



MFDS 2020 TERM PROJECT

GROUP 30

-SOURABH THAKUR (CH17B022)
-AVINASH G BAGALI (AE17B110)
-REX TOPLAN (EP17B022)
-JESWANT KRISHNA (AE17B030)
-ASWATH HARI (CH17B012)

Question 1: Image Compression with SVD

Approach to the Solution:

We first create the average image for a particular subject by taking the arithmetic mean of the grayscale values of the corresponding pixels of the 10 images for a given image. Once we have the average image of the subject, we use singular value decomposition to compress the average image. We get a result of 149/150 correct identifications using this approach.

Overview of the code:

We first create a full image database storing each image as a $x*1$ vector ($x=4096$ is the no. of pixels), for $y=10$ images per subject and for a total of $z=15$ subjects. We get a $x*y*z$ size matrix, we then find the avg image for each of the subjects and perform SVD on this average image. We then store this representative image as a vector giving us a $(x*z) = (4096*15)$ array with a representative image for each of the subjects. Finally we study the performance by assessing the number of images that can be identified correctly based on the representative image alone.

Working of the code:

Imp variables:

x→no of pixels in an image **y**→no of images per subject **z**→no of subjects

full_img→the uncompressed data matrix ; **count**→no of correct identifications of subjects

rep_img→compressed data matrix with 1 rep img per subject ; **comp**→no of SVD terms used for rep img

Our code consists of three main functions with the following functions:

1. **img_comp_main**→it is the main function where we can give all inputs like img address, no of subjects, no. of images per subject etc. it also assesses the performance by finding the number of images identified correctly
2. **pull_data**→ this function takes the address of the file consisting of the documents along with other inputs and it creates a data matrix (img_data) with all the input info
3. **create_rep_img**→this takes the img_data matrix and returns a smaller matrix with only one representative image per subject

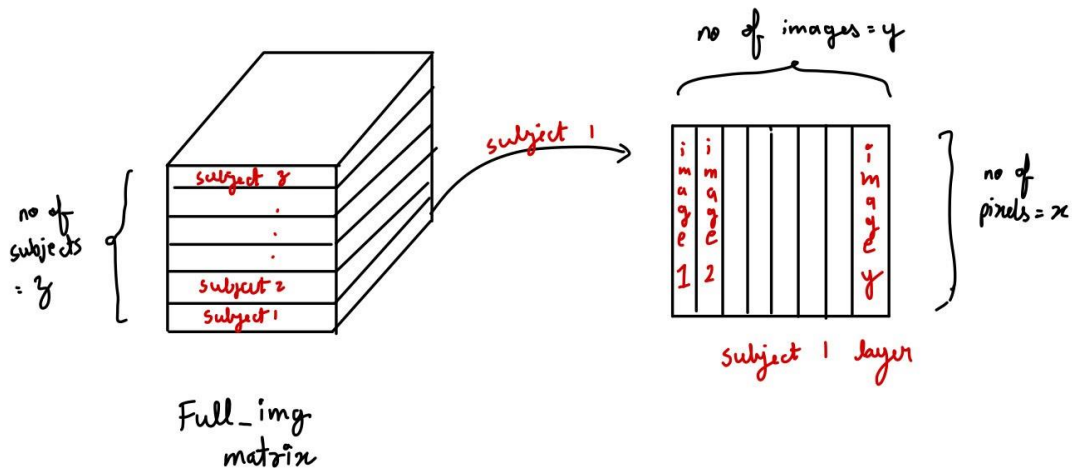
1. Pull_data:

```
function full_img = pull_data(main_address,x,y,z)
```

as we can see the pull_data function takes the main_address where we have stored the data along with **x**→no of pixels, **y**→no of images per subject and **z**→ no of subjects and it returns a data matrix of size $x*y*z$...called full_img this can be considered as the uncompressed full_size data of the images.

We visit the address passed on and load the data from each .pgm file into the full_img matrix by converting the $64*64$ image into a $4096*1$ column and appending it to the full_img matrix..we

can visualize the full_img matrix as follows, each layer corresponds to a subject, in each layer, every column corresponds to a single image taken in a particular lighting condition.



The create_rep_img takes this full_img matrix and compresses it into a $x \times z$ image i.e. it stores only one representative image per subject.

2. Create_rep_img:

```
function rep_img = create_rep_img(full_img, x, y, z, comp)
```

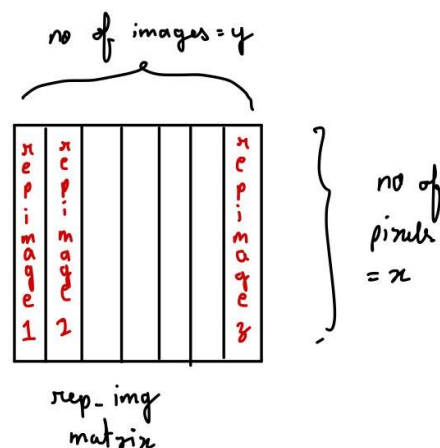
The create_rep_img takes an extra argument-comp → it is the number of the singular value decomposition components we take into creating the rep_img, the higher comp is better the chance of correctly identifying the subject, but also more data to be stored, we explain later how we chose the value of comp for our case.

To create the representative image for a subject, we begin by obtaining the average image by finding the arithmetic mean of all the columns of a layer of the full_img matrix. This average image looks like a blend of all the 10 images of the subject.

Note: we divide by the number of images before adding in order to avoid overflow, as the grayscale value can't exceed 255

Once we have the average image (4096×1) we reshape it into its original form, in this case (64×64) and perform SVD on the average image. Now depending on the value of comp, we take the corresponding number of SVD components (=comp) and create the representative image as follows

We reshape and append this representative image into the rep_img matrix, we iterate through all layers (i.e. all the subjects) of the full_img matrix and create the representative images for all the subjects and obtain the final rep_img matrix (It is a $x \times z$ matrix).



3. Img_comp_main:

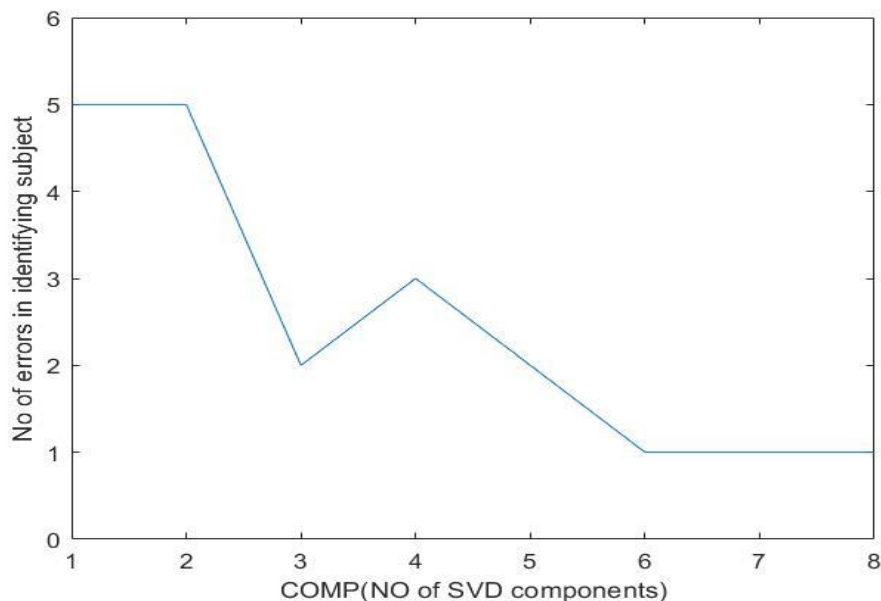
The main function calls the Pull_data function first and creates the full_img matrix and then calls the create_rep_img to give the rep_img matrix, we now need to see how well do the representative images perform in identifying the correst subject.

