2021

Pandemic Simulation

PRASHANT KUBSAD & BRIAN SCHUELER

ISYE 6644 | SIMULATION

TABLE OF CONTENTS

ABSTRACT:

This report summarizes simulation of spread of pandemic using 1,000 simulated 'people' to show how reducing the mobility of infected people (quarantine) helps reduce the spread of the disease, leading to better outcomes for the population. Our simulation models the random movement of people within a specified grid size and tracks infections, recoveries, and deaths over the course of 50 days. In our simulation, when simulated individuals come close to each other, they can pass on the infection. We simulate two scenarios, one where infected individuals stop moving (quarantine) and one where infected individuals do not change their behavior and continue moving around after contracting the virus. We've found that quarantine measures in our simulation slow the spread of the pandemic to about ½ to 1/3 the rate of spread when no quarantine measures are in place.

BACKGROUND AND DESCRIPTION OF PROBLEM:

Over the last year and a half, virtually the entire population of the world has become very familiar with the problem posed by a pandemic virus. Even though for many of us, COVID-19 is the first severe pandemic that we have experienced in our lifetimes, it is very likely that we will face another pandemic during our lifetimes.

Like coronaviruses, influenza viruses mutate rapidly, and have caused pandemics in the past. Health experts argue that the emergence of another influenza-based pandemic is not a question of if, but rather of when. Fortunately, the last three pandemics caused by influenza, in 1957, 1968, and 2009, were mild in comparison to the disruption caused by COVID-19. However, this is not necessarily the case. An influenza virus was responsible for the devastating pandemic in 1918, and it is possible that the next pandemic could be just as severe as the one we are currently battling[1].

Modeling the spread of pandemics is complicated, and even some professional models developed by teams of expert epidemiologists failed to accurately anticipate the impact and spread of the disease[2]. Our model will not be able to accurately predict the spread of any one of these pandemics. However, what we can do is model the hypothetical effects of one commonly used approach to mitigate the spread of a pandemic.

We aim to create a model that demonstrates the effects of quarantines on a population facing a pandemic. Our hope is that our simple model will show how reducing mobility of infected individuals can limit the spread and impact of a pandemic.

[1] World Health Organization, '*8 Things to Know About Pandemic Influenza',* 3/11/19. (Link)
[2] Ioannidis, Cripps and Tanner, *'Forecasting for Covid-19 has Failed'* 8/25/2020. (Link)

MAIN FINDINGS:

PROBLEM AT HAND:

Generally influenza is spread when individuals come into close contact with one another. The CDC argues that people who have the flu can spread it to others up to about 6 feet away, how many experts believe that this happens through the spread of droplets. It's also possible for an individual to contract the flu via fomites, or surfaces that have the flu virus on them[3]. However our simulation will not cover fomites, only individual to individual transmission.

To simulate these individual to individual transmissions, our baseline simulation models 1000 individuals over 50 days moving around an area of 500 units. this area at a simple level simulates a public place. Over the 50 days, individuals move around the area in a random walk, getting close to other individuals.

If an individual comes within 6 units of an infected individual, they also become infected. But infections don't last forever. After a period of time individuals either recover from the illness, and are no longer able to spread the disease, or if the virus is severe enough, they may die.

In our model, individuals either recover or die after 14 days since they were infected. During this 14-day period, if individuals come within 6 units of an uninfected individual, they will infect them. In our model we assume a particularly deadly form of influenza, with a case fatality rate of 10%. This means that 10% of infected individuals in our simulation will die of the disease.

SIMULATION AND BUILD:

Initializing the simulation's parameters: To build our simulation, we first imported packages from pandas, numpy, and scipy, as well as a few other packages to help with development and visualization. We then set our variables for the simulation.

Other users can adjust these variables as needed to simulate different conditions. However, we caution users against increasing the number of people simulated by much more than 1,000. As the number of individuals increases, the amount of time to execute our code increases exponentially. This is primarily a result of how we measure the distance of individuals from an infected person.

We also initialize two trackers to help us graph the path of two individuals to illustrate the random walk of individuals in the simulation.

```
import pandas as pd
import numpy as np
from scipy.spatial import distance_matrix
from random import randrange,randint,uniform,random,sample
from timeit import default_timer as timer
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
```

[3] Centers for Disease Control, *"How Flu Spreads"* 8/27/2018 (Link)

```
#Initialize Variables for the simulation
threshold=6
cfr=0.1
quarantine=True
Num_days = 50
num_ppl = 1000
movement = 10
move_sdev = 5
size_min = 0
size_max = 500


#Initialize two trackers to follow two individuals through the simulation
p_zero = {'day':[], 'xpos':[], 'ypos':[]}
p_rand = {'day':[], 'xpos':[], 'ypos':[]}
```

Create our Population: We then initialized a dataframe that produces our population of 1,000 individuals, and their attributes. We set their initial locations randomly, set them all to be not infected, alive, and not immune to the pandemic.

