CSC111 Project Written Report:

Picturing the Power of Policy in a Pandemic

Jia Hao Choo, Komal Saini

April 16, 2021

Problem Description and Research Question

Since the emergence of COVID-19 in 2019, governments across the world have had to act urgently and make difficult decisions concerning their health security policies. On 11 March 2020, the World Health Organization declared it a global pandemic, posing unprecedented challenges to not only governments and authority figures, but the entirety of humanity (Mintrom). In the past year, the virus has relentlessly travelled within and across countries, leading to tens of thousands of deaths, along with hundreds of thousands of people facing severe sickness through contracting COVID-19. From imposing strict social isolation rules to mask mandates, the policies that governments have enacted to curtail the spread of the virus have imposed massive costs to businesses, economies, and the livelihoods of humanity. Such policies have stirred controversy across the globe regarding finding an equilibrium between protecting public health and supporting the economy, while reducing inequity. As the public's response to policies is equally as significant as governance when it comes to reducing the transmission of COVID-19, the differences in the approaches that countries have undertaken on a global scale motivated us to understand which strategy is most optimal. As a result, the question that we are interested in investigating is as follows: **How can we compare different policy responses to the number of cases and deaths in the respective countries to recommend an optimal approach for future pandemics?** We are interested in analyzing the efficacy of global policy responses concerning stay-at-home restrictions, school and workplace closures, cancellation of public events and gatherings, public information campaigns, face coverings, testing and contact tracing, and vaccination policies in reducing deaths and cases, and administering vaccines (Ritchie). In doing so, we hope to learn more about the interconnections between policies and their role in reducing transmission of COVID-19, and shed light on the policy response(s) that have been most effective in their respective regions. We recognize and acknowledge that we cannot implement blanket solutions and expect them to work across the world; countries are diverse in cultures, economies, and societies as a whole. As a result, we hope our project plays a unique role in capturing both the power of policymaking and the limitations it possesses in the COVID-19 pandemic—insights that can be transferable and instrumental in tackling future pandemics. We understand that all pandemics are not the same, and the applicability of solutions will fluctuate along with a variety of factors, ranging from economic conditions to societal norms, As a result, what may be effective in tackling COVID-19 in today's world may not be necessarily effective in tackling another pandemic ten or twenty years from now. Reflecting on what worked and what did not is nonetheless crucial in both preparing for, preventing, and tackling future pandemics.

Dataset Description

We will be using 9 datasets:

1. main_dataset.csv (source: https://ourworldindata.org/covid-vaccinations)
   This dataset has been slightly modified from the original source, and contains information like the country's name, data, new cases count, new deaths count and population. Each entry represents the data for one day, and the interval between each entry for the same country is also one day.

iso_code	location	date	new_cases	new_deaths	population
AFG	Afghanistan	2020-03-22	4.0	1.0	38928341.0
AFG	Afghanistan	2020-03-23	7.0	0.0	38928341.0
AFG	Afghanistan	2020-03-24	2.0	0.0	38928341.0
AFG	Afghanistan	2020-03-25	33.0	1.0	38928341.0
AFG	Afghanistan	2020-03-26	4.0	1.0	38928341.0

2. stay_at_home.csv (source: https://ourworldindata.org/covid-stay-home-restrictions)
   This dataset contains information like the country's name, country code, data, and the level of stay at home requirements (Refer to Appendix A). Similarly, each entry represents the data for one day, and the interval between each entry for the same country is also one day.

Entity	Code	Date	stay_home_requirements
Afghanistan	AFG	2020-01-01	0
Afghanistan	AFG	2020-01-02	0
Afghanistan	AFG	2020-01-03	0
Afghanistan	AFG	2020-01-04	0

3. school_workplace_closures.csv (source: https://ourworldindata.org/covid-school-workplace-closures)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix B for level descriptors.

4. public_events_cancellation.csv (source: https://ourworldindata.org/covid-cancel-public-events)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix C for level descriptors.

5. public_campaigns_covid.csv (source: https://ourworldindata.org/covid-public-information-campaigns)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix D for level descriptors.

6. face_covering_policies.csv (source: https://ourworldindata.org/covid-face-coverings)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix E for level descriptors.

7. vaccinations_policy.csv (source: https://ourworldindata.org/covid-vaccination-policy)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix F for level descriptors.

8. testing_policy.csv (source: https://ourworldindata.org/covid-testing-contact-tracing)
   This dataset contains similar information as stay_at_home.csv. Refer to Appendix G for level descriptors.

