



PONDICHERRY UNIVERISTY

**Department of
Computer Science**

M.Sc. 1st Year

**Understanding the concept of Diffusion on
COVID-19 virus on India**

SUBMITTED BY:

Rajarsi Saha

19370037

ABSTRACT

This project proposes a technique to visualize a spread of Covid-19 corona virus in the Indian subcontinent and analyse the data collected from a reputed data source. It also aims to further propose new ideas in studying future pandemics and spread of viruses and biological diseases. New data is created everyday on Covid-19 but no proper method has been created to visualize the spread and understand this pandemic. This project proposes to provide an insight into the spread of the above-mentioned virus right from the beginning of the year, i.e. January, 2020 up to the present day in April, 2020 via live data collected from reliable source. We will gradually see how the virus started from the state of Kerala and spread across the subcontinent within weeks. Concepts of diffusion will be explained through visualisations over the datasets. Various tools like Social Network Visualizer and the Python programming language has been used to proceed with this project.

ACKNOWLEDGEMENT

In performing my project, I had to take the help and guidance of some helpful people, who deserve my greatest gratitude. The completion of this project gives me much pleasure. I would like to extend my heartfelt gratitude to **Dr. S. Sivasathya, Professor, Department of Computer Science, Pondicherry University**, for her valuable time in giving us good advice for the project.

In addition, I would like to thank the online course teachers, for guiding me through the concepts for last minute doubt clearances. Also I like to thank my family for supporting me in times of this lockdown for understanding all my problems in the due course of implementation of this project.

Many people, especially my classmates and my faculty have given my valuable suggestions on this proposal which gave me an inspiration to improve my project. I thank all the people for their help directly or indirectly in completing my project.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Social network analysis (SNA), in essence, is not a formal theory in social science, but rather an approach for investigating social structures, which is why SNA is often referred to as structural analysis. The most important difference between social network analysis and the traditional or classic social research approach is that the contexts of the social actor, or the relationships between actors are the first considerations of the former, while the latter focuses on individual properties. A social network is a group of collaborating, and/or competing individuals or entities that are related to each other. It may be presented as a graph, or a multi-graph; each participant in the collaboration or competition is called an actor and depicted as a node in the graph theory. Valued relations between actors are depicted as links, or ties, either directed or undirected, between the corresponding nodes. Actors can be persons, organizations, or groups – any set of related entities. As such, SNA may be used on different levels, ranging from individuals, web pages, families, small groups, to large organizations, parties, and even to nations [1].

Coronavirus: a type of common virus that infects humans, typically leading to an upper respiratory infection (URI). Seven different types of human coronavirus have been identified. Most people will be infected with at least one type of coronavirus in their lifetime. The viruses are spread through the air by coughing and sneezing, close personal contact, touching an object or surface contaminated with the virus and rarely, by faecal contamination. The illness caused by most coronaviruses usually lasts a short time and is characterized by runny nose, sore throat, feeling unwell, cough, and fever [2].

Examples of human coronaviruses that have been reported to cause severe symptoms include the MERS-CoV (the beta coronavirus that causes Middle East Respiratory Syndrome, or MERS), SARS-CoV (the beta coronavirus that causes severe acute respiratory syndrome, or SARS), and the new 2019 Novel Coronavirus (2019-nCoV) outbreak that began in Wuhan, China [2].

Covid-19: Coronavirus disease 2019 (COVID-19) is defined as illness caused by a novel coronavirus now called severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2; formerly called 2019-nCoV), which was first identified amid an outbreak of respiratory illness cases in Wuhan City, Hubei Province, China. It was initially reported to the WHO on December 31, 2019. On January 30, 2020, the WHO declared the COVID-19 outbreak a global health emergency. On March 11, 2020, the WHO declared COVID-19 a global pandemic, its first such designation since declaring H1N1 influenza a pandemic in 2009 [3].

Data source: **Kaggle** is reliable data source for collecting live datasets for various live projects and work. Some with the collaboration of various national and global databases create these data sets to make lives of people like us much easier. Data with proper attributes and accuracy can well be found on Kaggle. Such is one dataset that I have used here for my project. The link to the dataset is given below:

covid19-corona-virus-india-dataset: <https://www.kaggle.com/imdevskp/covid19-corona-virus-india-dataset/download>

This project proposes a carry out a visual inspection on the various stages of the spread of corona virus in Indian subcontinent right from Jan, 2020 up to April, 2020 and show the phenomenon of Diffusion in the network of nodes (states). The entire time period has been divided into 8 parts to better understand during which period the virus really took over the country and can be called as a pandemic. The diffusion can be better understood by recognising from which node or state the virus first hit.

1.2 OBJECTIVE

- My primary objective is to first receive the data from the data source and segregate it into 8 time periods starting from January 1, 2020 to April 30, 2020. The reason why a post-dated period has been taken into consideration is just for the fact that we will be able to use this project and work on the dataset for upcoming data, i.e. data that is going to be available till April 30, 2020
- My next objective is to produce adjacency matrices to use in the Network Visualization Software for better understanding the data.
- And my third and final objective is to understand the concept of diffusion using the graph images available from the Network Visualizer.

CHAPTER 2: SURVEY OF TECHNOLOGIES

2.1 PYTHON 3.8

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development [4].

Link to Python website: <https://www.python.org/>

2.2 SOCNETV (SOCIAL NETWORK VISUALIZER)

Social Network Visualizer (SocNetV) is an open-source project to build a flexible and user-friendly, cross-platform tool for social network analysis and visualization, targeting primarily the social researcher. It is developed in C++ and Qt, an open-source development toolkit and runs on Linux, OS X and Windows.

SocNetV offers an easy to use Graphical User Interface (see User Interface overview), which lets you construct social networks with a few clicks on a virtual canvas or load networks of various formats (GraphML, GraphViz, Adjacency, EdgeList, Pajek, UCINET, GML, etc, see more in Supported Formats) and modify them to suit your needs.

Random networks can be created using various random network generation models (Barabási–Albert Scale-Free, Erdős–Rényi, Watts-Strogatz Small-World, d-regular, ring lattice, etc). Read more about them in section Random network creation.

The application computes standard graph theory and network cohesion metrics, such as density, diameter, geodesic distances (shortest path lengths), clustering coefficient (see Coefficient), walks, connectedness, eccentricity, etc. Read details in: Cohesion measures.

It also offers structural statistics, such as node and network centrality and prestige metrics, i.e. betweenness, eigenvector and closeness centrality, proximity and pagerank prestige. Read more in: Centralities and Prestige.

Fast algorithms for community detection, such as triad census, clique census, etc are included. Read more in: Community detection.

With SocNetV you can also perform structural equivalence analysis, using hierarchical clustering, actor similarities and tie profile dissimilarities, Pearson coefficients, etc. See more in Structural Equivalence methods.

For meaningful visualization of a social network, SocNetV supports various layout algorithms and models. For example, you can embed a layout model (radial, level, nodal sizes and color) based on a prominence index (i.e. Degree Centrality score). Or you can choose one of the force-directed placement algorithms (i.e. Kamada-Kawai Spring Embedder, Fruchterman-Reingold, etc). See more in Visualization and layout algorithms [5].

Link to SocialNetV: <https://socnetv.org/>

2.3 DIFFUSION

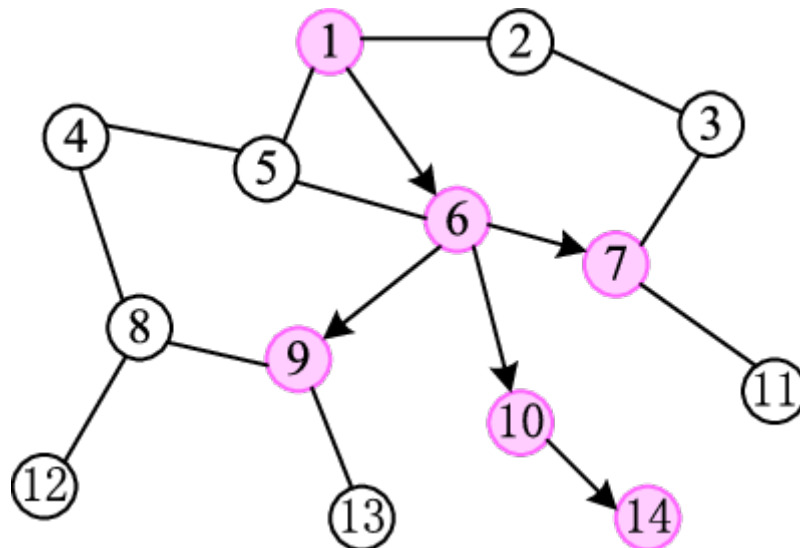


Fig 1: Diffusion

Let x_d^t denote the fraction of those agents with degree d who have adopted the behaviour B by time t , and let x^t denote the link-weighted fraction of agents who have adopted by time t . That is,

$$dx^t = \sum_d \frac{x_d^t dP(d)}{\bar{d}}$$

where \bar{d} is the average degree under P . The reason for weighting by links is standard: $\frac{dP(d)}{\bar{d}}$ is the probability that any given neighbour of some agent is of degree d (under the presumption that there is no correlation in degrees of linked agents).