We then start our pandemic by infecting 20 of the individuals. We started with 20 individuals, rather than just one individual, because it is unlikely that a pandemic would be identified before a few people had already contracted the virus.

```
def initialize():

    # various attributes of each person
    per_num = []
    x_pos = []
    y_pos = []
    infected = []
    time_infected = []
    recovered = []
    immune = []
    alive = []

    #for person number, x_pos and y_pos, we set these for each individual

    for i in range(0,num_ppl):
        per_num.append(i)
        x_pos.append(np.random.randint(size_min,size_max))
        y_pos.append(np.random.randint(size_min,size_max))


    # for infected, time_infected, and alive, we set these for all people to
    start with default values
    infected = [0]*num_ppl
    time_infected = [np.NaN]*num_ppl
    alive = [1]*num_ppl
    recovered = [0]*num_ppl
    immune = [0]*num_ppl

    population = pd.DataFrame(list(zip(per_num, x_pos, y_pos, infected,
    time_infected, recovered, immune, alive)),
       columns = ["per_num", "x_pos", "y_pos", "infected",
       "time_infected", "recovered", "immune","alive"])

    rand=[randint(0, num_ppl) for p in range(0, 20)]
    print(rand)
```

```
    population.loc[rand, 'infected'] = 1
    population.loc[rand, 'time_infected'] = 0
    return population
```

Measure Distance Between People: To measure the distance between individuals, and thus, to determine if they've been exposed to an infected individual, we create an initial distance matrix to map the distance from each individual to each infected individual.

```
def initiate_distance_matrix(population):

    # Setting parameters for movement of our simulated people
    #the average person moves X units on the x axis and 10 units in the y axis.
    #creating a log for the movement of patient zero (who we infect) & a random
    person (we'll pick person 25)

    p_zero = {'day':[], 'xpos':[], 'ypos':[]}
    p_rand = {'day':[], 'xpos':[], 'ypos':[]}

    infected_array = population.loc[(population['infected'] == 1),
    'x_pos':'y_pos']

    pop_pos_array = population.loc[:, 'x_pos':'y_pos']

    #Initial distance matrix
    mtx0 = distance_matrix( pop_pos_array, infected_array)
    return mtx0
```

Simulating 50 Days: We then run our simulation by simulating each day out of 50 in the following manner:

Identify New Infections: For each day, we set our new infections value to zero, and then loop through our population to determine if they are within our threshold distance to an infected individual. If so they become infected and we update their attributes.

```
def simulate(population,axs,mtx0):
    #creating a log for the simulation at each day
    results = {'day':[], 'new_infections':[], 'infected':[], 'recovered':[],
'immune':[], 'alive':[]}

    mtx = mtx0

    for day in range(0, Num_days):
        new_infections = 0

#loop through the population to determine status, and then change their
position
        for p in range(0,num_ppl):
            #first determine if they were infected
            if population.loc[p,'infected'] == 0 and
               population.loc[p,'immune'] == 0:
                if(any(mtx[p]<=threshold)):
                    population.loc[p,'infected'] = 1
                    population.loc[p,'time_infected'] = day
                    new_infections += 1
```

Determine if sick people have recovered, or died: We then review each sick individual, and evaluate the number of days that they have been sick. If they number of days is greater than 14, they then either die, or recover from the virus. We determine if they have died using a random

variable. If after 14 days, the random variable they get is less than or equal to the CFR, bad luck for those simulated individuals – they have succumbed to the pandemic.

```
#next, for sick people, determine if they are better or if they have died
            if (population.loc[p,'infected'] == 1 and
                population.loc[p,'alive'] == 1 and
                (day - population.loc[p,'time_infected']) >14):
#determine if they died or got better
                if (np.random.rand() <= cfr):
                    population.loc[p,'alive'] = 0
                    population.loc[p,'infected'] = 0
                else:
                    population.loc[p,'infected'] = 0
                    population.loc[p,'immune'] = 1
                    population.loc[p,'recovered'] = 1
```