9. centroids.csv (source: http://worldmap.harvard.edu/data/geonode:country_centroids_az8
   This dataset contains the name of the country as well as the longtitude and latitude of the country on the world map.

name	Longitude	Latitude
Aruba	-69.98267711	12.52088038
Afghanistan	66.00473366	33.83523073
Angola	17.53736768	-12.29336054
Anguilla	-63.06498927	18.2239595

Note that we have also created smaller test datasets for every dataset above (except centroids.csv) for quicker runtime and testing purposes. These datasets have the same information as the real datasets, but the information in these test datasets is not accurate as we created them using random data. The test datasets are all csv files that have a filename that starts with "test-".

Computational Overview

Creating Classes and Generating WeightedGraph:
This part is done in classes module and init_graph module. We first created a _WeightedVertex private class with the following instance attributes:

- country_name: str
- new_cases: list[float]
- new_deaths: list[float]
- population: int
- restrictions_level: dict[str, Union[int, str]]
- similar_policies: dict[_WeightedVertex, Union[int, float]]

For similar_policies attribute, it represents the neighbours the country has (the edges). For our project, the edges are determined by the level of restriction for each policy. To illustrate, if two countries have the same level of restriction for at least one policy, then they will be connected. The instance attribute similar_policies is a mapping keyed by another neighbour vertex, and each key is mapped to the weight of the edge. The weight of the edge is calculated using a method of _WeightedVertex class, called calculate_weight, in which the weight is calculated by dividing (the number of level of policies which are the same for both vertices) by (the total number of policies, which is 7 in our project).

Then, we created a _WeightedGraph class, which has _vertices as its only instance attribute with the type annotation of dict[str, _Vertex]. We have decided to use weighted graph to represent our data as we will be using the graph traversal method later in our project to calculate the average of the number of daily new cases/deaths based on given policy levels. We will use this average for a simulation to predict how the cases count and deaths count will grow with the given range of policies (more explanations in the "Calculating average daily case/death counts" section).

Afterwards, we created functions that initialize the weighted graph based on real world datasets. To do that, we created functions like get_main_data, get_population, get_policy_restrictions to initialize each vertex. In these functions, we also made use of the WeightedGraph methods (which in turn makes use of _WeightedVertex methods) to add vertices and edges to the graph. Initializing the new_cases, new_deaths, population and country_name for each vertex is easy because it is simply setting each attribute to the corresponding values that we read from the csv files. For new_cases and new_deaths, we use a dict accumulation to collect the new cases/deaths count for each

day in the form of a list keyed by the country name. Filtering method is used here such that when a country is already in the dict, the function will append a new value to the corresponding country's list instead of creating a new key-value pair. Then, we loop through each country in the dict and create a vertex for the country in the graph as well as assigning each country's list to the corresponding attribute of their vertex. During this process, we also include filtering methods using if statements to exclude all the missing data that is represented with a empty string.

The tricky part of initializing a graph object is assigning a level to each policy in the restrictions_level dict for each vertex since the policy level changes for each country at different time of the year. To deal with this, we have decided to take an average level for each policy, which we calculate by using the statistics.mean method. To gain a more accurate representation of the level for each policies, we did not just take that average. Rather, we also calculated the ceiling of that average using math.ceiling; if the ceiling $-$ the average < 0.5, then we will use the ceiling of the average as the level. Otherwise, we will use the floor of the average as the level of the policy. This calculation is done in get_policy_restrictions method in init_graph module. In this function, we have also used filtering methods to filter out any missing information from csv files that are represented by an empty string, such that the final return value must be an integer.

In both of these modules, we created two exception class: CountryNotInGraphError exception class, which inherits from Exception as well as a CountryNotFound, which inherits for CountryNotInGraphError. These exception classes raise an error message when the country is not in the graph (CountryNotInGraphError) or when the country cannot be found in the csv files (CountryNotFound). These two exception classes are primarily used for debugging purposes at early stages of initializing graph, when we are testing if all countries that are found in one file can be found in another, as well as in the generated graph. The completed version of the program should not raise these errors, unless the users try to do any illegal function calls on random countries that are not in our datasets.

Creating network maps:
At this point, we have initialized a weighted graph representing the real world dataset. We then decided to create a network map for each policy that shows which countries have the same level for that policy. This part of computation is done in plot_networks module. In each map, countries with the same level will be connected (similar to how edges work), countries with different levels will have different colour dots, which are placed in the central location of the country on the world map.

