CSM Klein 2 Final Report

Team: Anthony Zorovic, Jake LaLuna, James Mach, Simon Kotchou
Client: Professor Judith Klein
6/3/2020

Introduction

Professor Judith Klein is a researcher and Associate Professor in the Department of Chemistry at the Colorado School of Mines. Prof. Klein has her doctorate in Biological Chemistry and has done extensive work in bioinformatics in the past. After the advent of COVID-19 in the US, Klein took an interest in the characteristics of the disease's spread, and what data on said spread was publicly available to use for research. She later approached CSM's Computer Science department for help with aggregating the available data and creating a visualization for the spread of the virus. We were chosen as the field session team to assist her, and this is the task we were originally given:

"A major issue regarding the continued spread of coronavirus despite social distancing and economic restrictions revolves around the breakdown of social distancing in ways that are often not thought about. For example, an infected UPS or Amazon delivery driver could spread the virus to multiple people through contact with packages, or salon professionals could be making house calls to keep their business afloat. Once a person like that tests positive, it can be very difficult to determine who could have had contact with that person in such a manner that transmission is likely to have occurred. Currently, The State of Colorado is manually conducting interviews to try to solve this problem and is attempting to predict how much effort they should put into contacting people in certain regions.

The project goal is to create a graph network of coronavirus transmissions in which nodes represent people and edges represent the transfer of the virus, and then to characterize and analyze the graph to aid in developing a pipeline of finding possible infected people once a person has tested positive. The graph should be useful in identifying areas of concentration that officials can use to allocate resources effectively in dealing with the virus."

During the course of our research, we found that there are severe limitations on the accessibility of the personal data we'd need to meet these requirements. As such, we narrowed down the project scope. Our finalized project goal involved using infection data by county in the US and logistic growth models to measure the most likely infection paths from county-to-county.

Requirements

Professor Klein designed this project to be very open-ended. Our minimum expectation was to lay the groundwork for a network visualization method that future researchers could implement and expand upon.

Our finalized project consisted of these functional requirements:

1. See what datasets on COVID-19's spread throughout populations are publicly available.
2. Use the datasets we found to create a graph, with nodes as populations (individuals, counties, states, etc.) and edges as the spread between them[1]. In our final implementation, we specified that the nodes were the counties in each state of the US.
3. Create a visualization of the graph.
4. Create a simple database/spreadsheet of the sources we collectively looked into for available data

And these non-functional requirements:

1. Find additional population data sources related to the spread of COVID-19 to add characteristics to the graph.
2. Be able to extract US infection data over time from the Johns Hopkins COVID-19 Data Repository[2] and analyze its contents.
3. Be able to create our graph in a reasonable time complexity.
4. Become familiar with Cytoscape and Graphspace for the purpose of creating an intuitive visualization from our data.
5. Characterize each source in our spreadsheet by listing their name, primary use for our purposes, URL, availability of data (publicly available, private, or limited), and a description of the source.

These requirements were easily achieved in a timely manner by our team. As such, with the guidance and consent of our client, we began adding additional functionalities. Mainly, this was in the form of adding additional constraints and characteristics to the nodes and edges of our visualization. This was done to make it more true-to-life, and easily understandable for potential stakeholders that may use it in the future.

System Architecture

We first retrieve data from the Johns Hopkins Covid-19 US time series data set[2]. This csv file is updated daily by Johns Hopkins and contains the daily confirmed cases for each county in the United States. We focused on New York State because it has been hit hardest by Covid-19 and therefore has a breadth of data to work with. From the csv we load every data point into a Pandas[3] data frame and are able to easily filter the data just for New York. The data for each county is then populated into a networkx graph as a node. We then have a few ways that we can choose to make edges among the nodes:

1. The simplest way we achieved this was by giving each county a relative_time based on the first infection confirmed in that county. Then we simply connect them linearly based on this relative_time parameter. This produces a very linear network graph. The graph produced from this is stored in files named ‘first_infected_data.csv’ and ‘first_infected_network.csv’. This function can be found in ‘process_data.py’ under the name ‘create_edges_for_graph_first_infection’.
2. Have 2 parameters: threshold and distance_limit. We use the latitude and longitude of each node pair in our graph to calculate the distance between them. If this distance is greater than our arbitrary distance_limit, then we do not list that node pair as an edge in our graph but, if it is less than it we took the number of infected people in the infected county and divide it by the distance to that county and if it is greater than threshold an edge is added. The graph produced from this is stored in the ‘threshold_distance_data.csv’ and ‘threshold_distance_network.csv’ files. This edge producing function can be run with many different parameter sets to produce many different graphs. This function can be found in ‘process_data.py’ under the name ‘create_edges_for_graph_threshold_distance’.
3. Fit a logistic growth curve to the data of each county using the Scikit-learn library[4], and run a simulation where the first county is the only one infected at time = 0. We use the logistic curves of each county to predict the number of people infected at any given time. The first county will have a radius which is proportional to the number of confirmed infections. This proportion is set by a variable called radius_weight. As this first county infects others (their distance is less than the radius of the infected county) the new county is added to the infected list and also gets an infection radius proportional to the number infected

in that county based on the logistic curves prediction. This continues until a time limit on the simulation is reached which is set arbitrarily. This time limit variable is needed so the simulation is finite. The graph produced from this is stored in the 'logistic_simulation_data.csv' and 'logistic_simulation_network.csv' files. This function can be found in 'process_data.py' under the name 'create_edges_for_graph_logistic_simulation'.

We then take the networkx graph and go through its data to print it to two .csv files. The first .csv file is the data or information about each node. The second .csv file is the network or the edges of the graph.

Afterwards, we take these two files and convert them to a layout called coordinateLayout in Cytoscape which can arrange the nodes based on a coordinate system. The two files must be of the form (nodes file: index, name, group, latitude, longitude; edge file: source, target, weight This is useful for visualization because the nodes will be arranged as if they were on a map. This function can be seen in 'process_data.py' under the name 'convert_to_cyto_layout'.

Once our graph was separated into the 'logistic_simulation_data.csv' and 'logistic_simulation_network.csv' files, we imported them into Cytoscape using the coordinateLayout app. Cytoscape is a graphing application that was originally designed for the use of mapping connections between proteins, genes, etc., but its generalized framework makes it a useful tool for visualizing most network structures. It can rather intelligently interpret .csv files to create easily understandable graph visualizations, which made the process of importing our network and data .csv files very streamlined. Additionally, Cytoscape offers a number of style options for characterizing networks. Users can map elements like node and edge shape, color, labelling, etc. to data in their datasets, which gives them a large level of freedom to produce highly-detailed graphs.

For our purposes, we used the coordinateLayout along with two other style mappings, to give our graphs the characteristics that we wanted. These mappings were:

- The color variation of each edge with respect to distance. The edge-coloring used a spectrum of green to red, with green edges having a short distance between nodes, red edges having a large distance, and yellow edges having an intermediate distance.
- The color variation of each node with respect to the date of the first infection in the state. Red nodes had their first infections appearing soon after the first infection, blue nodes had their first infections well after the first infection, and purple nodes had their first infection appear at an intermediate time

- Usually, Cytoscape will default to a single algorithm that creates a best-fit model for the graph based on node and edge data. However, by using coordinateLayout to import our data and network files, we were able to circumvent that.

An early prototype of our graph visualization using these style mappings is shown in Figure 1. Once we had our graph structure to our client's specifications, we used Cytoscape to convert the graph to a .cys file for submission.