We initialtilze the variable count= 0 and for every subject identified correctly, we increment count by 1. We now iterate through the full_img matrix and find the norm of the difference between the test image and the representative image of each subject..we return the subject for which the norm of this difference is the least.

We notice that the count value depends on the value of comp (no of SVD components we include in the representative image)
















Deciding the parameter 'Comp':
















When we change the comp we change the representative image and it becomes closer to the average image(note that we have performed SVD on the average image) when comp is increased, but increasing comp also might increase the data that is being stored and the computations to be performed. Therefore we plot the error in recognizing subjects vs the comp to get a better idea of which value of comp is to be chosen. We see that we get only 1 error if $\text{comp} \geq 6$, so we chose $\text{comp} = 6$















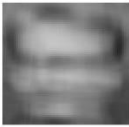


Below is the table showing our results of image compression using SVD, the first column is an example of the original quality of the image, the second is the average formed by taking the Arithmetic mean of the image and the third column consists of the SVD(upto 6 terms of the Average image)

NOTE: if it is not possible to load the images from the given address, the code can be run using the data included along with the code named full_img and rep_img.

Sr no	ORIGINAL IMAGE (one of the 10 images)	AVERAGE IMAGE (created by the avg of the 10 images)	COMPRESSED IMAGE (6 Term SVD of the avg image)
1			
2			
3			
4			
5			

6			
7			
8			
9			
10			

11	 	
12	 	
13	 	
14	 	
15	 	

Question 2: Logistic Regression

DATA STATISTICS

	Temperature	Pressure	Feed Flow rate	Coolant Flow rate	Inlet reactant concentration
Mean	546.76643	25.49327	125.02906	2295.79777	0.302692
Median	545.8	25.375	124.59	2268.71	0.30885
Std Deviation	86.85878045	14.25240732	43.50815917	763.6806253	0.116062229
Variance	7544.447741	203.1311143	1892.959914	583208.0974	0.013470441
Skewness	0.06925060	-0.0210707	0.0184997	0.037457525	-0.0259988

Report about the model -

- Continuous shuffling of the data set was done to make sure that the training data after making a 70-30 split is not unbiased to avoid assuming that the split gives unbiased training data
- The training data has almost a 58-42 split in the response variable
- Learning rate was set to 0.04 and weights were initialized at 0.
- The weights which came out as results in the final trained model are
[-0.16619278], [-0.56517619], [-0.79076728], [4.22817038], [-0.17620579]]
- The weights found for the same training split from the sklearn library are as follows-
- [[-0.14713029, -0.5363028 , -0.72215913, 3.69615027, -0.169207]]
- They deviate a bit from the actual model reason being the addition of regularization term in the model
- The confusion matrix for the data are as follows -
Confusion Matrix of Test Data
[[110 5][19 166]]
Confusion Matrix of Train Data
[[284 16] [30 370]]

- Other metrics reported are as follows -
- Accuracy on Test: 92.0
- Accuracy on Train: 93.42857142857143
- F1 Score of Test data: 0.9016393442622951
- F1 Score of Train data: 0.9250814332247557
- The model code takes everything as an input during instant creation.
- The model was trained iteratively with different initialisation of the weights the one with best accuracy has been reported

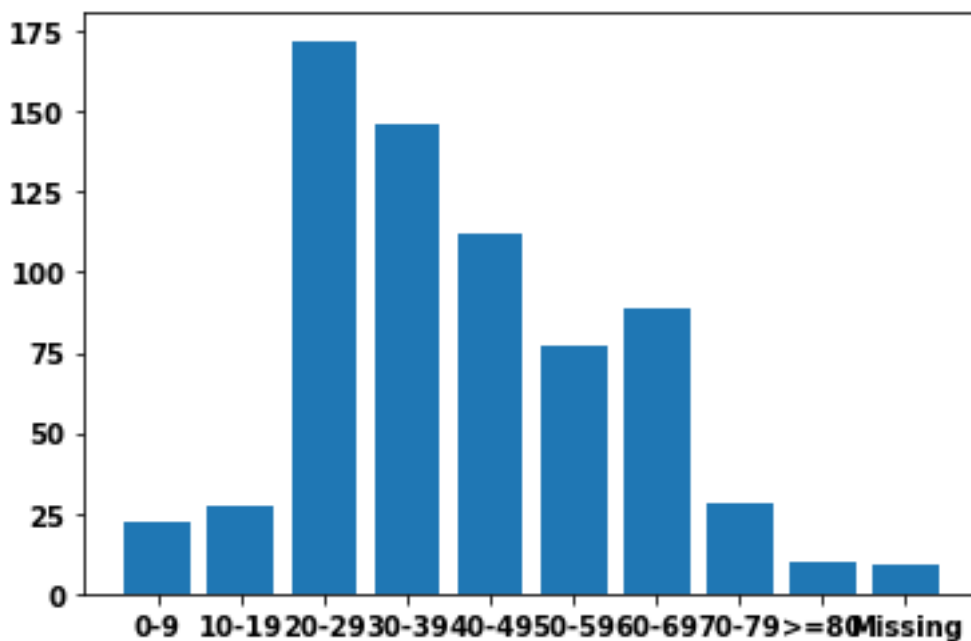
Question 3: Coronavirus Stats

NOTE: For Q 4,5,6,&8 New data sets from Kaggle are used

Q.1) Which age group is the most infected?

Code:

```
agegr = pd.read_csv('AgeGroupDetails.csv')
plt.bar(agegr['AgeGroup'],agegr['TotalCases'])
print(agegr[agegr['TotalCases']==agegr['TotalCases'].max()].AgeGroup)
```



Ans : Age Group 20 – 29

2. Plot graphs of the cases observed, recovered, deaths per day country-wise and state-wise.

Process involved:

Convert Covid 19 csv into a dataframe

Group the df country-wise and state-wise taking the count of cases, deaths and recovered