Before plotting, we decided to convert our weighted graph object to a weighted networkx object (NetworkX Developers). Different from how we represent our weighted graph, the networkx objects are cleaner in the sense that each networkx object created only represents a specific level of a policy. This means that every node in the object is connected. However, instead of leaving each node to be connected every other node in the networkx graph, we have also decided to make it so that each node only has maximum 2 edges. So, the edges become a clean chain instead of a really complex web. This is done in order to make the connections on the maps cleaner and easier to understand when they are plotted. This process is done in convert_policy_to_networkx.

To plot these networkx objects on a worldmap, we decided to use plotly.graph_objects.Scattergeo library, which allows us to add markers and lines representing the countries and the connections on a given location on a world map (Plotly). To find the location for each country, we created a function find_centroids_location which reads the centroids.csv and returns the central longitude and latitude data for the specified country. We have also decided to make use of annotations feature of plotly to display a summary statistics for each policy map (Plotly). Particularly, the summary will tell the readers the level of the policy that the colours of the markers represent, how many countries share the same level of policy, and the descriptions of what each level of the policy means.

There are 7 policies used in our project, but we didn't create a function to plot map for all of them. Rather, we only created 4 functions (plot_face_masks, plot_three_levels, plot_four_levels, plot_vaccination) for that, as each policy either has 3, 4, 5 or 6 levels. We simply reuse the same functions for the policies that have the same total number of levels, but we use filtering method through if statements to differentiate which policy the map represents. Note that the users will not be directly interacting with these 4 functions, as we have made a function called visualise() that simply calls the correct versions (again, using if statement filtering) based on the given policy argument. The users will directly be interacting with the visualise() function instead, which we call in the main module.

Calculating average daily case/death counts:
One major goal for our project is to be able to simulate the growth of cases and deaths given a range of different policy and policy levels. To do this, we have to create functions that are able to traverse the graph and find countries with the same policy levels, then take the average of the number of daily new cases/deaths. We will then use that average for our own simulation. This part of the program is completed in the computations module.

The first case of calculating these averages is to find whether there is any country in the graph with the exact same policies level as the ones given. This is done in the function exact_policies. If there is one or more countries with that meet the criteria, calculating the average will be easy. We just find the average of the daily new cases/deaths of each of those country by using statistics.mean on their _WeightedVertex.new_cases or _WeightedVertex.new_deaths attribute. We then divide that average by the number of population of the country which we can also get by accessing the population attribute of the vertex. This will give us the average number of new cases every day in terms of the percentage of a given population. For example, if the returned float is 0.01, then the average daily new cases is 0.01 of a country's population. We will do the same computation on each of the country that meet the criteria, then take the average from all of them. This is the final average that we will be using in our simulation.

However, when there is no country that meets the exact same condition as the given policies levels, we have decided to deal with each policy individually. For each of the policy, we will find one country in the graph that has the same level of policy. This is chosen randomly using the built-in random library. The country will act as a starting point for the traversal. In a similar fashion as above, we will find the average number of new cases/deaths of the country using statistics.mean and divided that average by the number of its population to get the the average number of new cases every day in terms of the percentage of a given population. We then append that average in a accumulating list. Then, we will loop through the neighbour of the countries, and for every neighbour that has the same policy level, we will calculate the average number of new cases every day in terms of the percentage of a given population using similar method. However, we also multiply that number by the weight of the edge to take into account that the countries may not be exactly the same and may only be connected as they only share one similar policy. If the countries are only similar because they only share one same policy level, taking directly that average may not accurately reflect what happens in real life, so we decided to mitigate that by multiplying the average by the weight of each edge. We will then append then final average value into the accumulating list. Then for each of the neighbour, we have used recursion to go through the neighbours of the neighbour, the neighbours of the neighbour of the neighbours and so on, such that the averages of all vertices in the graph with the same level of policy are appended to the accumulating list. This recursion is done in _WeightedVertex.get_neighbour_averages_cases or _WeightedVertex.get_neighbour_averages_deaths methods depending on whether we are looking at the deaths count and cases count. We then take the average of the accumulating list, and use this average as our daily new cases/deaths count in terms of a population in our simulation. This whole process is done in get_new_cases_growth_rate and get_new_deaths_growth_rate, which call the weighted vertex recursive methods mentioned above.

There are also cases where we cannot find a starting vertex, when there is no country with the given policy level. In these cases, we will find two countries with the policy level that is one higher and one below. So if there is no country with the level 4, we will find one country with level 3 and one with level 5. We will then use the same process to calculate averages for the two policy levels, then take the average of that. In situations where the policy level that is one above or below is not available, we will keep going higher or lower until finding one that is available. If there is only one level available and an average cannot be taken, the function will return the average based on that sole level that can be found. This process is done in two private functions: _get_new_cases_special and _get_new_deaths_special, which uses while loop to achieve the result. These two are private functions, but they are called in get_new_cases_growth_rate and get_new_deaths_growth_rate in an if branch when there is no correct starting vertex found.