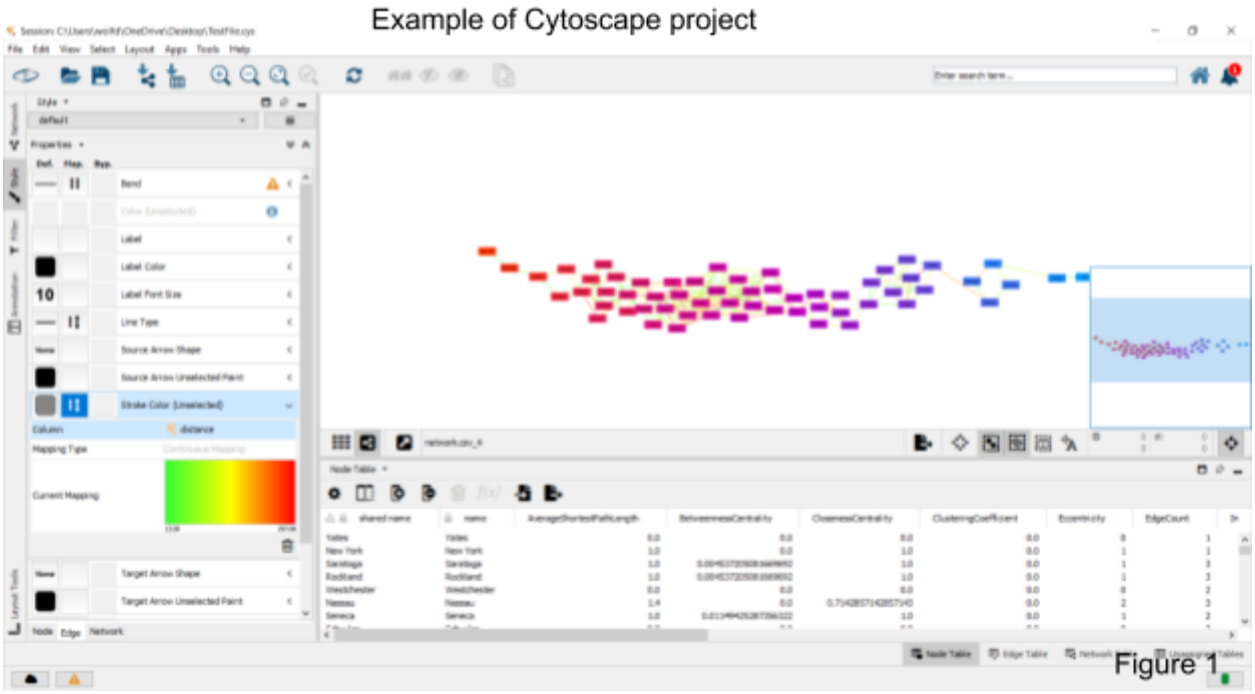

Figure 1

Technical Design

Logistic Curve Regression and Storage:

The logistic curve simulation model depends on three parameters; radius_weight, time_limit, and the logistic curves. Radius_weight, and time_limit are just numbers we provide but the logistic curves are a dict of scikitlearn[4] linear_model.LogicticRegression objects that have been fitted to the data of each county. This allows this simulation of Covid-19 to predict future spread of the virus. Each of these objects or models takes many iterations to create the most accurate model from the data. This makes running this simulation take up to 6 mins for just New York State (62 counties) at 10,000 iterations. At 10,000 iterations most of the models produced a good score but many gave a warning message that it failed to converge meaning that the accuracy would be terrible for that county. The only way to decrease the number of counties that would have poor models was to increase the number of iterations. This was a problem

because the time it could take to generate all these models could get to the scale of hours or days and this would have to be done every time we wanted to change the radius_weight or time_limit to see how it affected the graphs. The solution to this was to find a way to save the logistic curves so that they could be reused since they should be pretty similar or the same every time they are generated. The solution was joblib[5], a library built in python. Joblib. can be used to store scikitlearn[4] objects and their data so that they can be loaded in later by joblib again. This can be seen in our data flow diagram in the figure below. Joblib stores each object as a .pkl file and the size of the New York State logistics curves came out to around 113kB. This would scale up nicely to all the state since the size is not large. One factor we are unsure of is if the size of the