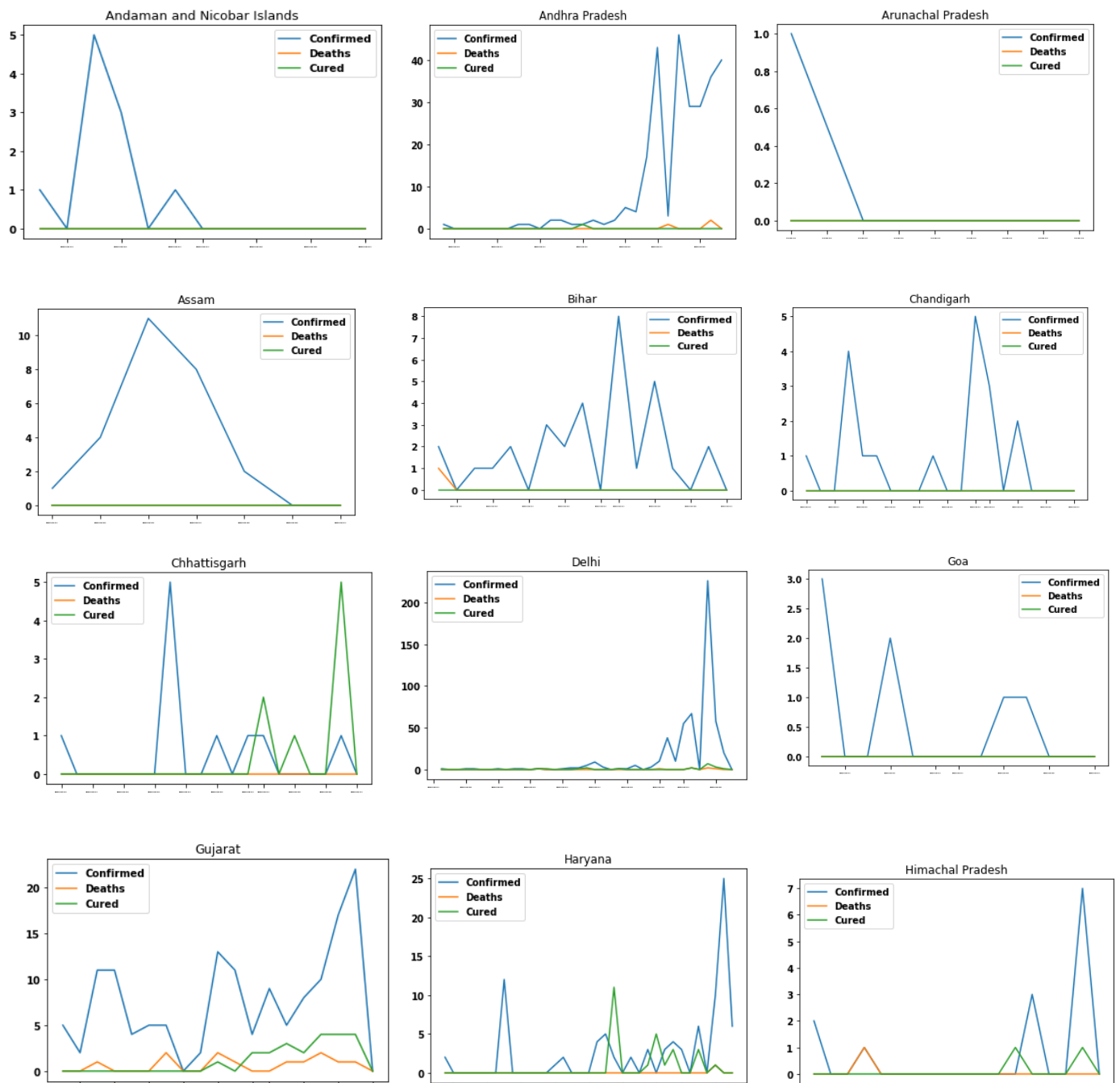
Create a column which has death, active and recovered cases per day and plot the graph

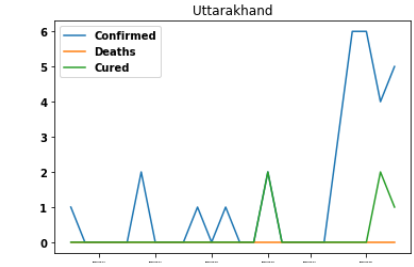
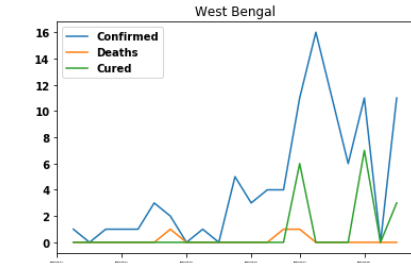
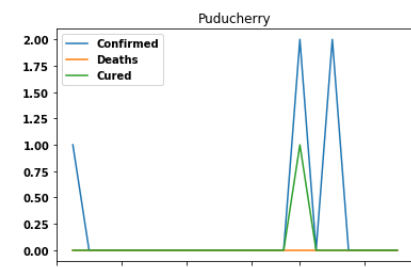
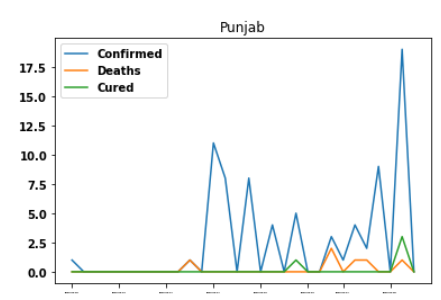
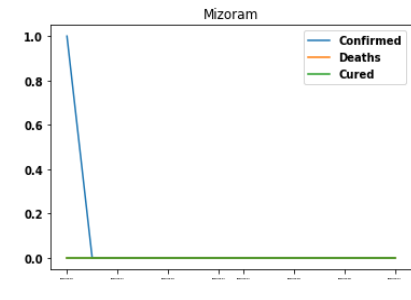
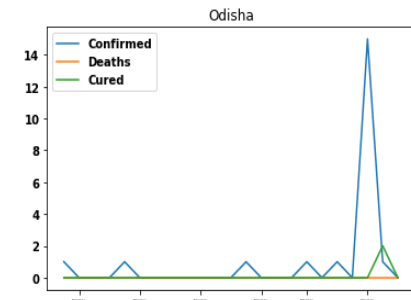
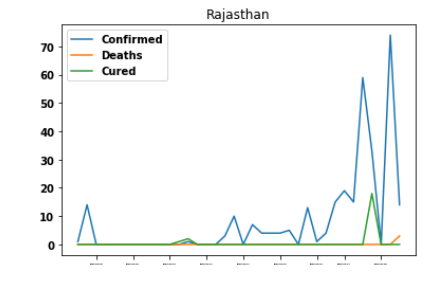
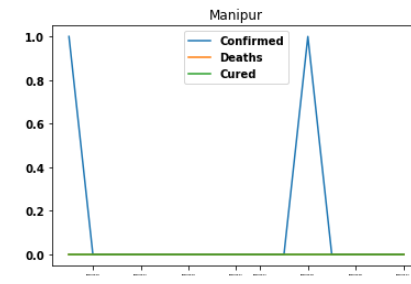
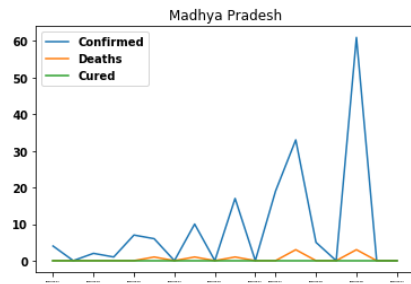
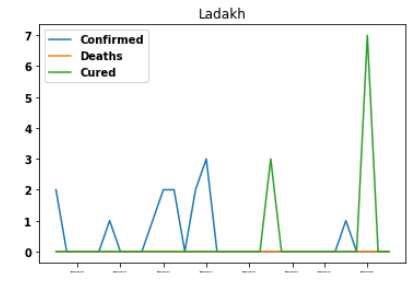
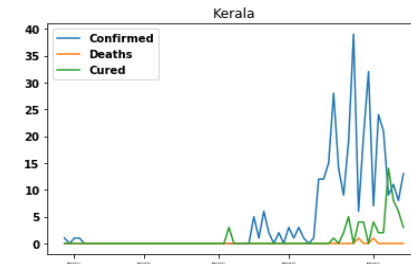
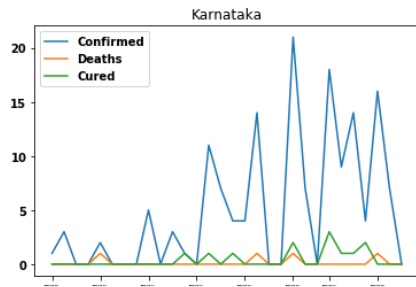
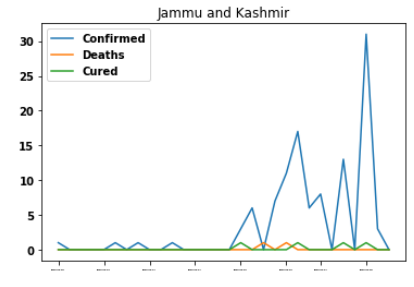
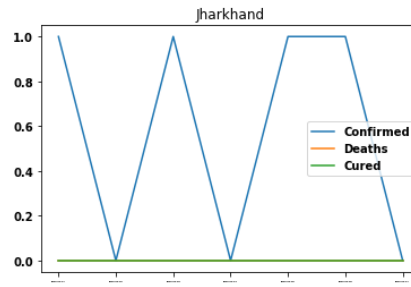
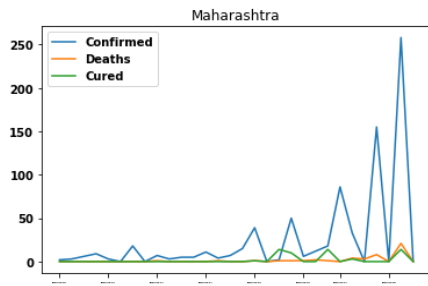
The graph is plotted no. of cases, recovered and deaths **per day and not cumulative**.