Note that there is an element of randomness in the process above, and different starting points will yield a different value as the weight of the edges will change depending on the vertices. Therefore, we will do this process 5 - 10 times to get a final average for our simulation. This process is done in get_final_case_average and get_final_death_average, which call get_new_cases_growth_rate and get_new_deaths_growth_rate. The number of times the process is repeated is chosen randomly using random.randint method.

Lastly, as mentioned earlier, this whole process above is done for every given policies, so if there are 7 policies specified for our simulation, we will get 7 average new daily cases/deaths count in terms of the percentage of a specified population number. We will take the average of that 7 values to use in our simulation. This part is done in get_total_average_case_growth and get_total_average_deaths_growth.

Simulation:

To create a simulation, we first need to generate data for simulation given a range of policy levels. From the previous section, we have already create functions that calculate the average daily cases/deaths count in terms of the percentage of a specified population number. We will use that average for our simulation. For our simulation, we decide to set the population number to be 7.8 billion, which is the global population in March 2020, around the time when the pandemic started (Kaneda). We want to see how the number of cases/deaths will grow over a year if every country in the world had simultaneously implemented the same level of policies. To get the average daily number of new cases/deaths for a population of 7.8 billion, we multiply the average that we get using functions from previous section by 7.8 billion. That way, the number of cases/deaths grow by a constant factor (linear growth) over a year. Obviously, linear growth does not capture what happened in the real world perfectly, but to avoid complication and further complexity for our simulation, we have decided to keep it a linear growth.

Regardless, from there, we have managed to generate a simulated dataset from the averages, though the process is not that simple, as we have to use filtering and while loops to make sure the number of population infected/dead is not more than the starting population, but also keep the loop going as long as there are still population left uninfected or not dead. Also, we decide to make it such that only starting day 19 will there be deaths, as the average time to death for COVID-19 is about 18.5 (Knapton). This helps to make our simulation more realistic instead of just having people die on the first day when people just start to get contracted with the virus. Therefore, we have use filtering as well to make sure that the first 18 days there is no death. At this point, we decide to generate data until there is no person left living instead of just constraining in to the first year (365 days). This whole process is done in the function create_predictions.

In the same function, we also decide to transform the data into pandas dataframe once we have generated a complete dataset (which was originally in the form of dict). Transforming the data into pandas dataframe allows

us to access the the number of cumulative cases/deaths at a specific day more easily, and this is going to help us when we are plotting the animation of the graph using plotly (Lynn). Also,it makes sense to transform the generated dataset into a pandas dataframe, because the structure of the data fits perfectly in tabular form like this:

Day	Total Cases	Total Deaths
1	1000	0
2	2000	0
3	3000	0

The next and last step would be to plot the data in the dataframe to a plotly line graph. We first extract the first 365 rows from the dataframe, cause our simulation only runs for one year. Then, we only plot the first row of data on the graph since we also want to create an animation that plots the rest of the data automatically in real time. We create the animation using plotly.graph_objects (Plotly).Frame function, which we use it with for loops to loop through every row in the dataframe and plot the data in real time. As well, we create a button which the users have to click on to start the animation. Lastly, we have also added summary statistics on the graph using the annotations tool of plotly to show the level of policies chosen for the simulation as well as the total number of cases and deaths after one year (Plotly).