We analyse a simple dynamic that leads to an overall equilibrium of the system. We begin with some random perturbation where x_d^0 of the agents of degree d have adopted. Given this, we then check each agent's best response to the system. This leads to a new x_d^1 for each d . Iterating on this process, we show that the system will eventually converge to a steady state. The convergence point is an equilibrium in the sense that given the state of the system, no additional agents wish to adopt, and none of the agents who have adopted would like to change their minds.

Given the complexity of the system, we use a standard technique for estimating the solutions. Namely, we use a mean-field analysis to estimate the proportion of the population that will have adopted at each time. This is described as follows. We start with the assumption that each i has the same initial fraction of neighbours adopting B , x^0 (and ignore the constraint that this be an integer). We also ignore the random distribution of initial adopters throughout the population. Each agent is matched with the actual distribution of the population.

So, i will adopt B in the first period if $v_i/c_i > 1/(g(d)x^0)$. Based on this, the fraction of degree d types who will adopt B in the first period is

We now have a new probability that a given link points to an adopter, which is $x^1 = \sum_d dP(d)x_d^1 / \bar{d}$. Iterating on this, at time t we get $x_d^t = 1 - F[1/(g(d)x^{t-1})]$. This gives us an equation:

$$x^t = \frac{1}{\bar{d}} \sum_d dP(d)(1 - F[1/(g(d)x^{t-1})]) \quad (1)$$

or

$$x^t = 1 - \frac{1}{\bar{d}} \sum_d dP(d)F\left[\frac{1}{g(d)x^{t-1}}\right] \quad (2)$$

Let us note a few things about this system. The right-hand side is non-decreasing in x^{t-1} , and when starting with $x^{t-1} = 0$ the generated next period level of adoption is $x^t = 0$ (noting that $F(\infty) = 1$). Provided $x^1 \geq x^0$, this system converges upwards to some point above x^0 . Note that this happens even if we allow the initial adopters to only stay adopters if they prefer to. Once we have gotten to x^1 , this includes exactly those who prefer to have adopted given the initial shock of x^0 , and now the level is either above or below x^0 , depending on the specifics of the system.

So, we can ask what minimal x^0 is needed in order to have the action B diffuse throughout the population; that is, to have x^t converge to a point above the initial point. We call this minimal x^0 the tipping point of the system. We can then also ask what x^t converges to.

In order to gain some insights regarding how the network structure (as captured through P) and how preferences vary with degree (as captured through g), we examine a case where F is the uniform distribution on some interval $[0, b]$.

In that case, (2) becomes

$$x^t = 1 - \sum_d \frac{dP(d)}{\bar{d}} \min \left[1, \frac{1}{bg(d)x^{t-1}} \right] \quad (3)$$

In a case where x^{t-1} is large enough so that $bg(d)x^{t-1} \geq 1$ for each d , then we can rewrite this as [5]

$$x^{t-1}(1 - x^t) = \sum_d \frac{dP(d)}{b\bar{d}g(d)} \quad (4)$$

2.4 CASCADING BEHAVIOUR

In any network, there are two obvious equilibria to this networkwide coordination game: one in which everyone adopts A, and another in which everyone adopts B. Guided by diffusion questions, we want to understand how easy it is, in a situation, to “tip” the network from one of these equilibria to the other. We also want to understand what other “intermediate” equilibria look like — states of coexistence where A is adopted in some parts of the network and B is adopted in others.

Specifically, we consider the following type of situation. Suppose that everyone in the network is initially using B as a default behaviour. Then, a small set of “initial adopters” all decide to use A. We will assume that the initial adopters have switched to A for some reason outside the definition of the coordination game — they have somehow switched due to a belief in A’s superiority, rather than by following payoffs — but we’ll assume that all other nodes continue to evaluate their payoffs using the coordination game. Given the fact that the initial adopters are now using A, some of their neighbours may decide to switch to A as well, and then some of their neighbours might, and so forth, in a potentially cascading fashion. When does this result in every node in the entire network eventually switching over to A? And when this isn’t the result, what causes the spread of A to stop? Clearly the answer will depend on the network structure, the choice of initial adopters, and the value of the threshold q that nodes use for deciding whether to switch to A.

The above discussion describes the full model. An initial set of nodes adopts A while everyone else adopts B. Time then runs forward in unit steps; in each step, each node uses the threshold rule to decide whether to switch from B to A. The process stops either when every node has switched to A, or when we reach a step where no node wants to switch, at which point things have stabilized on coexistence between A and B.

Let’s consider an example of this process using the social network in Figure 3(a).

- Suppose that the coordination game is set up so that $a = 3$ and $b = 2$; that is, the payoff to nodes interacting using behaviour A is $3/2$ times what it is with behaviour B. Using the threshold formula, we see that nodes will switch from B to A if at least a $q = 2/(3 + 2) = 2/5$ fraction of their neighbour are using A.

- Now, suppose that nodes v and w form the set of initial adopters of behaviour A, while everyone else uses B. (See Figure 3(b) where dark circles denote nodes adopting A and lighter circles denote nodes adopting B.) Then after one step, in which each of the other nodes evaluates its behaviour using the threshold rule, nodes r and t will switch to A: for each of them, $2/3 > 2/5$ of their neighbours are now using A. Nodes s and u do not switch, on the other hand, because for each of them, only $1/3 < 2/5$ of their neighbours are using A.

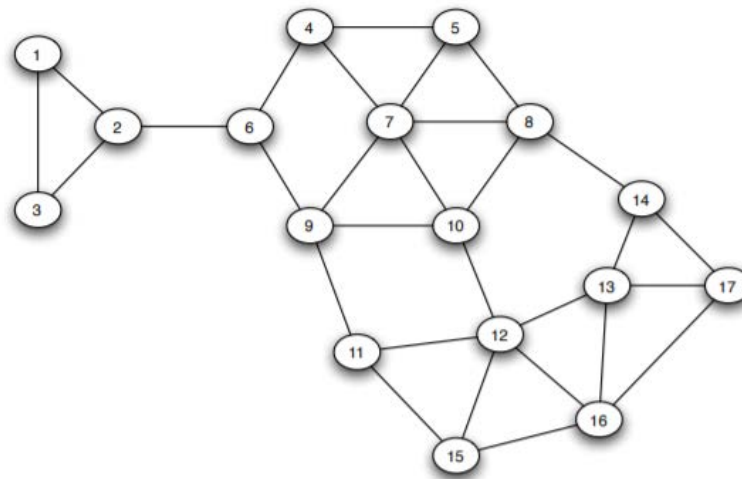


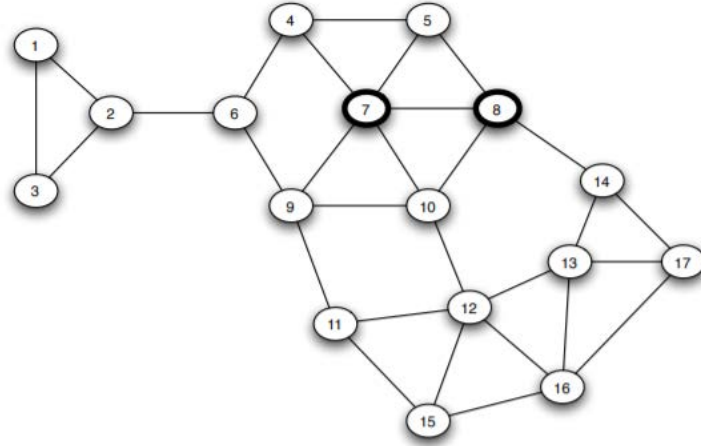
Fig 2: A larger example.

- In the next step, however, nodes s and u each have $2/3 > 2/5$ of their neighbours using A, and so they switch. The process now comes to an end, with everyone in the network using A

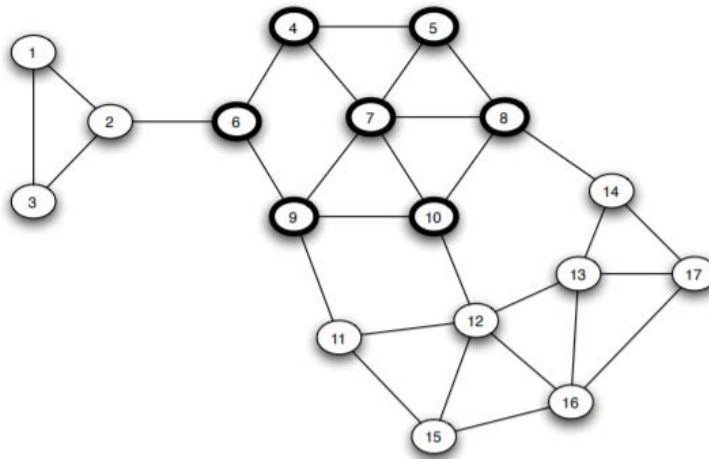
Notice how the process really is a chain reaction: nodes v and w aren't able to get s and u to switch by themselves, but once they've converted r and t, this provides enough leverage.