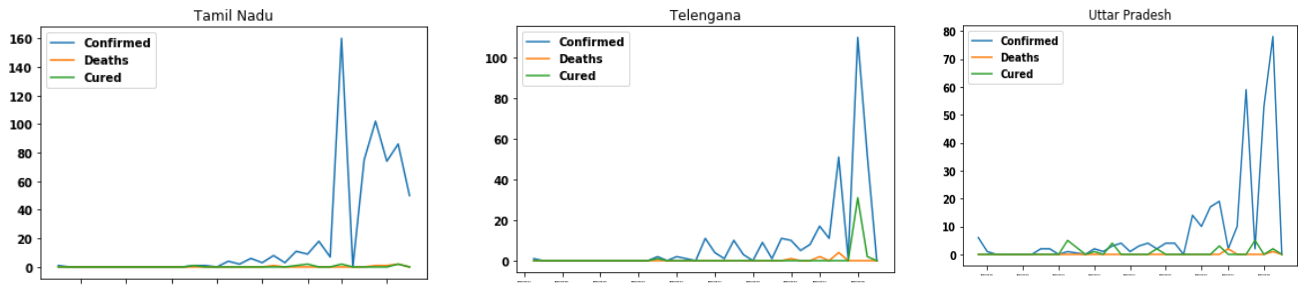
- Cases per day is found by subtracting two consecutive rows for a given state.

No. of Active cases will depend upon the number recovered and deceased at a given date.

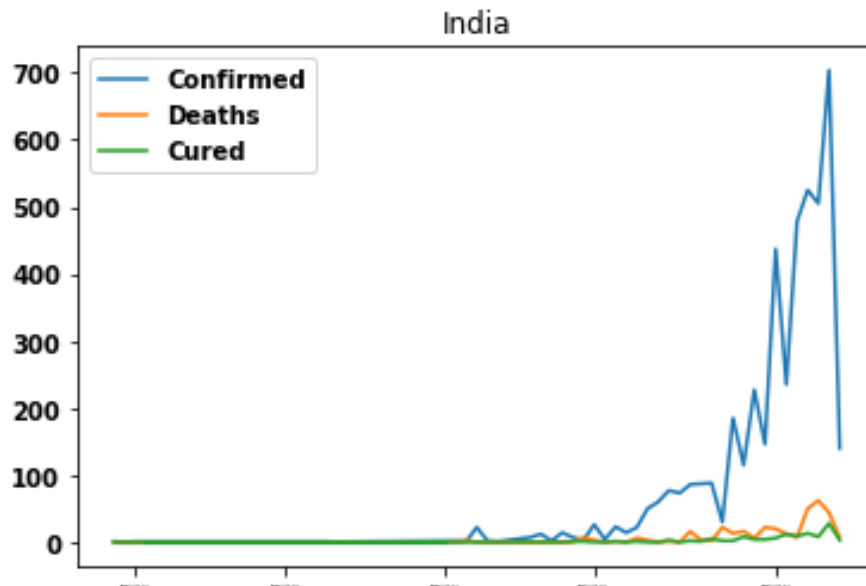
- No of active cases on a day is given by total cases - (recovered+deceased)







COUNTRY – WISE:



Q.3) Identify the positive cases on a state level. Quantify the intensity of virus spread for each state.*

Create a dataframe by grouping covid19 dataframe statewise and taking its sum.

● ● ● ● ● ●

Create a dataframe of population.csv. The value of pop density of a given state is extracted and added into a new column.

● ● ● ● ● ●

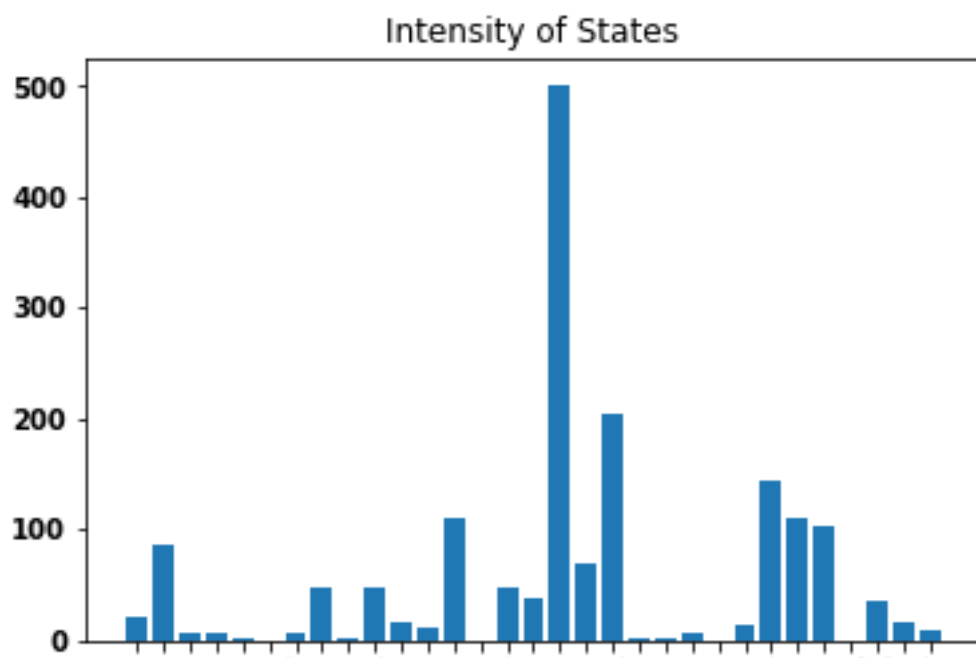
Calculate the intensity as cases/population using the two dfs

● ● ● ● ● ●

State	cases	intensity
Andaman and Nicobar Islands	10	21
Andhra Pradesh	266	87
Arunachal Pradesh	1	5
Assam	26	6
Bihar	32	2

Assumptions:

- Intensity's units are taken per km²
- The 'unassigned' row in covid-19.csv is dropped while calculating everything
- All the calculations are done as on 04.04.20



State	cases	intensity
Maharashtra	748	204
Tamil Nadu	621	111
Delhi	523	46
Kerala	327	38
Telengana	321	102

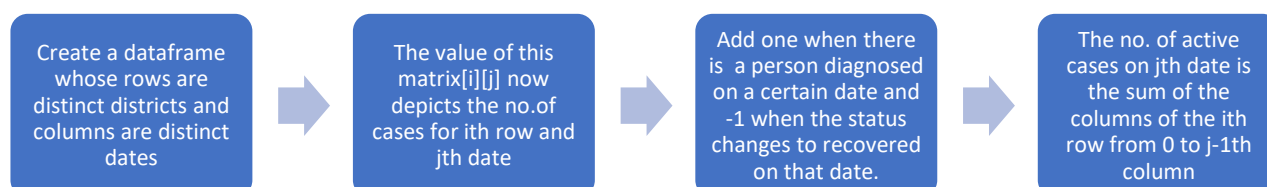
State	cases	intensity
Ladakh	14	500
Maharashtra	748	204
Rajasthan	288	143
Tamil Nadu	621	111
Jammu and Kashmir	109	111

Clearly, **Maharashtra** has the **highest No. of active** cases while **Ladakh** has the **maximum intensity** as on **04.04.10**.