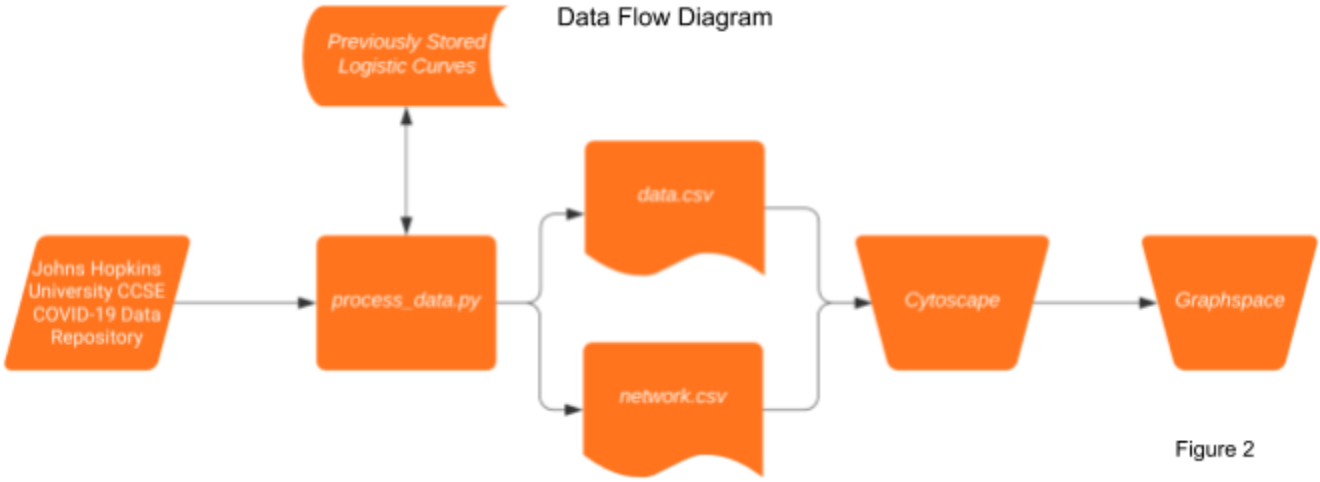

Figure 2

.pkl files get larger as the number of iterations the model performs increases.

coordinateLayout in Cytoscape:

One plugin we found very useful in Cytoscape was the coordinateLayout. It can be installed through the Cytoscape app manager. This layout has strict rules about how the data.csv and network.csv must be formatted. To use this layout we built a simple function to convert our current data and network files to the correct format. These new correctly formatted files can then be fed to this layout and helps visualize spatially where each county is to the others. Below is an example of the same graph but in the two different layouts the first being no defined layout, and second being based on coordinates. You can clearly see New York in figure 4.

