Infectious Disease Simulation

Cassandre Korvink-Kucinski EID: ck27244

December 2020

1 Introduction

Infectious diseases are diseases generated by pathogenic micro-organisms (i.e. viruses, bacteria, fungi, or parasites) and spread rapidly through a population from person to person both directly and indirectly. Examples of infectious diseases include Coronavirus (2019-nCoV), Ebola, Yellow Fever, and Smallpox, to name a few. These diseases have high fatality rates especially for the weak portions of the populations including the elderly, the very young, and those with pre-existing medical conditions.

The following model analyzes the Coronavirus (2019-nCoV), also called COVID-19, infectious disease. It is caused by a new coronavirus called SARS-CoV-2 and was first recognized by the World Health Organization (WHO) at the end of December 2019, originating in the People's Republic of China. COVID-19 can impact people of all ages with the most common symptoms similar to those of the common cold, but more severe symptoms such as shortness of breath, chest pain, loss of speech and movement, and may lead to death. Those most at risk for COVID-19 are people aged 60+ years and people with pre-existing medical problems including high blood pressure, heart and lung problems, diabetes, obesity, and cancer.

COVID-19 primarily spreads via respiratory droplets and has a higher rate of infection in close contact transmission compared to transmission over long distances from person to person. The problem with the COVID-19 outbreak stems from the young adult population that experiences less symptoms. These young adults can still be carriers of the disease, and spread it to the weaker population, even if they do not have active symptoms of COVID-19. To prevent the high rate of infection spreading through populations, countries have enforced social distancing to limit close contact of people with others outside of their household or immediate family. This has proven to be an effective method to limit a dramatic increase in cases in countries where citizens have abidden to the social distancing guidelines or where there is government enforcement of social distancing practices and stay-at-home orders.

In the simulation, the human population is modeled in the form of a 'contact network' model. In a contact network, each individual in the population is a node connected to another individual through disease-causing interactions. The following model will place limits on individuals in terms of their number of interactions, with whom they interact, possibility of spreading the disease, and the possibility of getting infected themselves.

The following steps were taken to obtain the final model:

(a) Model the infection of a single person and track their state of infection for the number days of infection.

(b) Model the initial infection of a random person within a vector and track that specific person's state through all days of infection until recovered.

(c) Model the initial randomly infected person, with a 100 percent infection rate, infecting their direct neighbors.

(d) Model the initial randomly infected person, infecting direct neighbors based on the probability of infection from a user input infection rate.

(e) Model a percentage of the population as vaccinated before initially and randomly infecting a person, and then infecting the person's neighbors from the the probability of infection.

(f) Model the number of random interactions the initially infected person has within the population, without direct infection of neighbors.

The results of the final model will output the number of infected, the number of susceptible, the number of recovered, and the number of vaccinated. Also included in the outputs will be the number of steps, or days, it took for the entire population to be recovered, vaccinated, or susceptible, but not infected.

2 Code

For the following code and outputs, the interpretation of data is as follows:

- 0 = susceptible: healthy, has not been infected; initial state of all people in population
- -1 = infected: sick for days of infection, can spread infection
- 1 = recovered: healthy, has been infected, cannot get infected again, cannot spread infection
- 2 = vaccinated: healthy, can never get sick, cannot spread infection
- 5 = days of infection
- -2 = day 2 of being sick, -3 = day 3 of being sick, -4 = day 4 of being sick, -5 = day 5 of being sick
- 6 = number of random people in contact (spreading model)

(a) SickPerson Class

The SickPerson class models the infection of a single person and tracks their state across the days of infection. The class has methods `status_string()` to return the status of the person, update() to update the days of being sick, infect() to infect the person and print their status for days of being sick, and `is_stable()` to return true if the person has recovered.

The main function has a loop that updates the overall day number. In the loop, a random number is generated as a decimal between 0 and 1. If that decimal is greater than 0.95 then the person will be infected and given a value of '-1'. So if the person hits the 5 percent of bad luck, then they will be infected for the days of infection. The infect method is called with an input of the number of days of infection. A print statement within the infect() prints the day number for each day and the status of the person on that day. Once the person has been infected for the days of infection, the person will have their value changed to '1' and the for loop will break using the `is_stable()` method as the person has recovered from the infection. The last line printed will be for the day of recovery.

(b) Initial Population Class

The Population class models a single person within a vector, of the size of the population input, randomly getting infected and tracking the overall days.

The class includes the methods `random_infection()` to randomly infect a person with in the population vector, `count_infected()` to count the number of infected, and update() which calls on the SickPerson update method.

The class creates a condition vector of the population size to track the states of each person in the population.

The main function reads in user input of the population size 'npeople' and declares and object population of 'npeople'. The `population.random_infection()` is then called.