- List places in the country which are active hotspots/clusters as on 10.04.2020.*
Hotspot is defined as an area in a city where 10 or more people have been tested positive.

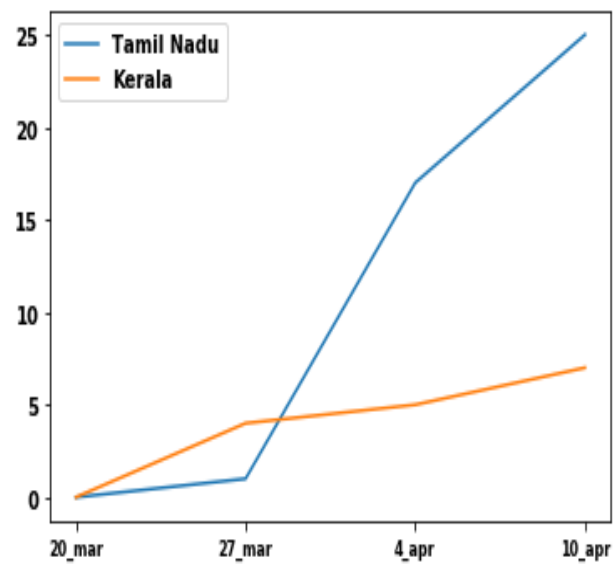
NOTE: For Q 4 New data sets from Kaggle are used

Process:



Index	2020-01-31	2020-02-05	2020-02-11	2020-02-18	2020-03-01	2020-03-08	2020-03-15	2020-03-22	2020-03-29	2020-04-05	2020-04-12	2020-04-19	2020-04-26	2020-05-03	2020-05-10	2020-05-17	2020-05-24	2020-05-31	2020-06-07	2020-06-14	2020-06-21	2020-06-28	2020-07-05	2020-07-12	2020-07-19	2020-07-26	2020-08-02	2020-08-09	2020-08-16	2020-08-23	2020-08-30	2020-09-06	2020-09-13	2020-09-20	2020-09-27	2020-10-04	2020-10-11	2020-10-18	2020-10-25	2020-11-01	2020-11-08	2020-11-15	2020-11-22	2020-11-29	2020-12-06	2020-12-13	2020-12-20	2020-12-27	2021-01-03	2021-01-10	2021-01-17	2021-01-24	2021-01-31	2021-02-07	2021-02-14	2021-02-21	2021-02-28	2021-03-06	2021-03-13	2021-03-20	2021-03-27	2021-04-03	2021-04-10	2021-04-17	2021-04-24	2021-05-01	2021-05-08	2021-05-15	2021-05-22	2021-05-29	2021-06-05	2021-06-12	2021-06-19	2021-06-26	2021-07-03	2021-07-10	2021-07-17	2021-07-24	2021-07-31	2021-08-07	2021-08-14	2021-08-21	2021-08-28	2021-09-04	2021-09-11	2021-09-18	2021-09-25	2021-10-02	2021-10-09	2021-10-16	2021-10-23	2021-10-30	2021-11-06	2021-11-13	2021-11-20	2021-11-27	2021-12-04	2021-12-11	2021-12-18	2021-12-25	2022-01-01	2022-01-08	2022-01-15	2022-01-22	2022-01-29	2022-02-05	2022-02-12	2022-02-19	2022-02-26	2022-03-05	2022-03-12	2022-03-19	2022-03-26	2022-04-02	2022-04-09	2022-04-16	2022-04-23	2022-04-30	2022-05-07	2022-05-14	2022-05-21	2022-05-28	2022-06-04	2022-06-11	2022-06-18	2022-06-25	2022-07-02	2022-07-09	2022-07-16	2022-07-23	2022-07-30	2022-08-06	2022-08-13	2022-08-20	2022-08-27	2022-09-03	2022-09-10	2022-09-17	2022-09-24	2022-10-01	2022-10-08	2022-10-15	2022-10-22	2022-10-29	2022-11-05	2022-11-12	2022-11-19	2022-11-26	2022-12-03	2022-12-10	2022-12-17	2022-12-24	2022-12-31	2023-01-07	2023-01-14	2023-01-21	2023-01-28	2023-02-04	2023-02-11	2023-02-18	2023-02-25	2023-03-04	2023-03-11	2023-03-18	2023-03-25	2023-04-01	2023-04-08	2023-04-15	2023-04-22	2023-04-29	2023-05-06	2023-05-13	2023-05-20	2023-05-27	2023-06-03	2023-06-10	2023-06-17	2023-06-24	2023-07-01	2023-07-08	2023-07-15	2023-07-22	2023-07-29	2023-08-05	2023-08-12	2023-08-19	2023-08-26	2023-09-02	2023-09-09	2023-09-16	2023-09-23	2023-09-30	2023-10-07	2023-10-14	2023-10-21	2023-10-28	2023-11-04	2023-11-11	2023-11-18	2023-11-25	2023-12-02	2023-12-09	2023-12-16	2023-12-23	2023-12-30	2024-01-06	2024-01-13	2024-01-20	2024-01-27	2024-02-03	2024-02-10	2024-02-17	2024-02-24	2024-03-02	2024-03-09	2024-03-16	2024-03-23	2024-03-30	2024-04-06	2024-04-13	2024-04-20	2024-04-27	2024-05-04	2024-05-11	2024-05-18	2024-05-25	2024-06-01	2024-06-08	2024-06-15	2024-06-22	2024-06-29	2024-07-06	2024-07-13	2024-07-20	2024-07-27	2024-08-03	2024-08-10	2024-08-17	2024-08-24	2024-08-31	2024-09-07	2024-09-14	2024-09-21	2024-09-28	2024-10-05	2024-10-12	2024-10-19	2024-10-26	2024-11-02	2024-11-09	2024-11-16	2024-11-23	2024-11-30	2024-12-07	2024-12-14	2024-12-21	2024-12-28	2025-01-04	2025-01-11	2025-01-18	2025-01-25	2025-02-01	2025-02-08	2025-02-15	2025-02-22	2025-02-29	2025-03-06	2025-03-13	2025-03-20	2025-03-27	2025-04-03	2025-04-10	2025-04-17	2025-04-24	2025-05-01	2025-05-08	2025-05-15	2025-05-22	2025-05-29	2025-06-05	2025-06-12	2025-06-19	2025-06-26	2025-07-03	2025-07-10	2025-07-17	2025-07-24	2025-07-31	2025-08-07	2025-08-14	2025-08-21	2025-08-28	2025-09-04	2025-09-11	2025-09-18	2025-09-25	2025-10-02	2025-10-09	2025-10-16	2025-10-23	2025-10-30	2025-11-06	2025-11-13	2025-11-20	2025-11-27	2025-12-04	2025-12-11	2025-12-18	2025-12-25	2026-01-01	2026-01-08	2026-01-15	2026-01-22	2026-01-29	2026-02-05	2026-02-12	2026-02-19	2026-02-26	2026-03-05	2026-03-12	2026-03-19	2026-03-26	2026-04-02	2026-04-09	2026-04-16	2026-04-23	2026-04-30	2026-05-07	2026-05-14	2026-05-21	2026-05-28	2026-06-04	2026-06-11	2026-06-18	2026-06-25	2026-07-02	2026-07-09	2026-07-16	2026-07-23	2026-07-30	2026-08-06	2026-08-13	2026-08-20	2026-08-27	2026-09-03	2026-09-10	2026-09-17	2026-09-24	2026-10-01	2026-10-08	2026-10-15	2026-10-22	2026-10-29	2026-11-05	2026-11-12	2026-11-19	2026-11-26	2026-12-03	2026-12-10	2026-12-17	2026-12-24	2026-12-31	2027-01-07	2027-01-14	2027-01-21	2027-01-28	2027-02-04	2027-02-11	2027-02-18	2027-02-25	2027-03-04	2027-03-11	2027-03-18	2027-03-25	2027-04-01	2027-04-08	2027-04-15	2027-04-22	2027-04-29	2027-05-06	2027-05-13	2027-05-20	2027-05-27	2027-06-03	2027-06-10	2027-06-17	2027-06-24	2027-07-01	2027-07-08	2027-07-15	2027-07-22	2027-07-29	2027-08-05	2027-08-12	2027-08-19	2027-08-26	2027-09-02	2027-09-09	2027-09-16	2027-09-23	2027-09-30	2027-10-07	2027-10-14	2027-10-21	2027-10-28	2027-11-04	2027-11-11	2027-11-18	2027-11-25	2027-12-02	2027-12-09	2027-12-16	2027-12-23	2027-12-30	2028-01-06	2028-01-13	2028-01-20	2028-01-27	2028-02-03	2028-02-10	2028-02-17	2028-02-24	2028-03-02	2028-03-09	2028-03-16	2028-03-23	2028-03-30	2028-04-06	2028-04-13	2028-04-20	2028-04-27	2028-05-04	2028-05-11	2028-05-18	2028-05-25	2028-06-01	2028-06-08	2028-06-15	2028-06-22	2028-06-29	2028-07-06	2028-07-13	2028-07-20	2028-07-27	2028-08-03	2028-08-10	2028-08-17	2028-08-24	2028-08-31	2028-09-07	2028-09-14	2028-09-21	2028-09-28	2028-10-05	2028-10-12	2028-10-19	2028-10-26	2028-11-02	2028-11-09	2028-11-16	2028-11-23	2028-11-30	2028-12-07	2028-12-14	2028-12-21	2028-12-28	2029-01-04	2029-01-11	2029-01-18	2029-01-25	2029-02-01	2029-02-08	2029-02-15	2029-02-22	2029-02-29	2029-03-06	2029-03-13	2029-03-20	2029-03-27	2029-04-03	2029-04-10	2029-04-17	2029-04-24	2029-05-01	2029-05-08	2029-05-15	2029-05-22	2029-05-29	2029-06-05	2029-06-12	2029-06-19	2029-06-26	2029-07-03	2029-07-10	2029-07-17	2029-07-24	2029-07-31	2029-08-07	2029-08-14	2029-08-21	2029-08-28	2029-09-04	2029-09-11	2029-09-18	2029-09-25	2029-10-02	2029-10-09	2029-10-16	2029-10-23	2029-10-30	2029-11-06	2029-11-13	2029-11-20	2029-11-27	2029-12-04	2029-12-11	2029-12-18	2029-12-25	2030-01-01	2030-01-08	2030-01-15	2030-01-22	2030-01-29	2030-02-05	2030-02-12	2030-02-19	2030-02-26	2030-03-05	2030-03-12	2030-03-19	2030-03-26	2030-04-02	2030-04-09	2030-04-16	2030-04-23	2030-04-30	2030-05-07	2030-05-14	2030-05-21	2030-05-28	2030-06-04	2030-06-11	2030-06-18	2030-06-25	2030-07-02	2030-07-09	2030-07-16	2030-07-23	2030-07-30	2030-08-06	2030-08-13	2030-08-20	2030-08-27	2030-09-03	2030-09-10	2030-09-17	2030-09-24	2030-10-01	2030-10-08	2030-10-15	2030-10-22	2030-10-29	2030-11-05	2030-11-12	2030-11-19	2030-11-26	2030-12-03	2030-12-10	2030-12-17	2030-12-24	2030-12-31	2031-01-07	2031-01-14	2031-01-21	2031-01-28	2031-02-04	2031-02-11	2031-02-18	2031-02-25	2031-03-04	2031-03-11	2031-03-18	2031-03-25	2031-04-01	2031-04-08	2031-04-15	2031-04-22	2031-04-29	2031-05-06	2031-05-13	2031-05-20	2031-05-27	2031-06-03	2031-06-10	2031-06-17	2031-06-24	2031-07-01	2031-07-08	2031-07-15	2031-07-22	2031-07-29	2031-08-05	2031-08-12	2031-08-19	2031-08-26	2031-09-02	2031-09-09	2031-09-16	2031-09-23	2031-09-30	2031-10-07	2031-10-14	2031-10-21	2031-10-28	2031-11-04	2031-11-11	2031-11-18	2031-11-25	2031-12-02	2031-12-09	2031-12-16	2031-12-23	2031-12-30	2032-01-06	2032-01-13	2032-01-20	2032-01-27	2032-02-03	2032-02-10	2032-02-17	2032-02-24	2032-03-02	2032-03-09	2032-03-16	2032-03-23	2032-03-30	2032-04-06	2032-04-13	2032-04-20	2032-04-27	2032-05-04	2032-05-11	2032-05-18	2032-05-25	2032-06-01	2032-06-08	2032-06-15	2032-06-22	2032-06-29	2032-07-06	2032-07-13	2032-07-20	2032-07-27	2032-08-03	2032-08-10	2032-08-17	2032-08-24	2032-08-31	2032-09-07	2032-09-14	2032-09-21	2032-09-28	2032-10-05	2032-10-12	2032-10-19	2032-10-26	2032-11-02	2032-11-09	2032-11-16	2032-11-23	2032-11-30	2032-12-07	2032-12-14	2032-12-21	2032-12-28	2033-01-04	2033-01-11	2033-01-18	2033-01-25	2033-02-01	2033-02-08	2033-02-15	2033-02-22	2033-02-29	2033-03-06	2033-03-13	2033-03-20	2033-03-27	2033-04-03	2033-04-10	2033-04-17	2033-04-24	2033-05-01	2033-05-08	2033-05-15	2033-05-22	2033-05-29	2033-06-05	2033-06-12	2033-06-19	2033-06-26	2033-07-03	2033-07-10	2033-07-17	2033-07-24	2033-07-31	2033-08-07	2033-08-14	2033-08-21	2033-08-28	2033-09-04	2033-09-11	2033-09-18	2033-09-25	2033-10-02	2033-10-09	2033-10-16	2033-10-23	2033-10-30	2033-11-06	2033-11-13	2033-11-20	2033-11-27	2033-12-04	2033-12-11	2033-12-18	2033-12-25	2034-01-01	2034-01-08	2034-01-15	2034-01-22	2034-01-29	2034-02-05	2034-02-12	2034-02-19	2034-02-26	2034-03-05	2034-03-12	2034-03-19	2034-03-26	2034-04-02	2034-04-09	2034-04-16	2034-04-23	2034-04-30	2034-05-07	2034-05-14	2034-05-21	2034-05-28	2034-06-04	2034-06-11	2034-06-18	2034-06-25	2034-07-02	2034-07-09	2034-07-16	2034-07-23	2034-07-30	2034-08-06	2034-08-13	2034-08-20	2034-08-27	2034-09-03	2034-09-10	2034-09-17	2034-09-24	2034-10-01	2034-10-08	2034-10-15	2034-10-22	2034-10-29	2034-11-05	2034-11-12	2034-11-19	2034-11-26	2034-12-03	2034-12-10	2034-12-17	2034-12-24	2034-12-31	2035-01-07	2035-01-14	2035-01-21	2035-01-28	2035-02-04	2035-02-11	2035-02-18	2035-02-25	2035-03-04	2035-03-11	2035-03-18	2035-03-25	2035-04-01	2035-04-08	2035-04-15	2035-04-22	2035-04-29	2035-05-06	2035-05-13	2035-05-20	2035-05-27	2035-06-03	2035-06-10	2035-06-17	2035-06-24	2035-07-01	2035-07-08	2035-07-15	2035-07-22	2035-07-29	2035-08-05	2035-08-12	2035-08-19	2035-08-26	2035-09-02	2035-09-09	2035-09-16	2035-09-23	2035-09-30	2035-10-07	2035-10-14	2035-10-21	2035-10-28	
-------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	--