Instructions for running the program

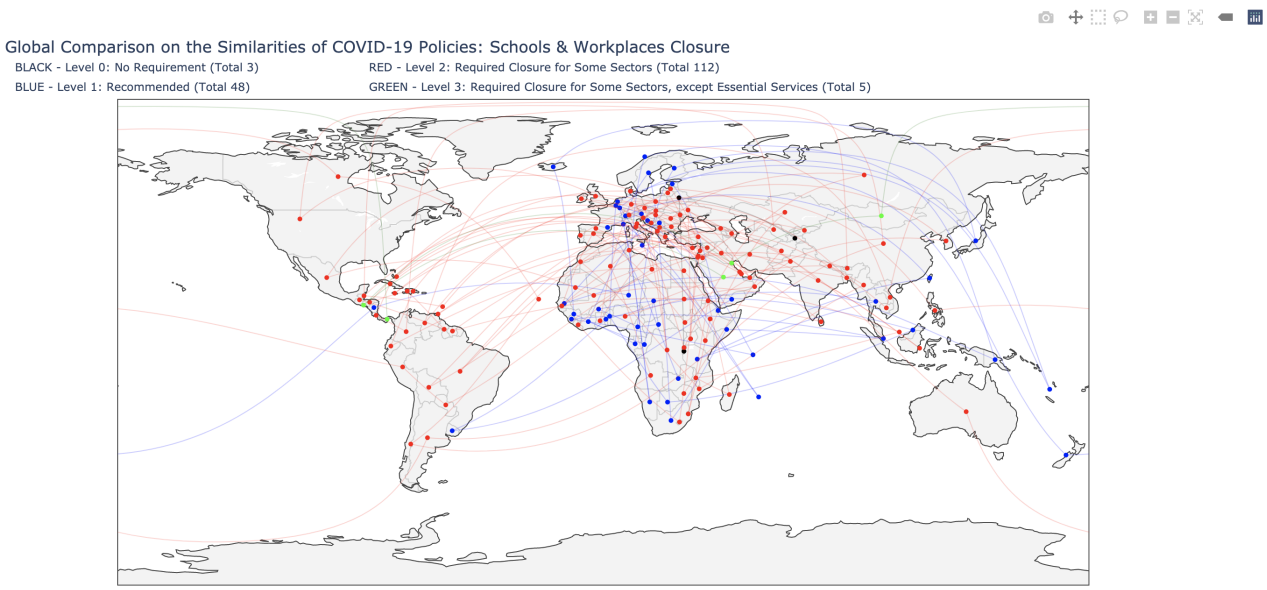
Getting the datasets:
Download datasets zip from UTsend:

- Claim ID: yjkVJtjZGdvTmQAC
- Claim Passcode: yJCZWfKzCAaXiJeb

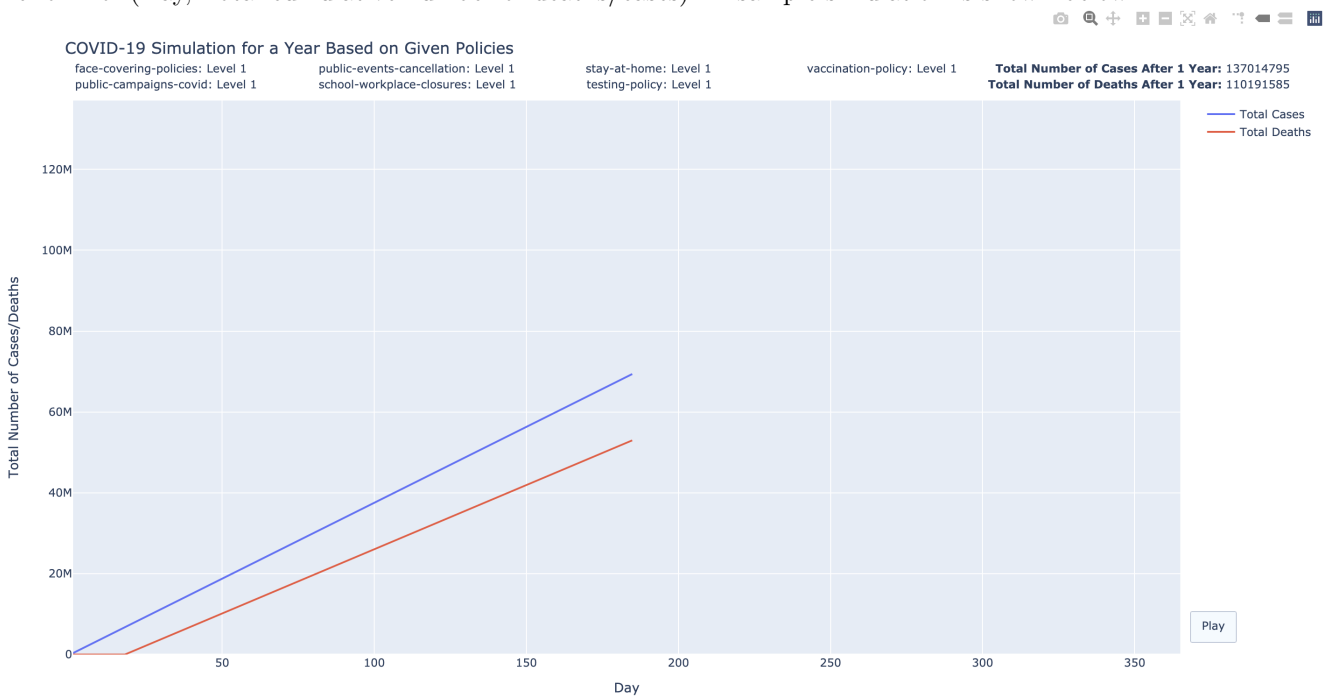
Open the zip and you should get an extracted folder (the folder should call "datasets" and contain 18 files in it). Put the folder directly in the main project folder where all other modules are stored in. Do not move any files out of the extracted folder.