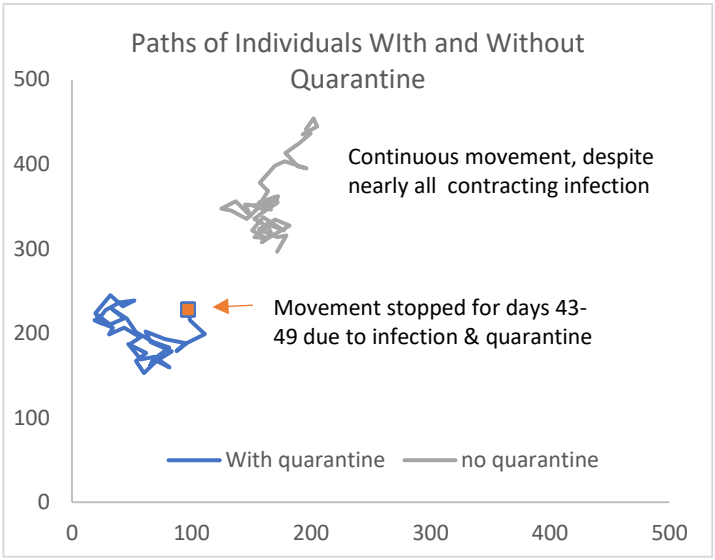
We then move each individual on a random walk, depending on quarantine status: Unless we have set the quarantine attribute to true, each individual will move a randomized amount. This random amount is determined by a normal distribution to determine the X and Y movement amount, and also by a random binomial to determine if they move in a positive, or negative direction.

To prevent our individuals from walking off of our map, we adjust the direction of their movement to prevent them escaping the bounds of our simulation.

```
#next, adjust movement (right now, everyone moves an average of 10 units each
day)
#the first part creates a random variable from a normal dist. the second
determines if it is positive or negative
            x_move = np.random.normal(movement,move_sdev) *
                     (-1 + 2*np.random.binomial(1,0.5))
            y_move = np.random.normal(movement,move_sdev) *
                     (-1 + 2*np.random.binomial(1,0.5))

            if not(quarantine and population.loc[p,'infected'] == 1):
                if ((population.loc[p,'x_pos'] + x_move) < size_max) &
                   ((population.loc[p,'x_pos'] + x_move) > size_min) :

                    population.loc[p,'x_pos'] = population.loc[p,'x_pos'] +
                                                x_move
                else:
                    population.loc[p,'x_pos'] = population.loc[p,'x_pos'] -
                                                x_move

                if ((population.loc[p,'y_pos'] + y_move) < size_max) &
                   ((population.loc[p,'y_pos'] + y_move) > size_min) :

                    population.loc[p,'y_pos'] = population.loc[p,'y_pos'] +
                                                y_move
                else:
                    population.loc[p,'y_pos'] = population.loc[p,'y_pos'] -
                                                y_move
```

Verifying our quarantine approach works: We logged the position of 'individuals in both of our simulations, with and without quarantine rules to determine how our random walk approach works, and if our quarantine rules apply. In the chart below, you can see that our 'individuals' move in a random-walk style. This approach allows individuals to bump into each other and move a bit more realistically than if they changed positions randomly.

We can also see that our quarantine rules work. Our trace with quarantine rules in place shows the tracked person stopping, (quarantining) on day 43, suggesting that our approach works.



New Distance Matrix & Logging Results Finally, at the end of each day, we create a new distance matrix based on the new positions of each individual, and record statistics for the day to track.

```
#updated distance matrix
        mtx = distance_matrix( pop_pos_array_day, infected_array_day)
#update the log each day:
#path of patient zero
        p_zero['day'].append(day)
        p_zero['xpos'].append(population.loc[0,'x_pos'])
        p_zero['ypos'].append(population.loc[0,'y_pos'])

#path of a random person (25)
        p_rand['day'].append(day)
        p_rand['xpos'].append(population.loc[25,'x_pos'])
        p_rand['ypos'].append(population.loc[25,'y_pos'])

#overall results

        tot_infected = sum(population['infected'] == 1)
        tot_recovered = sum(population['recovered'] == 1)
        tot_immune = sum(population['immune'] == 1)
        tot_alive = sum(population['alive'] == 1)
        results['day'].append(day)
        results['new_infections'].append(new_infections)
        results['infected'].append(tot_infected)
        results['recovered'].append(tot_recovered)
        results['immune'].append(tot_immune)
        results['alive'].append(tot_alive)
```