Index	20_mar	27_mar	4_apr	10_apr	week1	week2	week3
Kerala	0	4	5	7	4	1	2
Delhi	0	1	2	2	1	1	0
Telangana	1	1	5	12	0	4	7
Rajasthan	1	2	7	12	1	5	5
Haryana	1	2	3	4	1	1	1
Uttar Pradesh	0	1	6	9	1	5	3
Ladakh	0	1	1	1	1	0	0
Tamil Nadu	0	1	17	25	1	16	8
Jammu and Kashmir	0	0	3	8	0	3	5
Karnataka	1	1	4	4	0	3	0
Maharashtra	2	4	6	10	2	2	4
Punjab	0	1	2	7	1	1	5
Andhra Pradesh	0	0	9	11	0	9	2
Uttarakhand	0	0	1	1	0	1	0
Odisha	0	0	1	1	0	1	0
Puducherry	0	0	0	0	0	0	0
West Bengal	0	0	1	1	0	1	0
Chandigarh	0	0	1	1	0	1	0
Chhattisgarh	0	0	0	1	0	0	1
Gujarat	0	1	4	7	1	3	3
Himachal Pradesh	0	0	0	1	0	0	1
Madhya Pradesh	0	1	3	6	1	2	3
Bihar	0	0	0	1	0	0	1
Manipur	0	0	0	0	0	0	0
Mizoram	0	0	0	0	0	0	0



From the table we can answer the above question clearly,

Maximum Increase:

WEEK 1: KERALA

WEEK 2: TAMIL NADU

WEEK 3: TAMIL NADU

Decrement is not significant enough to specify a single state.

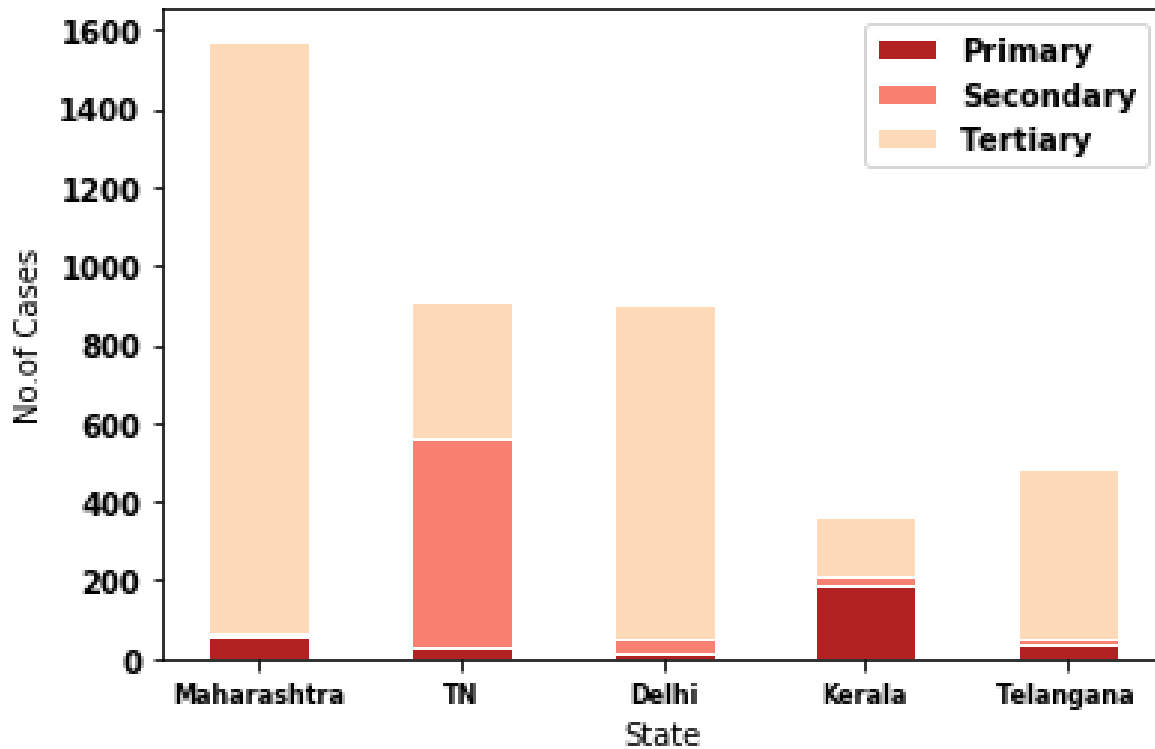
6. For the given data, identify cases with international travel history (primary case), personal contact with primary case (secondary case). Cases which do not fall in the primary and secondary fall into tertiary case. Quantify them based on the percentage for the top 5 states with maximum cases till 10.04.2020.*

For Q6 new datasets from Kaggle have been used

Assumption:

- The null values in notes column is replaced with 'Details Awaited' and is considered as tertiary cases.
- Primary cases are considered when a country name is present in the 'notes' column of Individualdetails.csv
- Secondary cases are considered when a few keywords like ['Delhi', 'Religious', 'Conference', 'delhi', 'Rajasthan', 'rajasthan', 'Kolkata', 'kolkata', 'P36', 'P35', 'P37'] are present in the notes column. This was done after a few background check on the reasons of secondary contacts and the kind of keywords used in the 'notes' column.
- All the other cases are classified as tertiary cases.

The top 5 states with maximum cases are plotted:



7. Find out the number of additional labs needed from the current existing labs (assume 100 tests per day per lab) with an increase rate of 10% cases per day from 11.04.2020 - 20.04.2020. List out any further assumptions considered.

For Q7 new datasets from Kaggle have been used

As there was no proper data on 20th April I have chosen to work for the dates from 9th April to 19th April

The following is the output of the code snippet submitted. This list out the details of samples taken per day, cases per day from 9th April to 19th April.

	DateTime	SamplesPerDay	CasesPerDay	ratio
0	09/04/20 21:00	16991.0	591.0	28.749577
1	10/04/20 21:00	16420.0	1167.0	14.070266
2	11/04/20 21:00	18044.0	831.0	21.713598
3	12/04/20 21:00	16374.0	609.0	26.886700
4	13/04/20 21:00	21806.0	1029.0	21.191448
5	14/04/20 21:00	27339.0	966.0	28.301242
6	15/04/20 21:00	29706.0	990.0	30.006061
7	16/04/20 21:00	28357.0	1284.0	22.084891
8	17/04/20 21:00	32167.0	1517.0	21.204351
9	18/04/20 21:00	37000.0	2267.0	16.321129
10	19/04/20 21:00	29463.0	1250.0	23.570400

Here the ratio column gives the rough estimate about no of samples to be taken in order to cater to the cases per day. So

$$Ratio = \frac{\text{samples per day}}{\text{cases per day}}$$

And the average value of the Ratio = 23.099 (or) approx 23.1 samples lead out to a positive cases.