Graph with default layout

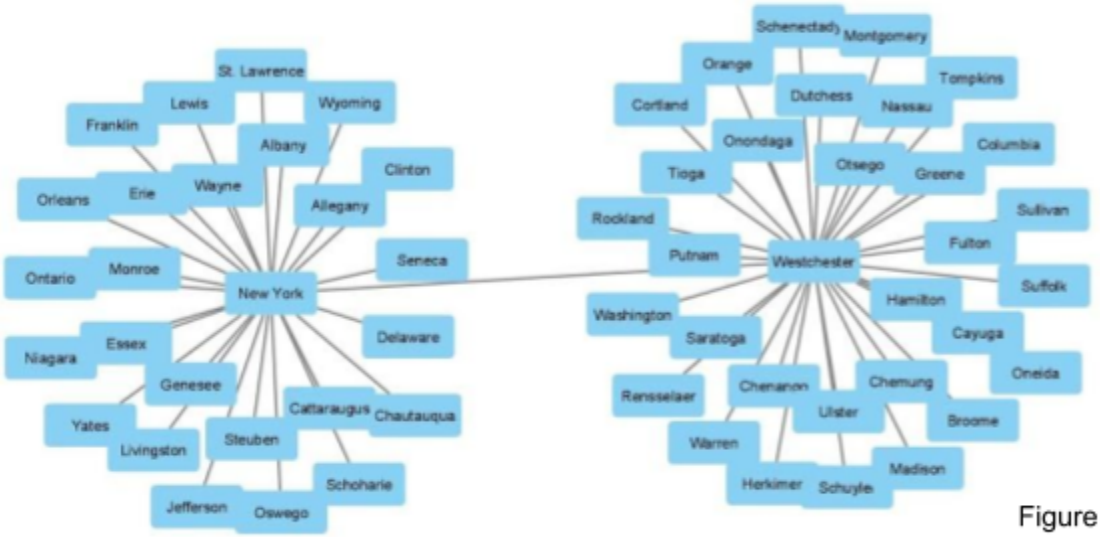

Figure 3

Graph with coordinateLayout

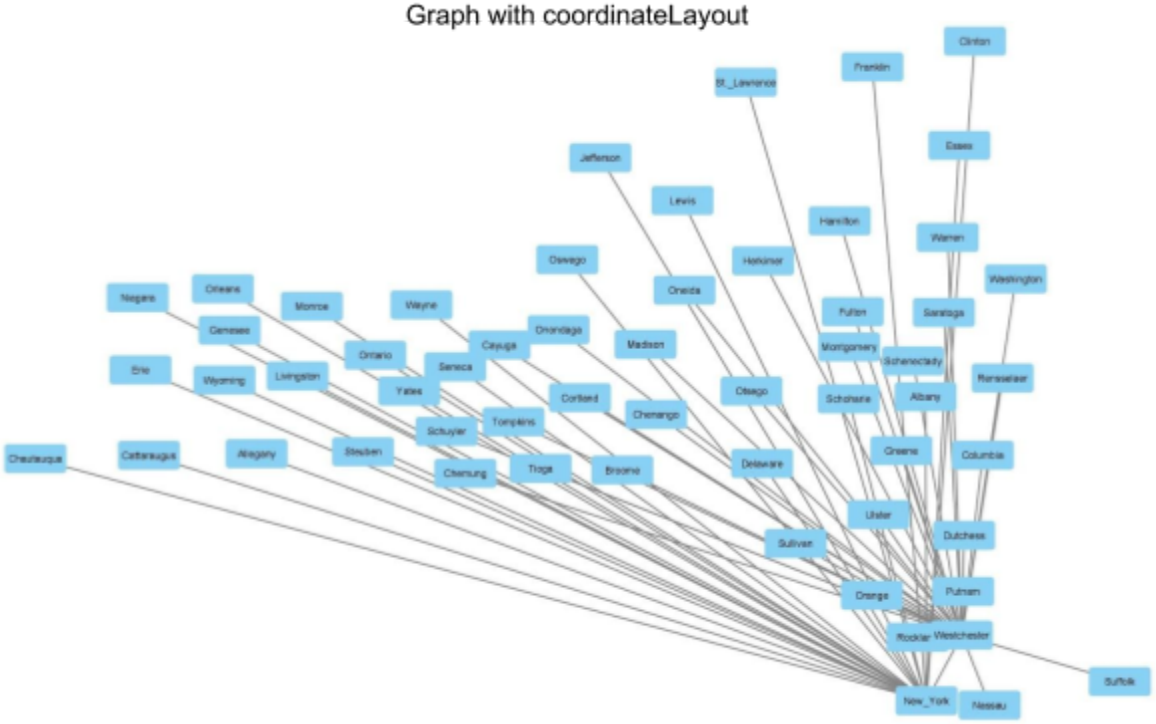

Figure 4

Quality Assurance

Quality assurance was a malleable, ongoing process for our team, and varied from stage-to-stage.

For the first two weeks of our project, our primary goal was data acquisition. Our team started by splitting this task up into subsets of data for each person to research. These subsets primarily took the form of contact tracing[Glossary] data; superspreader event[Glossary] data; climate data, and how climate affects the spread of COVID-19; delivery data to track fomite contamination through delivered packages; Twitter data; and traffic data to identify common travel routes that COVID-19 could take to spread between population centers. To maintain the legitimacy of our research and to protect the privacy of those involved with our datasets, we had to ensure that each dataset we found was obtained from a legitimate source. To do this, we made sure to only reference data that was either linked to scientific articles, trustworthy APIs, and government databases.

After acquiring all the data we could find, we determined that using a scraper[Glossary] on Twitter alongside an archive of every reported superspreader event[6] to create a graph network of hypothetical SSEs would be the most promising course of action. This process would've been streamlined by getting access to Twitter's COVID-19 API. Prior to working on our implementation, we applied for access. Then we began work on a proof-of-concept data scraper that could be used with the API. We used an existing framework from Scrapy[7] to create a spider that went from user-to-user, tracking their followers and following for mutual followers. If a user was a mutual of another user, and one of them had mentioned a collection of SSE-related keywords in a recent tweet, then that connection would be logged as an edge in our graph. However, in reading over Twitter's Terms of Service and robots.txt, we found that this method of data acquisition was strictly against their policy without proper authorization. We hadn't gotten data access approval by Twitter after two weeks of waiting. As such we had to abandon that implementation, since further development would be considered a nonconsensual collection of users' data. The spider we created doesn't follow Twitter's TOS so the account being used to scrape would ultimately be banned at some point by twitter. We only ran this bot for minimal data and have deleted this data as our scope changed.

After abandoning our previous effort, we decided our next best course of action was to visualize data from the Johns Hopkins COVID-19 API[2]. We settled on this dataset for several reasons:

- It was an aggregation of datasets from states and countries, so we didn't have to worry about anonymizing the data of individuals or consider if our data-collection was consensual like we did in our Twitter dataset
- The time-series dataset that we used contained enough detail such that we could create a visualization that was relatively true-to-life; something that would have been difficult with our Twitter data visualization
- It isn't difficult to find additional data sources that are divided by state and county, so it will be easy for future researchers to add onto our framework with characteristics outside of the data we provided,

Once we began developing our visualization, quality assurance was relatively straightforward. Our methodology for implementing our visualization was simple. First we would brainstorm with our client about what attributes we could reasonably implement into our visualization. After that, we researched what external libraries, if any, we could utilize to implement said characteristic; if we couldn't find one that fit our needs, we manually implemented it. Once we had a running program that included the new characteristic, we ran our newly-altered convert.py file on the dataset and exported the formatted data to Cytoscape for visualization. Since our starting dataset was small - we only used the time-series data for New York state - we opted to manually look over the nodes and edges of the graph for inconsistencies and errors. If we found any, we looked back at the portion of our code that likely caused the inconsistency and worked through how to fix it. If there weren't any issues, we would collaboratively refactor our code as necessary.