It's also instructive to consider an example in which the adoption of A continues for a while but then stops. Consider the social network in Figure 2, and again let's suppose that in the A-B coordination game, we have $a = 3$ and $b = 2$, leading to a threshold of $q = 2/5$. If we start from nodes 7 and 8 as initial adopters (Figure 19.5(a)), then in the next three steps we will first see (respectively) nodes 5 and 10 switch to A, then nodes 4 and 9, and then node 6. At this point, no further nodes will be willing to switch, leading to the outcome in Figure 3(b).

We'll call this chain reaction of switches to A a cascade of adoptions of A, and we'd like to distinguish between two fundamental possibilities: (i) that the cascade runs for a while but stops while there are still nodes using B, or (ii) that there is a complete cascade, in which every node in the network switches to A. We introduce the following terminology for referring to the second possibility.



(a) Two nodes are the initial adopters



(b) The process ends after three steps

Figure 3: Starting with nodes 7 and 8 as the initial adopters, the new behaviour A spreads to some but not all of the remaining nodes.

Consider a set of initial adopters who start with a new behaviour A, while every other node starts with behaviour B. Nodes then repeatedly evaluate the decision to switch from B to A using a threshold of q . If the resulting cascade of adoptions of A eventually causes every node to switch from B to A, then we say that the set of initial adopters causes a complete cascade at threshold q [7].

CHAPTER 3: REQUIREMENTS & ANALYSIS

3.1 PROBLEM DEFINITION

To demonstrate the use of the proposed method to visualise and analyse the data obtained from sources in the form of graphs for better understanding the concept of diffusion in the case of the spread of the pandemic coronavirus COVID-19 in the Indian subcontinent from one location to almost all states now at the end of the period taken into consideration for the project.

The system will give us the segregated data i.e., filtered data only containing the total number of cases both recovered and uncured of all the state as and when data is available. The system will then take those filtered data and calculate the adjacency matrix for plotting of the graphs with an external software. The external software will finally give us the final output graph images of the plots and save them for future use and reference.

The concept of Diffusion:

The concept of Diffusion plays an important role in the fact that it explains why the pandemic network in the Indian subcontinent keeps increasing every day and will keep increasing till the whole network is saturated. From the very definition of Diffusion, (definition given below) it is quite evident that a disease that is concentrated in a region will with high chances spread to a region with lower concentration or zero concentration.

Definition: Diffusion is the net movement of anything (for example, atom, ions, molecules) from a region of higher concentration to a region of lower concentration. The concept of diffusion is widely used in many fields, including physics (particle diffusion), chemistry, biology, sociology, economics, and finance (diffusion of people, ideas, and price values). The central idea of diffusion, however, is common to all of these: an object (for example, atom, idea, etc.) undergoing diffusion spreads out from a point or location at which there is a higher concentration of that object [8].

3.2 SOFTWARE AND HARDWARE REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

- Operating Systems: Windows 10/ Windows 7 Service pack1/ Windows Server 2016/ Windows Server 2012 R2/ Windows Server 2012
- RAM: 4 GB
- Disk Space: 2 GB of HDD space for Python and various other modules of Python like numpy and pandas.
- Processor: Any Intel or AMD x86-64 processor
- Browser: Internet Explorer 9 and above, Firefox, Google Chrome

3.2.2 SOFTWARE REQUIREMENTS

- Python 3.x
- Social NetV 2.5 (Social Network Visualizer ver. 2.5)
- Jupyter Notebook (optional)

CHAPTER 4: SYSTEM DESIGN

4.1 BASIC MODULES

This project is broadly divided into 3 modules:

- 1 **Data filtering and segregation module:** This module solely focuses on filtering out unnecessary data and segregating data according to required time periods of about 15 days each.
- 2 **Adjacency matrix creation module:** This module entirely focuses on creation of adjacency matrices for every time period.
- 3 **Graph plotting module:** This task is basically implemented using an external software named **SocialNetV** where the adjacency matrices are imported and the software produces the required graph by itself.
- 4 **Bar chart plot module:** This module basically plots the latest data on corona virus cases all the country state wise through a vertical bar chart. **This is optional.**

These are implemented by the three components below:

- 1 **data_filter.ipynb:** This module is written in Python using Jupyter Notebook to first import the data in csv format into a dataframe. A dataframe is a component of pandas module in Python which helps in working on data analysis. A large dataset can be very easily imported into a dataframe and worked with. Data from the data source is taken and fed into filter which filters out all other columns except for the '**Total confirmed cases**' column. Now the datafram is further divided into 8 parts roughly of 15 days each starting from Jan 1, 2020 and ending at April 30, 2020.

These 8 parts are then filtered further for duplicates and finally the max total cases are taken into consideration and put into the dataframes. Now the 8 dataframes look a lot cleaner and healthier than before. These 8 parts are then exported into separate csv files and kept for future use and reference.

- 2 **adjacency_create.ipynb:** This module is used for creating adjacency matrices from the data that was saved in those 8 files in the previous section. This part is much easier. Here only the integer values are taken and stored as matrices. If a state has positive number of cases i.e., cases greater than zero then the corresponding value is stored in the matrix, otherwise that position is kept zero (0).

These matrices are now stored into separate csv files with only integer values and no column or row names are stored. The matrices basically look like square matrices and stored for further visualisation which is explained in next point.

- 3 **graph_visualization:** This is an external module and involves no coding. This is basically a software run on the data matrices created in the above methods. The adjacency matrices are imported into SocialNetV to visualize the data in the form of graph plot. We can make changes and tweak here with the software to give us various different models of the network that as created by the software from the dataset.

We can choose various different layout options from the menu to display the graph in different ways. These plots basically give us an idea of how diffusion works and how the corona virus had spread into the entire subcontinent.

- 4 **graph_plot.ipynb:** This is again a python module written in Jupiter notebook. This basically takes the filtered data, calculates the maximum cases per state in India and plots a bar chart which helps us to visualize the present scenario of the country. This an optional module and the progress of the project is not affected by the failure or success of this module.

4.2 DATA DESIGN

The data was taken from a reliable source named Kaggle. These people provide data about technically everything in this world and the data is collected and maintained by mostly researchers and scientists, basically field workers.

About Kaggle:

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

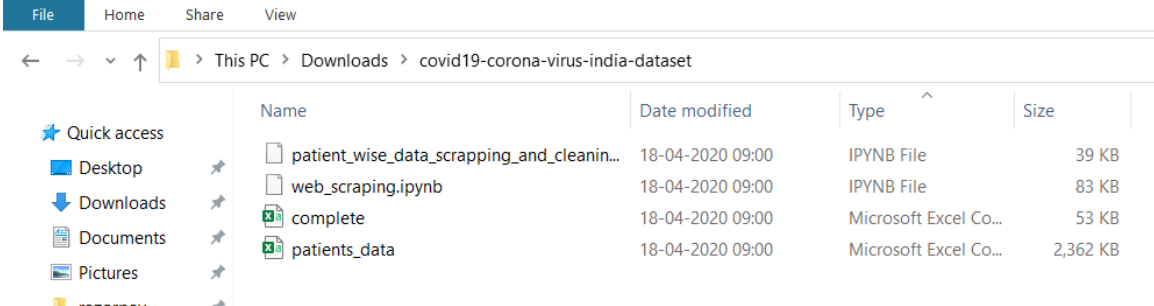
Kaggle got its start in 2010 by offering machine learning competitions and now also offers a public data platform, a cloud-based workbench for data science, and Artificial Intelligence education. Its key personnel were Anthony Goldbloom and Jeremy Howard. Nicholas Gruen was founding chair succeeded by Max Levchin. Equity was raised in 2011 valuing the company at \$25 million. On 8 March 2017, Google announced that they were acquiring Kaggle [9].