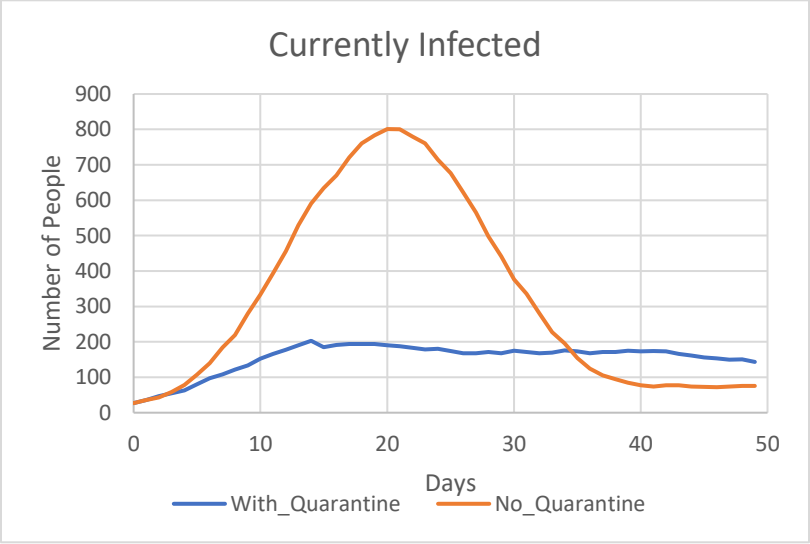
Running the Simulation: After defining our functions above to create a population, an initial distance matrix, and the simulation, we ran two simulations. The first, with quarantine as a default value, and the second with no quarantine efforts.

```
pop1=initialize()
mtx0=initiate_distance_matrix(pop1)
simulate(pop1,axs[0],mtx0)

quarantine=False
pop2=initialize()
mtx0=initiate_distance_matrix(pop2)
simulate(pop2,axs[1],mtx0)
```

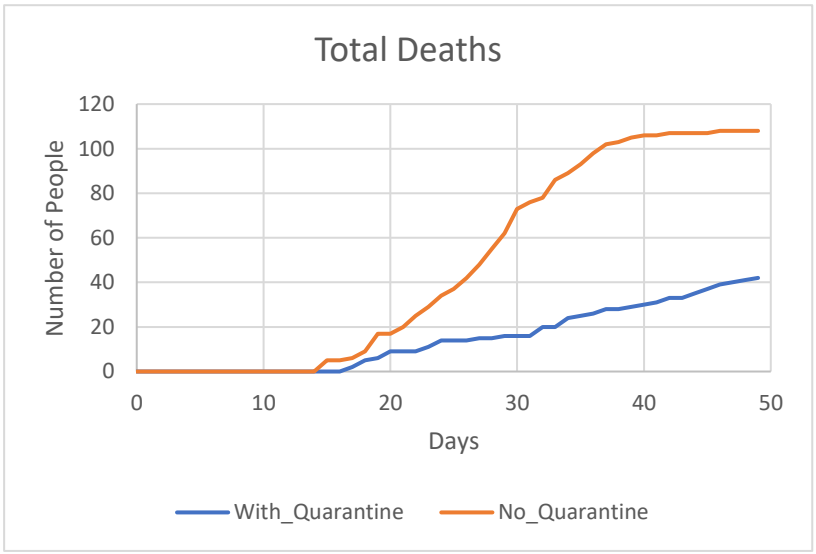
RESULTS AND ANALYSIS:

Quarantine Rules Slow the Spread of the Pandemic: Simply by having infected individuals stop in place, the spread of the pandemic slows remarkably. By day 50 of our simulation 537 of our people in the simulation with quarantine rules contracted the virus. This is not great, but much better than the simulation with no quarantine rules, where every person was infected.

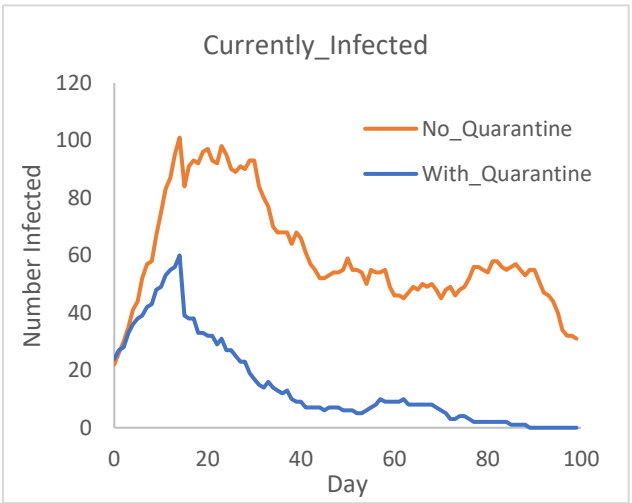
While our simulation doesn't consider the impacts on hospitals and medical facilities, the rapid surge that we see in our non-quarantine scenario would very likely overwhelm hospital facilities. In doing so, it would likely also cause more deaths. By contrast the slower and steadier infection rate in our quarantine example is reminiscent of calls at the beginning of the COVID-19 pandemic to flatten the curve.