Results

Features we did not have time to implement:

There are a number of features which we did not have time to implement, as well as a few features we only had time to partially implement. The partially implemented feature revolved around showing where each phase of reopening occurred, and trying to find a correlation between reopening phases and new cases within New York counties. In this case, the group created a CSV with the county names and dates of initial shutdown, phase 1 reopening, and phase 2 reopening where applicable. Unimplemented, but considered or desired features included Twitter data which we never got access to, comparing our infection network with public transportation and highway data, and potentially implementing contact trace or location data to create a network.

Summary of Testing:

Since the process for building the network graph was very linear, we implemented a new feature with data, and by looking at the graph, it was easy to tell if anything was off. As for the logarithmic regression, we utilized an R^2 value to rate the model based on how effective it was, with more data and iterations leading to a higher R^2 value.

Future Work:

This project has lead us to some interesting questions and areas that the client can explore for the future, including applying methods used to different parts of the world/country, drawing connections for methods of spread based on analysis of the graph, as well as creating conclusions on reopening phase effectiveness from the spread of the virus based on regression graphs. Here is a list of ideas we had but never followed and how they may be used:

- Traffic data - lower the average speed, the more people on road, less distancing
- Delivery data - more packages delivered, less people going out.
- Weather data - if spread depends on weather like some studies suggest this could be helpful for larger regions

- Refactor code to handle larger regions - in its current state the process_data.py will break on larger regions. This is due to county names being the same and we think the index or key of the node is limited in size so there will be collisions.
- Create some way to update from John Hopkins data automatically instead of manually.
- Build better system for reading in data to nodes - must be changed as the data being worked with changed, make this step more dynamic

Lessons Learned:

- Data acquisition was by far the most challenging aspect of the project, and this actually fits into data science principles, which state that acquisition, organization, and cleaning of data takes up ~60% of the time.
- Don’t bite off more than you can chew. The project scope kept growing and growing until the point it became unfeasible to complete, and then we had to narrow it down to a dataset we had from the beginning. It felt like a large waste of time, since we didn’t get to use many of the datasets we spent a long time trying to find.
- Pay attention to social media terms of services and ethics of data acquisition
- Try to get the client's full expectations up front- sometimes it felt as if we had to change the project in major ways because it didn’t fit what the client was looking for. Communication is key.

Appendix

Data Cited:

[1] Every dataset we looked for and considered using in our research
https://docs.google.com/spreadsheets/d/1EC9bEwsXeyKuBSXT1GQAjvx3oGPOiwk9P6tk2IZX7lU/edit#gid=0

[2] Johns Hopkins COVID-19 Time Series Data
https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

[3] Pandas Python Library
https://pandas.pydata.org/

[4] ScikitLearn Python Library
https://scikit-learn.org/stable/index.html

[5] Joblib Python Library
https://joblib.readthedocs.io/en/latest/

[6] Superspreader Event Dataset
https://docs.google.com/spreadsheets/d/1nAWy94mS14sJBtSCBgh6GLCWKP_JvAaz0yanvq4kmzs/edit#gid=1025534428

[7] Twitter Scraping Method Used
https://scrapy.org/

Glossary:

Contact Tracing: The process of tracking and monitoring the contact of infected people. Contact tracing tools allow government organizations and private companies to track the people an infected person came into contact with prior to showing symptoms/testing positive for COVID-19. If someone has come into contact with an infected person unknowingly, they will be notified to self-isolate.
Superspreader Event (SSE): An event in which a small number of people infected with COVID-19 spread the virus to a significantly larger than average number of uninfected people.
Twitter Mutuals: If two Twitter users follow each other, they're considered "mutual followers", or just "mutuals" for short.

Setup:

With all the files there is a setup.ps1 file which runs all the pip install commands for all libraries needed for process_data.py. The only thing this does not install is the Cytoscape coordinateLayout plugin which is not needed for process_data.py. This can be installed through the Cytoscape App Manager by searching for it.