Link to Kaggle: <https://www.kaggle.com/>

The link to the data source is given below:

<https://www.kaggle.com/imdevskp/covid19-corona-virus-india-dataset/download>

This is basically a zip file that contains the following files:



Name	Date modified	Type	Size
patient_wise_data_scrapping_and_cleanin...	18-04-2020 09:00	IPYNB File	39 KB
web_scraping.ipynb	18-04-2020 09:00	IPYNB File	83 KB
complete	18-04-2020 09:00	Microsoft Excel Co...	53 KB
patients_data	18-04-2020 09:00	Microsoft Excel Co...	2,362 KB

Fig 4: Contents of the dataset folder

From the above contents I have used the **complete.csv** file which basically contains data in raw format

	A	B	C	D	E	F	G	H	I	J
	Date	Name of S	Total Conf	Total Conf Cured/Disch	Latitude	Longitude	Death	Total Confirmed cases		
2	30-01-2020	Kerala	1	0	0	10.8505	76.2711	0		1
3	31-01-2020	Kerala	1	0	0	10.8505	76.2711	0		1
4	01-02-2020	Kerala	2	0	0	10.8505	76.2711	0		2
5	02-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
6	03-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
7	04-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
8	05-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
9	06-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
10	07-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
11	08-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
12	09-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
13	10-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
14	11-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
15	12-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
16	13-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
17	14-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
18	15-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
19	16-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
20	17-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
21	18-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
22	19-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
23	20-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
24	21-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
25	22-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
26	23-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
27	24-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
28	25-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3
29	26-02-2020	Kerala	3	0	0	10.8505	76.2711	0		3

Fig 5: The contents of the file

4.3 PROCEDURAL DESIGN

The basic procedure that was followed is as follows:

- 1 Data is imported into the **data_filter** module
- 2 The data is taken into data frame
- 3 Data is clean and filtered
- 4 Data is segregated into 8 parts
- 5 Data is saved as csv files
- 6 Date from csv files is imported into **adjacency_create** module
- 7 Data is taken into dataframes
- 8 It is then processed to create corresponding adjacency matrix
- 9 Adjacency matrix is saved back into separate csv files
- 10 Adjacency matrices imported into SocialNetV
- 11 Graph is plotted for visualization

A few images that explain most of the activity:

```

jupyter data_filter Last Checkpoint: 10 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [33]: import pandas as pd
import numpy as np
import datetime

In [34]: covid_data = pd.read_csv('complete.csv')

In [39]: df = pd.DataFrame(covid_data)
states = df['Name of State / UT'].unique()
np.savetxt("states.csv", [states], delimiter=',', fmt='%s')
month_names = ['jan_1', 'jan_2', 'feb_1', 'feb_2', 'mar_1', 'mar_2', 'apr_1', 'apr_2']

In [36]: months = list()
months.append(df[(df['Date'] >= '2020-01-01') & (df['Date'] <= '2020-01-15')])
months.append(df[(df['Date'] >= '2020-01-16') & (df['Date'] <= '2020-01-31')])
months.append(df[(df['Date'] >= '2020-02-01') & (df['Date'] <= '2020-02-15')])
months.append(df[(df['Date'] >= '2020-02-16') & (df['Date'] <= '2020-02-29')])
months.append(df[(df['Date'] >= '2020-03-01') & (df['Date'] <= '2020-03-15')])
months.append(df[(df['Date'] >= '2020-03-16') & (df['Date'] <= '2020-03-31')])
months.append(df[(df['Date'] >= '2020-04-01') & (df['Date'] <= '2020-04-15')])
months.append(df[(df['Date'] >= '2020-04-16') & (df['Date'] <= '2020-04-30')])

In [37]: statewise = pd.DataFrame(columns = df.columns)
for month, name in zip(months, month_names):
    for state in states:
        temp = month[month['Name of State / UT'] == state]
        statewise = statewise.append(temp[temp['Total Confirmed cases'] == temp['Total Confirmed cases'].max()])
        statewise = statewise.drop_duplicates(subset = "Name of State / UT", keep = 'last')
        statewise = statewise[['Name of State / UT', 'Total Confirmed cases']]
        statewise.to_csv(str(str(name) + '_output.csv'), index=False)
  
```

Fig 6(a): Working of data_filter

```

jupyter adjacency_create Last Checkpoint: 9 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [115]: import pandas as pd
import numpy as np

In [128]: states = np.genfromtxt('states.csv', delimiter=',', dtype=str)

In [117]: month_names = ['jan_1', 'jan_2', 'feb_1', 'feb_2', 'mar_1', 'mar_2', 'apr_1', 'apr_2']
month_inputs = list()
for month in month_names:
    temp = pd.read_csv(month + '_output.csv')
    month_inputs.insert(month_names.index(month), temp)

In [127]: for month in month_names:
    adj = np.zeros((len(states), len(states)), dtype=int)
    arr = np.array(month_inputs[month_names.index(month)]['Name of State / UT'])
    for elem in arr:
        for i in range(len(states)):
            if elem == states[i]:
                j = int(np.where(states == elem)[0])
                for k in range(j+1):
                    adj[k][j] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed cases']
                    adj[j][k] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed cases']
                break
    np.savetxt(month + "_adjacency.csv", adj, delimiter=',', fmt='%s')

In [ ]:
  
```

Fig 6(b): Working of adjacency_create

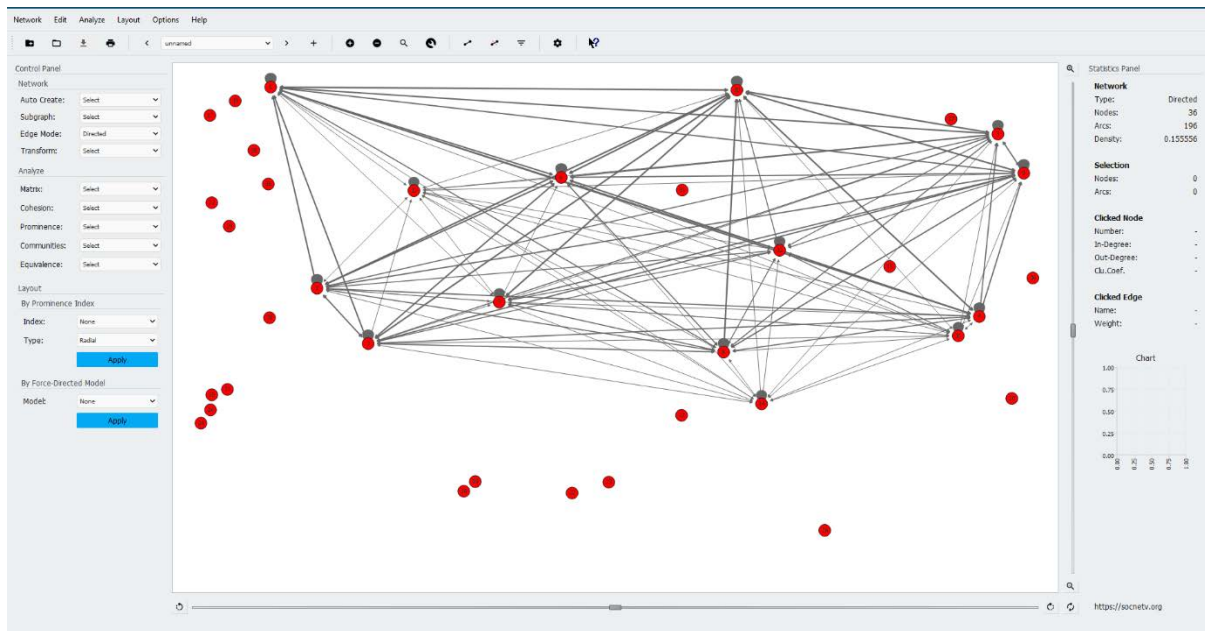


Fig 6(c): Graph visualization of April 1, 2020 – April 15, 2020

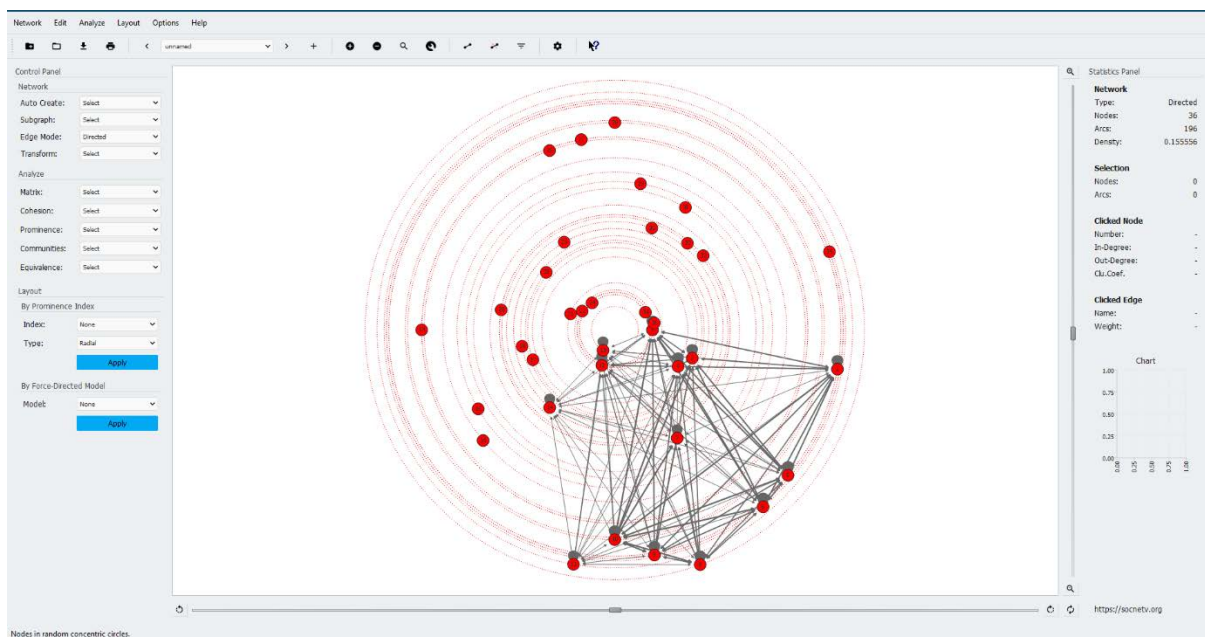


Fig 6(d): Graph visualization of April 1, 2020 – April 15, 2020 using a radial layout