Running the program:
Run main.py to see the visualisations for our program. By default, the real datasets will be used to run the program. Note that because generating a complete graph from the real datasets take a long time, we have use the pickle library to serialise the generated graph state to a file called "saved_graph" in the dataset folder, and we load that graph state from the file directly (Vanderheyden) . If you wish to run a complete version of the program, including generating a graph from scratch, run the test datasets version, which has been commented out at the bottom of the main module.

What you should see:
Once you run main.py, you should see 8 graphs. The first 7 visuals are a map representing the network of each policy. In each map, countries with the same level of policy are linked together and are in the same colour. You can hover over each dot to see what country it represents. Also, a summary statistics is included directly underneath the title that tells users the number of countries that have the respective level of policy as well as what each level represents. You can also zoom in and out of the map using the buttons at the top right corner of the screen, and you can use the mouse to move the map around. A sample map is included below:



The last graph is a simulation graph. The default version of this graph is set to all policies level equal to level 1. You can play around with the plot_simulation function, specifically its dict argument to take out any policy, or change any policy level to see how that affect the simulation. Note that you should only change the dict argument of the function. When the graph first open, there is no line plotted. You will need to click the "Play" button to see the animation, and the number of cases and deaths will rise as more days pass by. In the graph, underneath the title, there is also a summary showing which levels of policy have been chosen for this simulation - this should directly reflect the dict argument inputted into the function. There is also a summary statistics on the right side showing the total number of cases and deaths after one year based on simulation from the policies chosen. Similar to the network maps, feel free to zoom in or out, and you can also hover over the line which will show a tuple of values in the form of (Day, Total cumulative number of deaths/cases). A sample simulation is shown belowl:



Feel free to comment out any functions in the main module if you do not wish to see so many graphs at the same time.

Changes From Proposal

There is not a lot of changes from the proposal to our final project. One big main change that we have decided to make is to switch from representing our data as a simple graph to a weighted graph instead. We have decided to make this change because a weighted graph which allows us to specify a weight between two different countries, making our program more sensitive to the potential differences between two countries. In the computation overview section, we mention that our program calculates average daily new case/death counts out of all countries with the specified policy using recursion by traversing through countries with the given policy. However, this might be unrealistic as the two countries may only be similar in terms of the given policy but have different levels for the rest of the policies. Therefore, by using a weighted graph, we can yield a more realistic representation by considering how similar both countries truly are. We also initially mentioned that we would be analyzing vaccination rates in our research question, but later chose to focus solely on case counts and death counts because introducing a third variable made our data increasingly difficult to deal with, especially since the vaccinations data was relatively new.

We have also decided to include one more csv file: centroids.csv which allows us to obtain the longtitude, latitude information for each country. We were originally planning to get this information using geopandas library, but installing geopandas may be time-consuming and tricky to do as it involves installing other dependencies first (GeoPandas Developers). Also, we have decided to use pandas dataframe for our simulation data, which was not thought about during our proposal phase. Lastly, we switched from deciding to using matplotlib to plotly for all the graphing used in our program, because plotly is simpler and there are fewer line of codes necessary to achieve the world map graphics as well as the animation.

Discussion

The results of our computational exploration partially help answer our research question, "How can we compare different policy responses to the number of cases and deaths in the respective countries to recommend an optimal approach for future pandemics?" Through visualizing each policy using our simulation, we can draw conclusions on their efficacy. Firstly, we will analyze the face covering policy. According to the total number of cases and deaths after 1 year in each of the policy levels, it is clear that level 0 (no policy) resulted in the least number of cases and deaths after 1 year. Though it may initially come off as surprising, this actually makes sense because the countries with the higher policy level (and higher restrictions) are often the countries with the most number of cases and deaths. Level 4 shows the second least number of cases, which indicates that high restriction results in lower case counts. As a result, we can deduce that face covering policies, at the highest restriction, are effective. With the public events cancellation policy, the highest case count and highest death count after 1 year is with the highest restriction (level 2), and lowest is with no restrictions (level 0). This is a great representation of one of the limitations of our project; it paints a picture of "more restrictions results in more cases and deaths" without considering the fact that higher restrictions often only take place when cases and deaths are at an all-time high. A similar trend can be seen in the school and workplace closure policy, public campaigns policy, testing policy, as well as the vaccine policy. This reasoning is also well supported by observation of our network maps, which plot how many countries have the same level for each policy. By observing the maps, we can see that the majority countries have a high restriction level. For example, with face covering policies, a total of 100 out of 168 countries have a restriction level 2 out of 4. This definitely have an impact in our prediction, as most of the countries with a lot of covid-19 cases or deaths like UK and India both fall within this level.