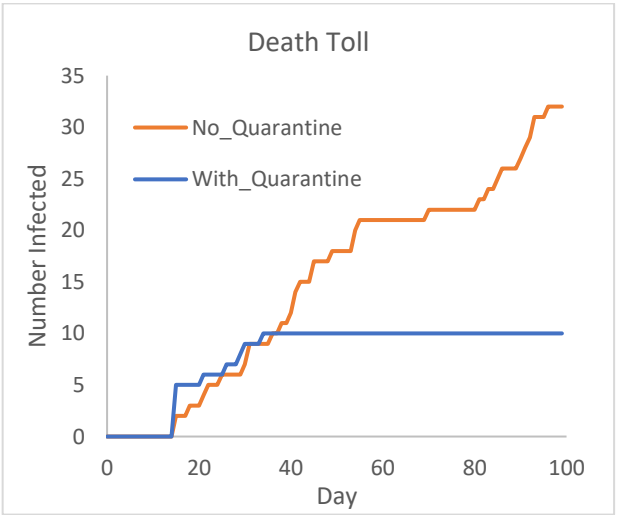


Slowing the Spread of the Virus also Slows Death rates: In our simulation that included a quarantine, 42 individuals 'died' from the pandemic flu, about half of the 108 who died in the simulation with no quarantine actions taken. This is roughly proportional to the total number infected in each scenario, and what we would expect given our simplistic method to determine if someone 'dies' from the virus. What we can see is that with quarantine in place, the number of deaths increase much more slowly compared with the scenario with no quarantine rules. In some simulations these numbers would converge in the long run, but not all. Depending on the parameters of the simulation, the quarantine measures can slow the infection's spread enough to kill it off, and prevent the virus from infecting the whole population.



Killing off the virus with Quarantines (and a bit more space): While quarantining in our baseline simulation only slowed the spread of the virus rather than stopping it entirely, if we increase the size of our 'map', quarantining can eliminate the spread of the virus entirely. When we increase the size of our map, it makes it less likely that individuals will bump into each other, or encounter a stationary, quarantining individual, infecting others. When we increased the size of our simulation to 900 from 500, increasing the area of our simulation by a factor of about 3 1/4 , and increased the length of the simulation, we found that the quarantine measures could fully eliminate the disease. Increasing the size of the map slowed transmission in the no-quarantine scenario too, but not enough to fully eradicate the virus. As a result, infections continued, and the death toll continued to increase.





Conclusions: Our simple simulation helps demonstrate one of the fundamental ideas of epidemiology, that quarantining infected individuals can help reduce the spread of an infectious disease. With more space to move around in, (and lower chances of transmission), quarantines proved more effective. Even in our more cramped base case, quarantines still slowed transmission of the virus.

This model provides more of a theoretical result than a real-world one, but with improvements, it could be used to provide a more realistic simulation. Adjusting infection probabilities to match observed values, shifting the behavior of simulated people to reflect normal human movement patterns like commuting, and allowing non-infected people to adjust their behavior (e.g., quarantine, or wear masks), are a few ways we could improve the model to be more applicable to the real world.

CODE AND EXECUTION:

All the code is checked into the project deliverables. It is a python notebook package and can be run using jupyter notebook. The code is also available at Prashant's github repository and can be accessed from:

https://github.com/Prazhant/pandemic_simulation/blob/main/Baseline_Pandemic_Simulation.ipynb

REFERENCES & SOURCES:

Our approach to modeling a pandemic was inspired partially by Harry Steven's graphics and article in the Washington Post, "Why Outbreaks like coronavirus spread exponentially, and how to 'flatten the curve'". Our simulation uses a similar approach, but with far more simulated individuals, adjusted behavior for individual elements, and adjustable parameters.
https://www.washingtonpost.com/graphics/2020/world/corona-simulator/

Other references & Sources:

World Health Organization, '*8 Things to Know About Pandemic Influenza',* 3/11/19. (Link)
Ioannidis, Cripps and Tanner, *'Forecasting for Covid-19 has Failed'* 8/25/2020. (Link)
Centers for Disease Control, *"How Flu Spreads"* 8/27/2018 (Link)