CHAPTER 5: IMPLEMENTATION & TESTING

5.1 IMPLEMENTATION APPROACHES

The followed steps were followed during the entire implementation procedure:

Step 1 Code import and filtering of data

```
In [33]: import pandas as pd
import numpy as np
import datetime

In [34]: covid_data = pd.read_csv('complete.csv')

In [39]: df = pd.DataFrame(covid_data)
states = df['Name of State / UT'].unique()
np.savetxt("states.csv", [states], delimiter=',', fmt='%s')
month_names = ['jan_1', 'jan_2', 'feb_1', 'feb_2', 'mar_1', 'mar_2', 'apr_1', 'apr_2']

In [36]: months = list()
months.append(df[(df['Date'] >= '2020-01-01') & (df['Date'] <= '2020-01-15')])
months.append(df[(df['Date'] >= '2020-01-16') & (df['Date'] <= '2020-01-31')])
months.append(df[(df['Date'] >= '2020-02-01') & (df['Date'] <= '2020-02-15')])
months.append(df[(df['Date'] >= '2020-02-16') & (df['Date'] <= '2020-02-29')])
months.append(df[(df['Date'] >= '2020-03-01') & (df['Date'] <= '2020-03-15')])
months.append(df[(df['Date'] >= '2020-03-16') & (df['Date'] <= '2020-03-31')])
months.append(df[(df['Date'] >= '2020-04-01') & (df['Date'] <= '2020-04-15')])
months.append(df[(df['Date'] >= '2020-04-16') & (df['Date'] <= '2020-04-30')])
```

Fig 7(a): Filtering Data

Step 2 Segregation and saving into csv

```
In [37]: statewise = pd.DataFrame(columns = df.columns)
for month, name in zip(months, month_names):
    for state in states:
        temp = month[(month['Name of State / UT'] == state)]
        statewise = statewise.append(temp[temp['Total Confirmed cases'] == temp['Total Confirmed cases'].max()])
statewise = statewise.drop_duplicates(subset = "Name of State / UT", keep = 'last')
statewise = statewise[['Name of State / UT', 'Total Confirmed cases']]
statewise.to_csv(str(str(name) + '_output.csv'), index=False)
```

Fig 7(b): Segregation

Step 3 How the filtered data file looks like

	A	B	C	D
1	Name of State / UT	Total Confirmed cases		
2	Union Territory of Ladakh	10		
3	Union Territory of Jammu and Kashmir	4		
4	Union Territory of Chandigarh	1		
5	Kerala	387		
6	Delhi	1561		
7	Telangana	624		
8	Rajasthan	969		
9	Haryana	199		
10	Uttar Pradesh	660		
11	Tamil Nadu	1204		
12	Karnataka	260		
13	Maharashtra	2687		
14	Punjab	176		
15	Andhra Pradesh	483		
16	Uttarakhand	37		
17	Odisha	60		
18	Andhra Pradesh	7		
19	West Bengal	213		
20	Chhattisgarh	33		
21	Gujarat	650		
22	Chandigarh	21		
23	Himachal Pradesh	33		
24	Jammu and Kashmir	278		
25	Ladakh	17		
26	Madhya Pradesh	730		
27	Bihar	66		
28	Manipur	2		
29	Mizoram	1		

Fig 7(c): Filtered data file

Step 4 Import of data for conversion to adjacency matrix

```
month_index = month_names.index(month)
femb = bq.query('SELECT * FROM `bigquery-public-data.covid19.google` WHERE month = %s' % month_index)

femb.to_csv('femb.csv')

month_index = month_names.index(month)
femb = bq.query('SELECT * FROM `bigquery-public-data.covid19.google` WHERE month = %s' % month_index)

femb.to_csv('femb.csv')
```

Fig 7(d): Importing filtered and segregated data

Step 5 Conversion to adjacency matrix

```
In [127]: for month in month_names:
adj = np.zeros((len(states), len(states)), dtype=int)
arr = np.array(month_inputs[month_names.index(month)]['Name of State / UT'])
for elem in arr:
    for i in range(len(states)):
        if elem == states[i]:
            j = int(np.where(states == elem)[0])
            for k in range(j+1):
                adj[k][j] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed cases']
                adj[j][k] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed cases']
            break
np.savetxt(month + "_adjacency.csv", adj, delimiter=',', fmt='%s')
```

Fig 7(e): Creating adjacency matrix and saving as csv

Step 6 How the adjacency matrix looks like

#	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	
4	387	1561	624	969	199	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
5	1561	1561	624	969	199	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
6	624	624	624	969	199	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
7	969	969	969	969	199	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
8	199	199	199	199	199	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
9	660	660	660	660	660	660	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
10	1204	1204	1204	1204	1204	1204	1204	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
11	260	260	260	260	260	260	260	260	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
12	2687	2687	2687	2687	2687	2687	2687	2687	2687	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
13	176	176	176	176	176	176	176	176	176	176	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
14	483	483	483	483	483	483	483	483	483	483	483	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
15	37	37	37	37	37	37	37	37	37	37	37	37	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
16	60	60	60	60	60	60	60	60	60	60	60	60	60	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
17	7	7	7	7	7	7	7	7	7	7	7	7	7	7	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
18	213	213	213	213	213	213	213	213	213	213	213	213	213	213	213	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
19	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
20	650	650	650	650	650	650	650	650	650	650	650	650	650	650	650	650	650	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
22	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
23	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	278	17	730	66	2	1	11	7	32	27	1	2	1	0		
24	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	730	66	2	1	11	7	32	27	1	2	1	0			
25	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	730	66	2	1	11	7	32	27	1	2	1	0		
26	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	2	1	11	7	32	27	1	2	1	0		
27	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	11	7	32	27	1	2	1	0		
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	11	7	32	27	1	2	1	0	
29	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	7	32	27	1	2	1	0	
30	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	32	27	1	2	1	0		
31	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	27	1	2	1	0		
32	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	1	2	1	0	
33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
34	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	0
35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 7(f): Adjacency matrix