With the stay-at-home policy, the lowest case count after 1 year is in level 3 (highest restriction), indicative of its success in reducing the transmission of COVID-19. We also compare the simulation when we include all our

policies, and set each of their level to their highest and lowest. A similar pattern can be observed as before. The simulation with all maximum level of restrictions result in more deaths/cases over a year than the simulation with all level 0 restrictions. The reasoning to this is the same: many countries with higher level of restrictions are ones with a lot of COVID-19 cases and deaths.

However, we understand that correlation doesn't imply causation. In our analyses, we are merely attempting to make sense of the real-world data we have found, and we cannot actually conclude a cause-and-effect relationship between the variables we have discussed. In the process of attempting to make sense of our findings, we also introduce a variety of assumptions taking into account our current situation with COVID-19 and our personal observations. Though our findings help us draw conclusions in better understanding the efficacy of different policy responses, we cannot truly deduce what is optimal for future pandemics based solely on our project. We do, however, believe this is a good first step in understanding the results of different policy responses, and potentially leveraging this information for tackling future pandemics.For instance, through our project, we understand that countries with higher level of policy often have high number of cases and deaths, and that could probably be due that the countries only implementing the policies when the pandemic is out of control. Therefore, to successfully control the pandemic, perhaps it is the best to start implementing policies as soon as possible when the pandemic starts.

Note that we decide to focus our discussion solely based on a simulation that compare each level of policy, as well as comparing the simulation with the maximum level and minimum levels across all policies. It is, however, possible to combine different level for different policies to see how that would change the simulation in our program. Feel free to play around with the program to see if any interesting trend/pattern shows up!

Conclusion and Further Exploration

There are two main challenges that we faced while completing the project:

1. Figuring out a way to calculate average daily number of new cases/deaths is tricky given that the information is incomplete in the sense that there can be no country with the exact policy level for simulation. However, we solved this by taking averages and using graph traversal (and repeating this process several times, then take the average out of all trials).
2. Figuring out how to animate a line graph on plotly was also pretty challenging though we managed to figure it out in the end. However, right now our graph only contains really basic animation, which can be further improved by including a slider so users can move the time forward and backward.

Though we aimed to answer our research question with the results of our project to the best of our ability, there are a number of ways in which we can improve its accuracy in representing the real world. First and foremost, we can consider the possibility that higher case counts can increase the rate of transmission, resulting in exponential growth instead of linear growth which our project is limited to. As we know, this is the case in many regions where COVID-19 hot-spots are contributing excessively to transmission rates. Additionally, though we have analyzed the COVID-19 case counts and death counts related to the policies in each country, it would be interesting to analyze the changes in case counts and death counts after a country switches their policy. This can also make our simulation more realistic. It would be quite interesting too see if switching to a "better policy" actually results in increased cases and deaths; perhaps even a country with high death and case counts is employing an optimal strategy as per their norms, economy, governmental intervention, etc. This would be fascinating, but not surprising, as the rules that work best differ from country to country—often even city to city. Analyzing a larger variety of policies (ex. More intricate policies such as curbside pickup policies) would help us gain a better idea of what is effective in not only countries, but provinces, cities, regions, and towns.

Works Cited

GeoPandas Developers. "Installation." Installation - GeoPandas 0.9.0 Documentation, geopandas.org/getting_started/install.html.

Kaneda, Toshiko, et al. 2020 World Population Data Sheet Shows Older Populations Growing, Total Fertility Rates Declining. Population Reference Bureau, 13 Apr. 2020, www.prb.org/2020-world-population-data-sheet/.

Knapton, Sarah. "Coronavirus Kills in an Average of 18 Days." The Telegraph, Telegraph Media Group, 12 Mar. 2020, www.telegraph.co.uk/news/2020/03/12/coronavirus-kills-average-185-days/.

Lynn, Shane. "Using Pandas Iloc, Loc, & Ix to Select Rows and Columns in DataFrames." Shane Lynn, www.shanelynn.ie/pandas-iloc-loc-select-rows-and-columns-dataframe/.