Date	Cases per day	Est. samples per day	No of labs needed in a day	no of samples tested in a day if 355 labs are being operated
9-Apr	591	13651.509	136.51509	38.45495493
10-Apr	650.1	15016.6599	150.166599	42.30045042
11-Apr	715.11	16518.32589	165.1832589	46.53049546
12-Apr	786.621	18170.15848	181.7015848	51.18354501
13-Apr	865.2831	19987.17433	199.8717433	56.30189951
14-Apr	951.81141	21985.89176	219.8589176	61.93208946
15-Apr	1046.992551	24184.48094	241.8448094	68.12529841
16-Apr	1151.691806	26602.92903	266.0292903	74.93782825
17-Apr	1266.860987	29263.22193	292.6322193	82.43161108
18-Apr	1393.547085	32189.54413	321.8954413	90.67477218
19-Apr	1532.901794	35408.49854	354.0849854	99.7422494

The above table states the cases per day as on 9th April and is being incremented each day by 10%.

Thus the estimated samples in a day can be calculated by the product of Average Ratio and Cases per day

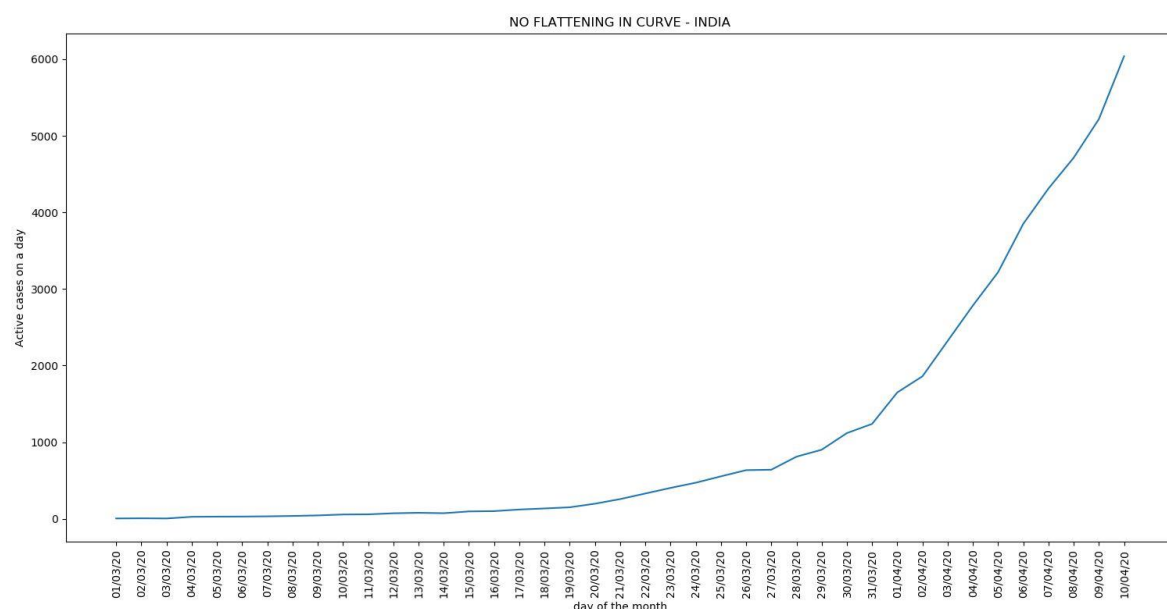
Now number of labs operating on 9th April is roughly assumed to be $\frac{\text{No of estimated samples on 9 April}}{\text{No of tests in a lab per day}} \approx 137$

Consequently on 19th April by the same calculation on the above step the requirement of labs is 355.

It can be verified by the last column of the table.

Therefore, Number of additional labs needed = 355 – 137 = 218.

8. Plot the number of cases starting from 1st March - 10th April. Based on this plot can you comment on the popular notion of 'flattening the curve'. *



For Q8 new datasets from Kaggle have been used

The above graph plots the data of number of active cases of COVID 19 in India from 1st MARCH to 10th April.

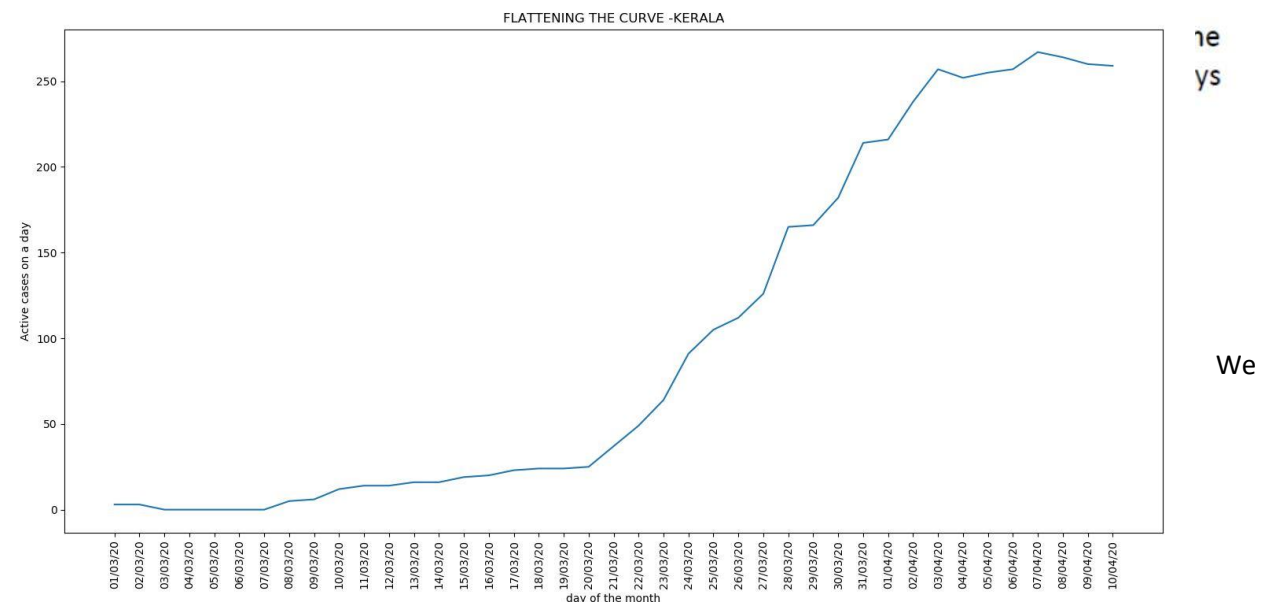
It is very obvious from the graph that never there was an intention in flattening the curve once it is rising. In the dates from 26th to 27th the curve is flat and then it is rising exponentially. In short India is trying their best in flattening the curve but has not achieved it still.

Q.9) "Notion of flattening the curve"

Flattening the curve is a [public health](#) strategy to slow down the spread of the COVID-19 virus during the [COVID-19 pandemic](#). The curve being flattened is the [epidemic curve](#), a visual representation of the number of infected people needing health care over time

Measures such as social distancing, sanitizing hands at regular intervals and stay-at-home orders reduce and delay the peak of active cases, allowing more time for healthcare capacity to increase and better cope with patient load. Time gained through thus *flattening the curve* can be used to *raise the line* of healthcare capacity to better meet surging demand.

Surprisingly, **KERALA** have achieved flattening the curve which is contradicting the fact that India is not which brings up to the point that public health inside few states are very strong and because of some states India as a whole takes more time in flattening the curve.



observed that over the 21 day period, the corona virus curve hasn't flattened significantly, this can be attributed to the size of our country, as enforcing a strict lockdown on 1.5billion people is next to impossible, this coupled with the lower testing rates at the beginning of the lockdown has prevented us from seeing any significant flattening of the curve. However having said that, the lockdown has definitely helped curb the spread of the virus, as without it the situation would have been much more grave. As the lockdown is gradually lifted, it is to be seen whether we can keep the virus from springing back up again.