Step 7 Import adjacency matrix into SocialNetV

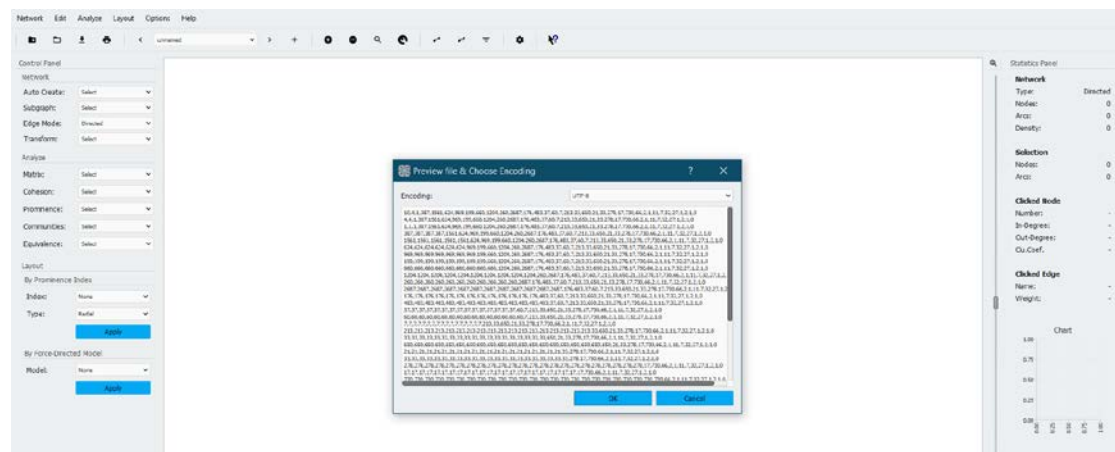


Fig 7(g): Importing Adjacency Matrix into SocialNetV

Step 8 Graph Plot visualization

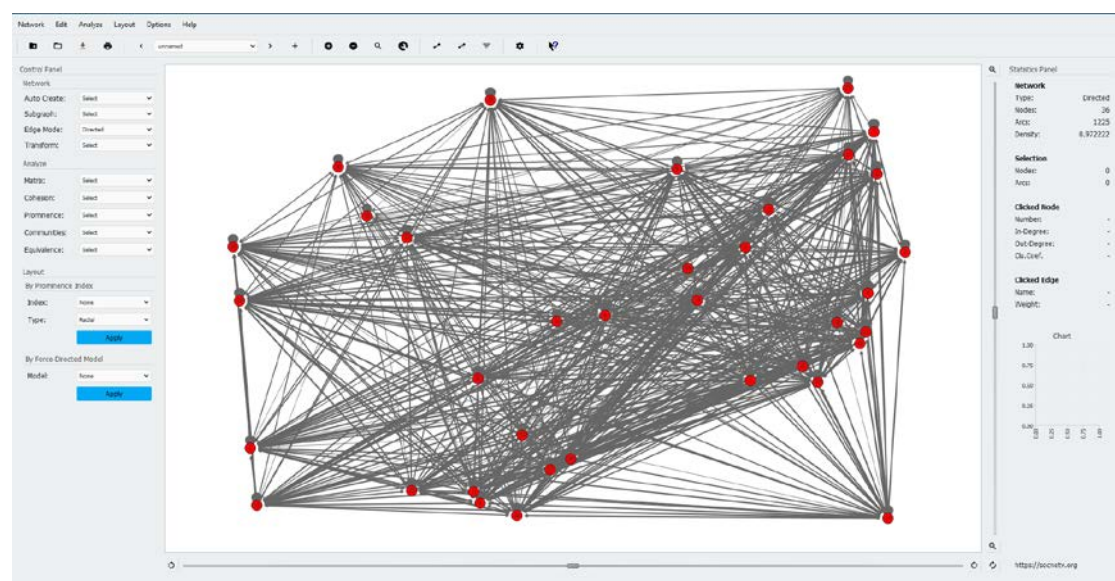


Fig 7(h): Graph Plot visualization on SocialNetV

Step 9 Image format (PNG)

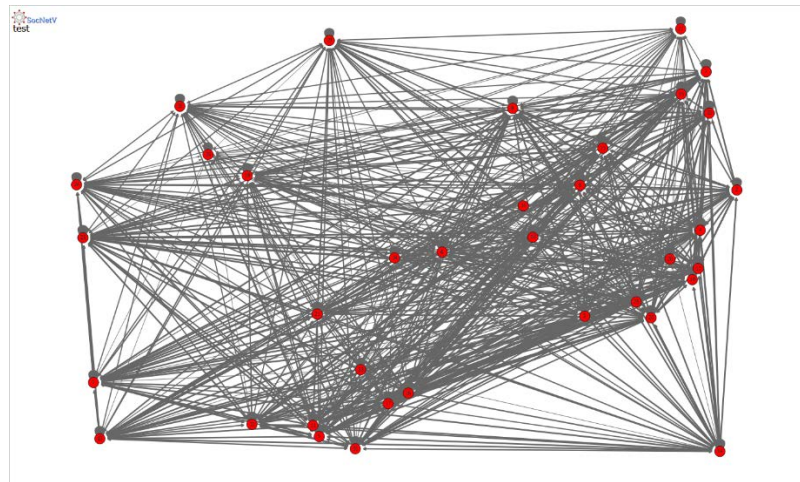


Fig 7(i): Image form of graph (img.png)

5.2 ANALYSIS

The following images have been collected after running the Social Network Visualizer software over all the available data matrices available to us. These data matrices are the ones that we created from the python codes in the first two phases. The following images have been in sequence of the time periods that was taken into consideration while dividing the main dataset. These graph visualizations provide enough insights into the diffusion factor in the spread of COVID-19 in India. Please note how the graph gradually becomes denser leading to the belief that the virus has spread more and more into the different states.

The images have been given timestamp heading for better understanding:

Jan 1, 2020 – Jan 15, 2020



Fig 8(a): Jan 1 – 15, 2020

Jan 15, 2020 – Jan 31, 2020

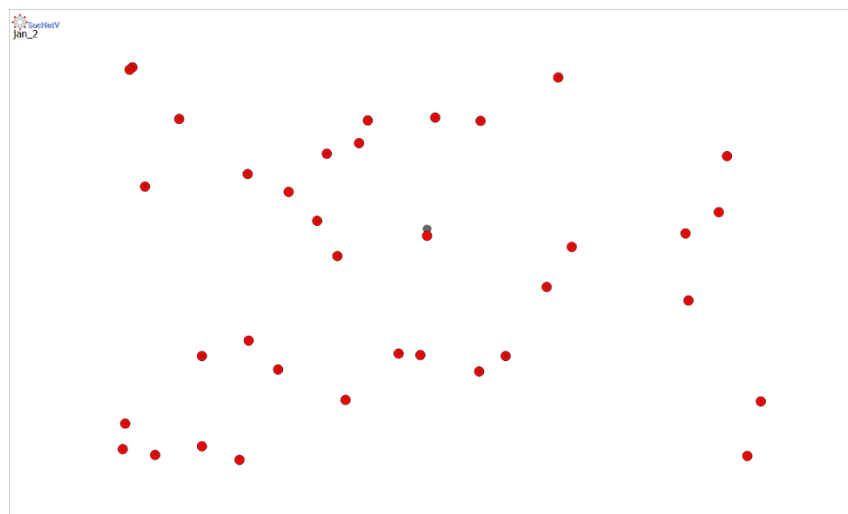


Fig 8(b): Jan 15 – 31, 2020

Feb 1, 2020 – Feb 15, 2020

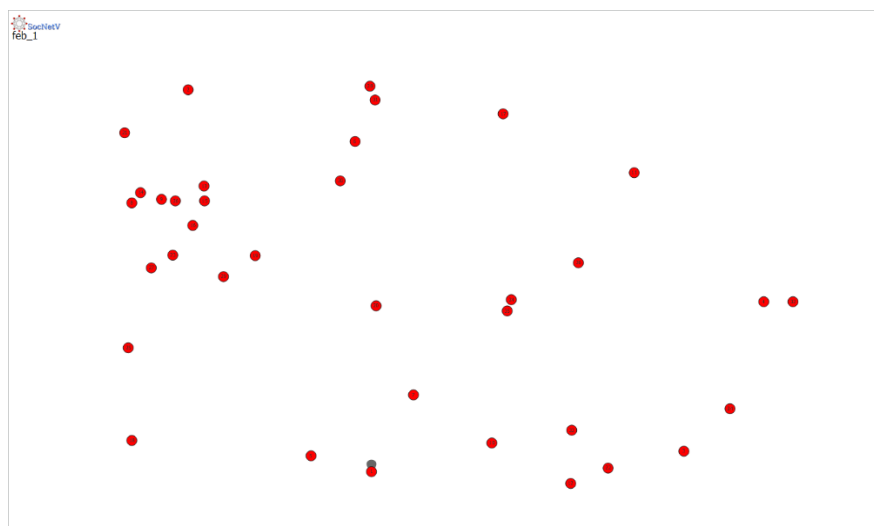


Fig 8(c): Feb 1 – 15, 2020

Feb 15, 2020 – Feb 29, 2020

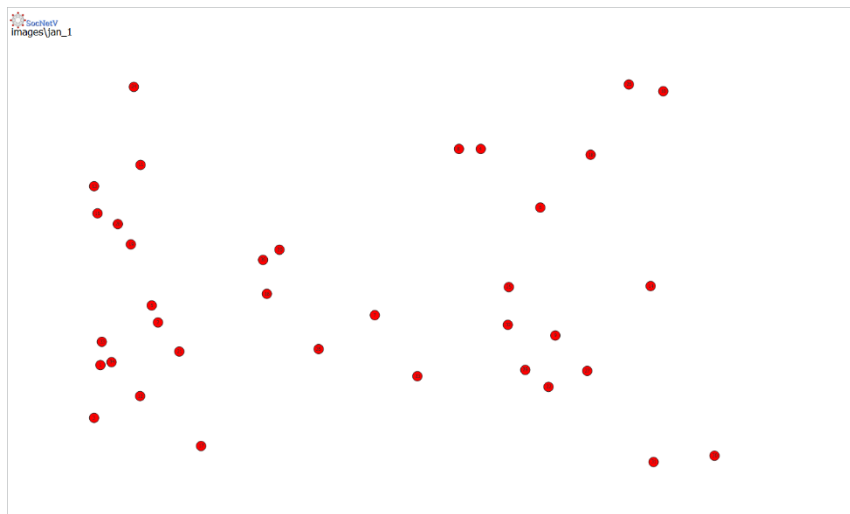


Fig 8(d): Feb 15 – 29, 2020

Mar 1, 2020 – Mar 15, 2020

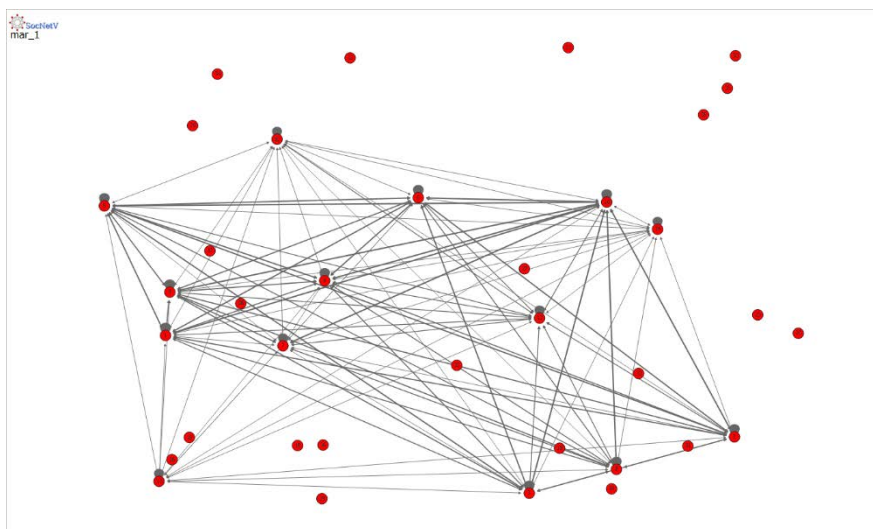


Fig 8(e): Mar 1 – 15, 2020

Mat 15, 2020 – Mar 31, 2020

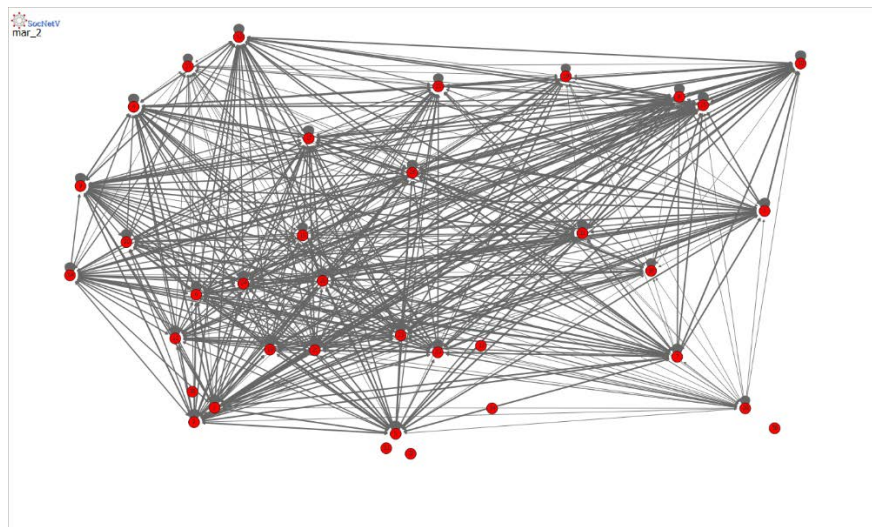


Fig 8(f): Mar 15 – 31, 2020

Apr 1, 2020 – Apr 15, 2020

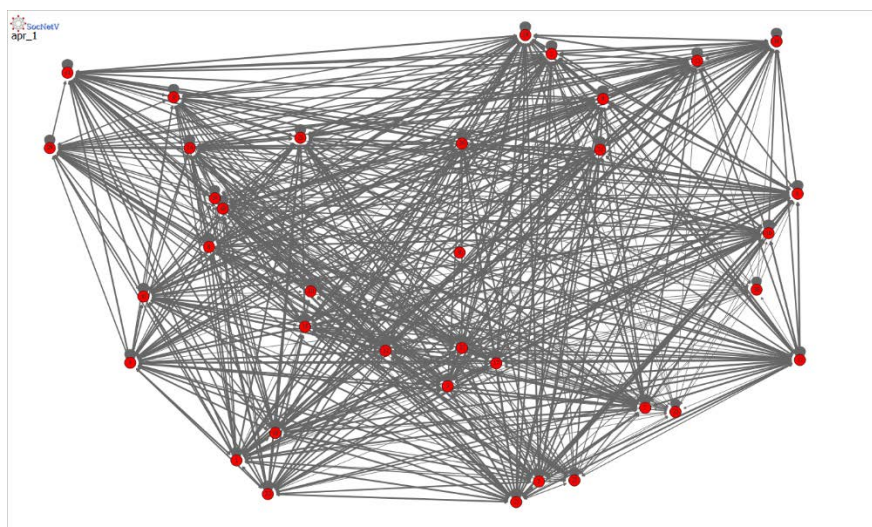


Fig 8(g): Apr 1 – 15, 2020

Apr 15, 2020 – Till date

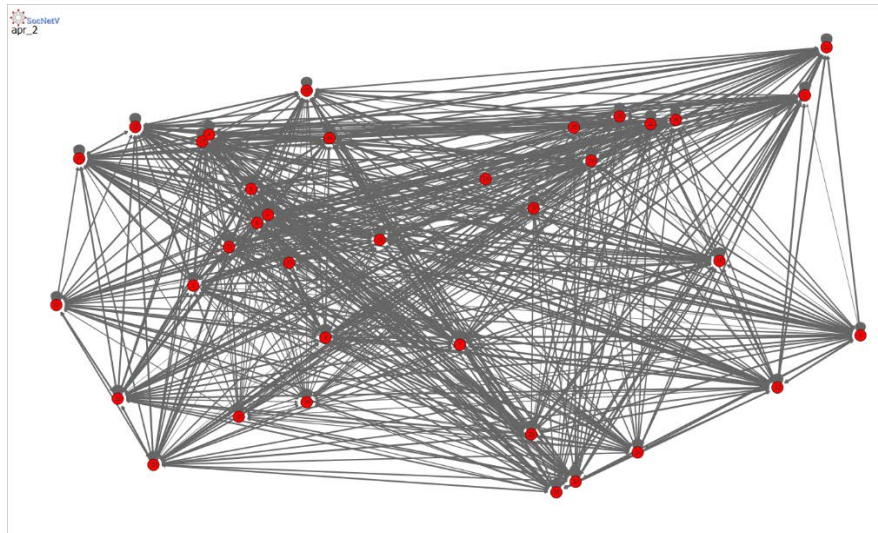


Fig 8(h): Apr 15, 2020 –

In the above images it is not hard for us to see that graph for **Mar 1 – 15, 2020** is way denser than the previous ones. On the other hand, we see that the graph for **Mar 15 – 31, 2020** is denser than the one for **Mar 1 – 15, 2020**. And we can very easily see that the rest of them are the densest. This is because the pandemic had already covered the entire country i.e., covered every state in the subcontinent.

This is a proper visualization of the concept of diffusion in the spread of this virus and disease. Diffusion starts taking place the day a single person got affected in Kerala in the period **Jan 15 – 31, 2020**. It spreads and now by the end of this month we have a country full of coronavirus cases.

The reason why we see the initial few images to be less dense and the subsequent images to be highly dense is because of the exponential rate at which the COVID-19 virus spread into the entire country. The exponential growth is the main factor why we have highly dense graphs in the end.

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Name of S	Total Conf	Total Conf	Cured/Disc	Latitude	Longitude	Death	Total Confirmed cases		
2	30-01-2020	Kerala	1	0	0	10.8505	76.2711	0	1		
3	31-01-2020	Kerala	1	0	0	10.8505	76.2711	0	1		
1086											

Fig 9: The no of cases in Jan, 2020

Latest Graph Visualization

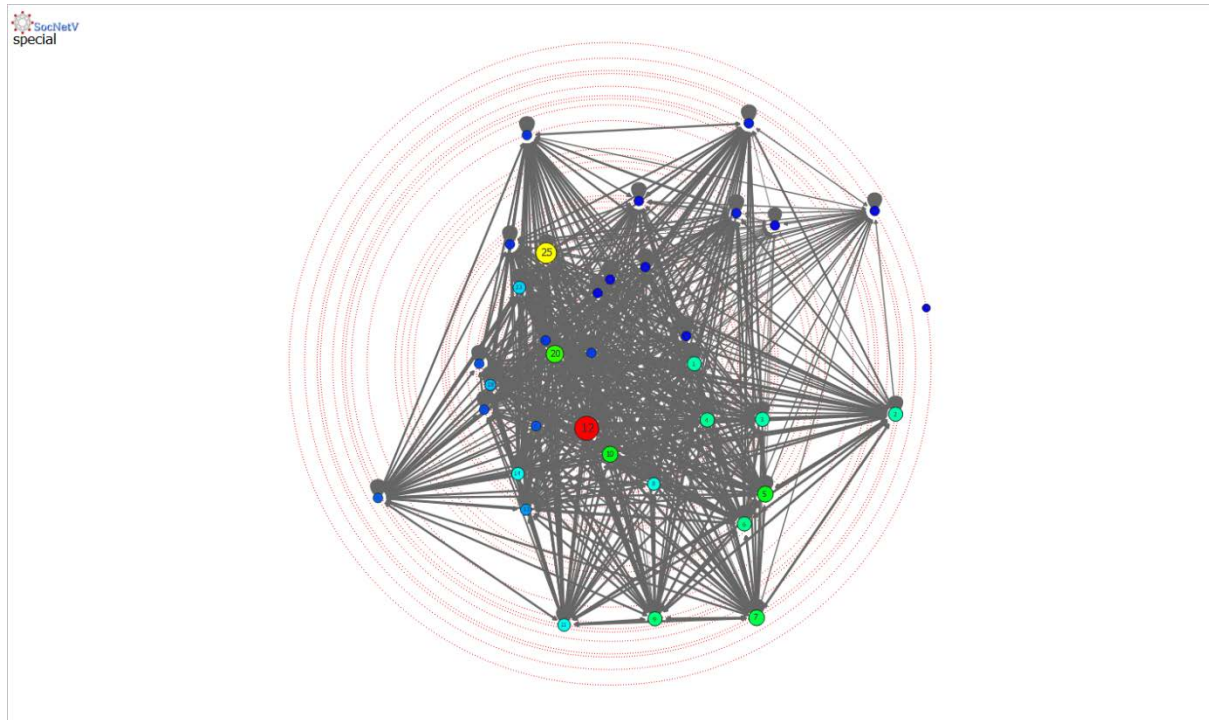


Fig 10: A different visualization. Colours symbolize the weight of the edges.