Mintrom, Michael, and Ruby O'Connor. "The Importance of Policy Narrative: Effective Government Responses to Covid-19." Taylor and Francis, 4 Sept. 2020, www.tandfonline.com/doi/full/10.1080/25741292.2020.1813358.

NetworkX Developers. "Weighted Graph." Weighted Graph - NetworkX 2.6rc1.dev0 Documentation, 12 Apr. 2021, networkx.org/documentation/latest/auto_examples/drawing/plot_weighted_graph.html.

Our World In Data. "Coronavirus (COVID-19) Vaccinations - Statistics and Research." Our World In Data, 2021, ourworldindata.org/covid-vaccinations.

Our World In Data. "COVID-19: Cancellation of Public Events and Gatherings." Our World in Data, 2021, ourworldindata.org/covid-cancel-public-events.

Our World In Data. "COVID-19: Face Coverings." Our World in Data, 2021, ourworldindata.org/covid-face-coverings.

Our World In Data. "COVID-19: Public Information Campaigns." Our World in Data, 2021, ourworldindata.org/covid-public-information-campaigns.

Our World In Data. "COVID-19: School and Workplace Closures." Our World in Data, 2021, ourworldindata.org/covid-school-workplace-closures.

Our World In Data. "COVID-19: Stay-at-Home Restrictions." Our World in Data, 2021, ourworldindata.org/covid-stay-home-restrictions.

Our World In Data. "COVID-19: Testing and Contact Tracing." Our World in Data, 2021, ourworldindata.org/covid-testing-contact-tracing.

Our World In Data. "COVID-19: Vaccination Policy." Our World in Data, 2021, ourworldindata.org/covid-vaccination-policy.

O'Neill, Kelly. "Harvard WorldMap." Country Centroids - WorldMap, worldmap.harvard.edu/data/geonode:country_centroids_az8.

Plotly. "Intro to Animations." Plotly, plotly.com/python/animations/.

Plotly. "Plotly.graph_objects.Scattergeo." Plotly.graph_objects.Scattergeo - 4.14.3 Documentation, plotly.github.io/plotly.py-docs/generated/plotly.graph_objects.Scattergeo.html.

Plotly. "Text and Annotations." Text and Annotations in Python, plotly.com/python/text-and-annotations/.

Ritchie, Hannah, et al. "Policy Responses to the Coronavirus Pandemic - Statistics and Research." Our World in Data, 2021, ourworldindata.org/policy-responses-covid.

Vanderheyden, Théo. "Python Pickle Tutorial." DataCamp Community, 5 Apr. 2018, www.datacamp.com/community/tutorials/pickle-python-tutorial.

Appendix

Appendix A

There are 4 levels of stay at home order: 0 corresponds to "no requirement", 1 corresponds to "recommended not leaving the house", 2 corresponds to "required to not leave the house, with exception", and 3 corresponds to "required to not leave the house with minimal exceptions".

Appendix B

There are 4 levels of workplace and school closures: 0 corresponds to "no requirement", 1 corresponds to "recommended closure", 2 corresponds to "required closure for some sectors", and 3 corresponds to "required closure for all sectors, except essential services".

Appendix C

There are 3 levels of public events cancellation policy: 0 corresponds to "no requirement", 1 corresponds to "recommended cancelling", 2 corresponds to "required cancelling".

Appendix D

There are 3 levels of public campaigns policy: 0 corresponds to "no COVID-19 campaign", 1 corresponds to "officials urging caution about COVID-19", 2 corresponds to "coordinated public information campaign across traditional and online platforms".

Appendix E

There are 5 levels of face coverings policy: 0 corresponds to "no policy", 1 corresponds to "recommended", 2 corresponds to "required in some public spaces", 3 corresponds to "required in all public spaces", and 4 corresponds to "required outside the home at all times regardless of location or presence of other people".

Appendix F

There are 6 levels of vaccinations policy: 0 corresponds to "no policy", 1 corresponds to "availability for ONE of following: key workers/ clinically vulnerable groups / elderly groups", 2 corresponds to "availability for TWO of following: key workers/ clinically vulnerable groups / elderly groups", 3 corresponds to "availability for ALL of following: key workers/ clinically vulnerable groups / elderly groups", 4 corresponds to "availability for all three plus partial additional availability (select broad groups/ages)", and 5 corresponds to "Universal availability".

Appendix G

There are 4 levels of testing policy: 0 corresponds to "no policy", 1 corresponds to "only those who have symptoms and meet specific criteria", 2 corresponds to "testing of anyone showing COVID-19 symptoms", 3 corresponds to "open public testing".