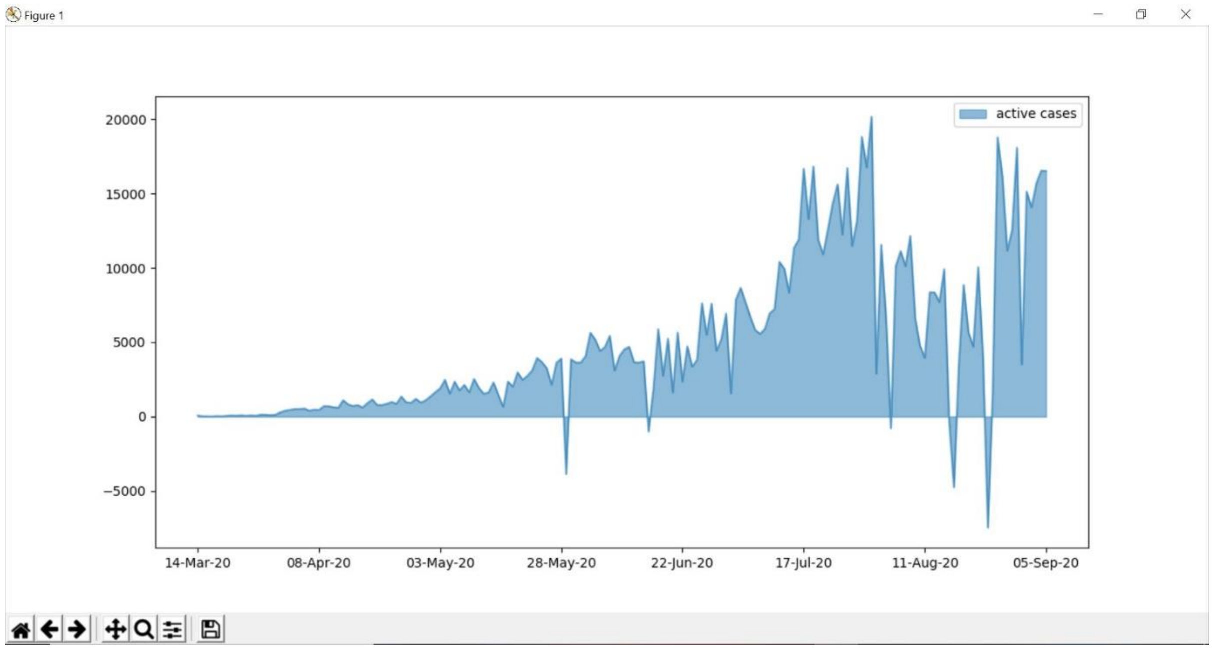
1. Area trend line for total number of “Confirmed”, “Recovered” and “Deceased” from start date to end date



2. Area trend line for total number of “Confirmed”, “Recovered” and “Deceased” from start date to end date for state Delhi



3. Area trend line for active cases of “Confirmed”, “Recovered” and “Deceased” from start date to end date



df.plot(kind='area', stacked=False, figsize=(18, 8))

We have used this piece of code for area trend line and it helps us compare the relationships between the 3 status values side by side and fills the area under the lines

We have provided Per- day plots along with cumulative plots (the ones asked for), as we felt that the per day plots give us a very different insight into the figures and statistics, and are able to give us a better idea of how the values change from day to day, and the rate of increase or decrease of cases is much more visible in per day plots. But also, cumulative plots offer us a different insight into this and help us analyse the bigger picture for our issue and is useful as well.

Q3. Linear Regression

Linear regression on the state Delhi data over dates, separately for "Confirmed", "Recovered" or "Deceased" and report intercept and slope coefficients for all 3 cases from start date to end date.

For linear regression, we have used the normal form equation to get our parameters directly from the dataset given, but before that we have to pre- process the dataset and add a column of 1s necessary for our normal form equation and it gives us our coefficient values.

Confirmed intercept: -11.684415584416456 Confirmed slope: 12.214269205370904
Recovered intercept: -158.44266233766257 Recovered slope: 12.30552828527405
Deceased intercept: 8.948441558441544 Deceased slope: 0.19023332599603793

These are the values returned by our normal form equation for the status values.

Assumption- We have calculated coefficients of linear regression for cases "per day" and not a cumulative distribution.