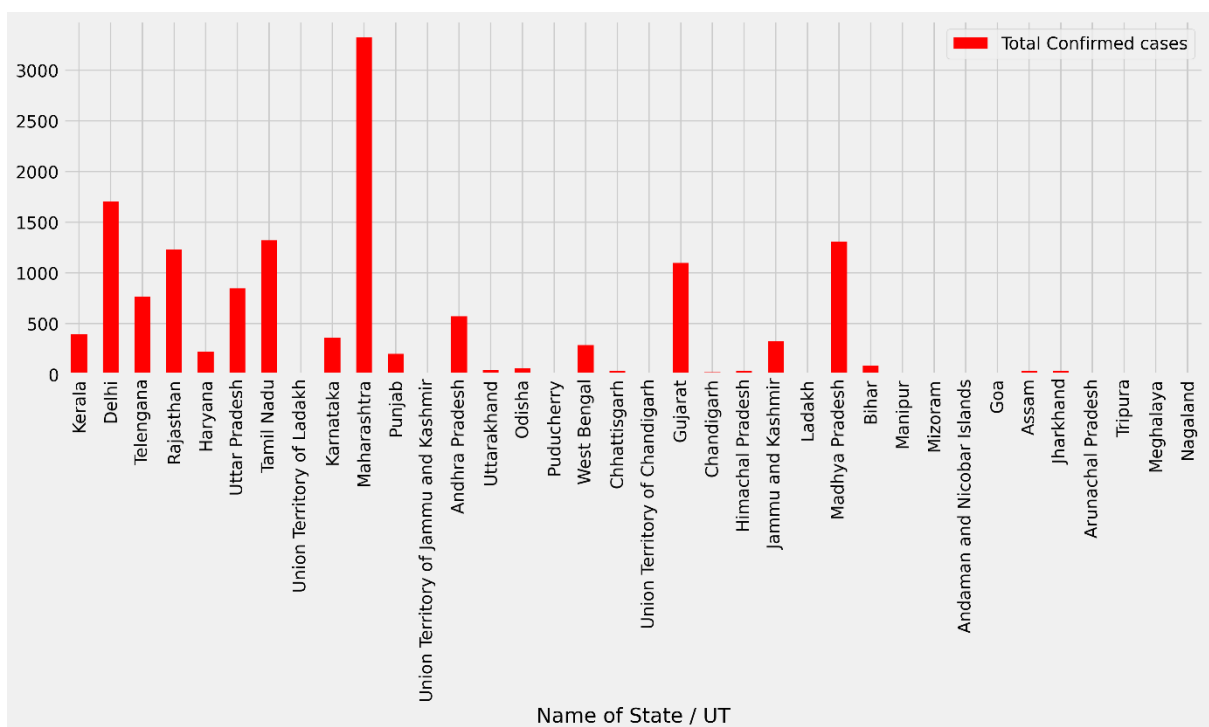


Fig 11: Latest data on the spread of COVID-19

5.3 CODE DETAILS AND CODE EFFICIENCY

5.3.1 CODE DETAILS

PYTHON FILES

data_filter.py

```
import pandas as pd
import numpy as np
import pandas as pd
covid_data = pd.read_csv('complete.csv')
df = pd.DataFrame(covid_data)
states = df['Name of State / UT'].unique()
np.savetxt("states.csv", [states], delimiter=',', fmt='%s')
month_names = ['jan_1', 'jan_2', 'feb_1', 'feb_2', 'mar_1', 'mar_2', 'apr_1', 'apr_2']
months = list()
months.append(df[(df['Date'] >= '2020-01-01') & (df['Date'] <= '2020-01-15')])
months.append(df[(df['Date'] >= '2020-01-16') & (df['Date'] <= '2020-01-31')])
months.append(df[(df['Date'] >= '2020-02-01') & (df['Date'] <= '2020-02-15')])
months.append(df[(df['Date'] >= '2020-02-16') & (df['Date'] <= '2020-02-29')])
months.append(df[(df['Date'] >= '2020-03-01') & (df['Date'] <= '2020-03-15')])
months.append(df[(df['Date'] >= '2020-03-16') & (df['Date'] <= '2020-03-31')])
months.append(df[(df['Date'] >= '2020-04-01') & (df['Date'] <= '2020-04-15')])
months.append(df[(df['Date'] >= '2020-04-15') & (df['Date'] <= '2020-04-30')])
statewise = pd.DataFrame(columns = df.columns)
for month, name in zip(months, month_names):
    for state in states:
        temp = month[(month['Name of State / UT'] == state)]
        statewise = statewise.append(temp[temp['Total Confirmed cases'] == temp['Total Confirmed cases'].max()])
statewise = statewise.drop_duplicates(subset = "Name of State / UT", keep = 'last')
statewise = statewise[['Name of State / UT', 'Total Confirmed cases']]
statewise.to_csv(str(str(name) + '_output.csv'), index=False)
```

adjacency_create.py

```
import pandas as pd
import numpy as np
states = np.genfromtxt('states.csv', delimiter=',', dtype=str)
month_names = ['jan_1', 'jan_2', 'feb_1', 'feb_2', 'mar_1', 'mar_2', 'apr_1', 'apr_2']
month_inputs = list()
for month in month_names:
    temp = pd.read_csv(month + '_output.csv')
    month_inputs.insert(month_names.index(month), temp)
for month in month_names:
    adj = np.zeros((len(states), len(states)), dtype=int)
    arr = np.array(month_inputs[month_names.index(month)][['Name of State / UT']])
    for elem in arr:
        for i in range(len(states)):
            if elem == states[i]:
                j = int(np.where(states == elem)[0])
                for k in range(j+1):
                    adj[k][j] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed
cases']
                    adj[j][k] = month_inputs[month_names.index(month)].iloc[j]['Total Confirmed
cases']
                break
    np.savetxt(month + "_adjacency.csv", adj, delimiter=',', fmt='%s')
```

5.3.2 CODE EFFECIENCY

The codes work with every dataset available out there. The code is independent of space complexity or time complexity. The codes have been tested with large datasets and works just fine. The Social Network Visualizer can get a little slow if the adjacency matrix taken into consideration is really large. It was seen that the algorithms used in the software were not efficient enough. So, it is well advised to not go for large datasets while plotting graphs on low end systems.

CHAPTER 6: CONCLUSION

6.1 CONCLUSION

The proposed method was successful in giving insights into the spread of the COVID-19 virus over the Indian subcontinent from the period of Jan 1, 2020 till date. The spread was well visualized with the help of graphs which showed us that diffusion is a factor for the spread of such a pandemic into the country. Further improvements can be done to this get more detailed insights into the COVID-19 cases. There is always scope for future work in this over this project.

Here is provided the **GitHub** link to my project, for your understanding of the actual working of the project: https://github.com/RajarsiGit/SNA_project

6.2 FUTURE SCOPE OF THE PROJECT

As mentioned above this project works really well with medium to large size datasets. But any dataset which is extremely large cannot be used because this could slow down the system and software too. Future work may also take into consideration the large data factor and use graph visualization software which are efficient enough to work on large adjacency matrices.

REFERENCES

1. Zhang M. (2010) Social Network Analysis: History, Concepts, and Research. In: Furht B. (eds) Handbook of Social Network Technologies and Applications. Springer, Boston, MA
2. <https://www.medicinenet.com/script/main/art.asp?articlekey=22789>
3. <https://emedicine.medscape.com/article/2500114-overview>
4. [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
5. <https://socnetv.org/docs/index.html>
6. <https://journals.openedition.org/economiepublique/pdf/1721>
7. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. David Easley, Dept. of Economics, Cornell University & Jon Kleinberg, Dept. of Computer Science, Cornell University
8. <https://en.wikipedia.org/wiki/Diffusion>
9. <https://en.wikipedia.org/wiki/Kaggle>