(i) `count_infected()`

This method is used to calculate the number of infected individuals on a specific day and is used often. For every position across the vector, if the value at that position is less than zero, then the person is infected and the counter gets updated.

```
int num_infect++ = 0;
for (int j = 0; j < pop_size; j++) {
        if (condition.at(j) < 0) {
                num_infect++;
        }
}
return num_infect++;
```

(ii) `population.random_infection()`

The following is a snippet of choosing a random person within the array. The random generator calculates a decimal between 0 and 1 and multiplies it by the population size. The random person's value can then be between 0 and the population size, and using the round function, the random person's value is now an integer. The issue at this point is that the array begins numbering at 0 until (population size - 1), so there is no position at population size. If the random number is equal to the population size, one is subtracted from the population size to fit the memory of the vector. Afterwards, the person in the vector is tested for values not equal to '-1', so not infected, and changes that person to '-1', infecting the person.

```
srand(time(NULL));
ran_person = (((double)rand() / (RAND_MAX))*pop_size);
if(ran_person == pop_size){
        ran_person--;
}
```

(c) Infecting Neighbors

The next part was to model the initial randomly infected person, with a 100 percent infection rate, infecting their direct neighbors.

(i) For the first day, choose only one random person to infect and print out the vector for step 1 with one infection

```
if (day_num == 1) {
    srand(time(NULL));
    ran_person = (((double)rand() / (RAND_MAX))*pop_size);
    if(ran_person == pop_size){
            ran_person--;
    }
    condition.at(ran_person) = -1;
    cout << "step " << update() << " has " << count_infected() << " infected   ";
    for (int pn = 0; pn < pop_size; pn++) {
            cout << condition.at(pn) << "  ";
    }
    cout << endl;
}
```

(ii) After the first day, change the value of the infection to match the number of sick days if the person, k, is infected

```
for (int k = 0; k < pop_size; k++) {
        if (condition.at(k) == 0) {}
        else if (condition.at(k) < 0) {
                if (condition.at(k) == -5) {
                        condition.at(k) = 1;
                }
                else {
                        condition.at(k)--;
                }
        }
        else if (condition.at(k) == 1) {}
}
```

(iii) After the first day, infect with 100 percent probability to the left and 100 percent probability to the right

```
for (int k = 0; k < pop_size; k++) {
    if (condition.at(k) == -2) {
            p1 = k - 1;
            p2 = k + 1;
            if (p1 >= 0){
                    if (condition.at(p1) == 0){
                            condition.at(p1) = -1;
                    }
            }
            if (p2 <= (pop_size -1)){
                    if (condition.at(p2) == 0) {
                            condition.at(p2) = -1;
                    }
            }
        }
}
```

(iv) Print Vector at Day Number

```
for (int k = 0; k < pop_size; k++) {
    if (condition.at(k) == 0) {
            cout << " " << condition.at(k) << " ";
    }
    else if (condition.at(k) < 0) {
            cout << condition.at(k) << " ";
    }
    else if (condition.at(k) == 1) {
            cout << " " << condition.at(k) << " ";
    }
}
```

(d) User Input of Transfer Infection Rate
(i) Probability of transfer method using the user's input

```
void set_probability_of_transfer(double x) {
    probs = x;
    if (condition.at(ran_person) != -1) {
            random_infection();
    }
};
```

(ii) Randomly can infect or not infect the neighbors to the left and to the right

```
double chance_inf;
if (p1 >= 0) {
    if (condition.at(p1) == 0) {
            chance_inf = (rand() / double(RAND_MAX));
            if (chance_inf < probs) {
                    condition.at(p1) = -1;
            }
    }
}
if (p2 <= (pop_size - 1)) {
    if (condition.at(p2) == 0) {
            chance_inf = (rand() / double(RAND_MAX));
```

```
            if (chance_inf < probs) {
                    condition.at(p2) = -1;
            }
        }
}
```

(e) Vaccinating Percentage of Population

Before the first infection, the percentage of random people the user wants to vaccinate will be calculated of the population into an integer. A random generator will randomly pick that many people out of the population and vaccinate them. The vaccinated people in the vector will have a value of '2'. If the person has already been given a value of '2', then a new person will randomly be chosen.

(i) Probability method update

```
probs = x;
int vac_people = round(vac * pop_size);
for (int v = 1; v <= vac_people; v++) {
    int y;
    do{
    y = round((rand() / double(RAND_MAX)) * pop_size);
            if (y == pop_size) {
            y--;
            }
    } while (condition.at(y) == 2);
    condition.at(y) = 2;
}
```

(f) Global Random Infection

Randomly infect six other people per infected person on the previous day. A vector meet is created for individual k values to track who has met with whom on each day.

```
for (int k = 0; k < pop_size; k++) {
    double chance_inf;
    int b;
    for (int j = 0; j < 6; j++) {   // loading vector meet for six people
            b = round((rand() / double(RAND_MAX)) * pop_size); // person
            if (b == pop_size) {
                b--;
            }
            for (int q = 0; q < meet.size(); q++) {  // six positions for vector meet
                    while ((b == meet.at(q)) || (b == k)) {
                            b = round((rand() / double(RAND_MAX)) * pop_size);
                            if (b == pop_size) {
                                    b--;
                            }
                            q = 0;
                    }
            }
            meet.at(j) = b;
    }
    for (int m = 0; m < 6; m++) {
            b = meet.at(m);
            if ((condition.at(b) <= -2) && (condition.at(k) == 0)) {
                    chance_inf = (rand() / double(RAND_MAX));
                    if (chance_inf < probs) {
                            condition.at(k) = -1;
                    }
            }
            else if ((condition.at(k) <= -2) && (condition.at(b) == 0)) {
                    chance_inf = (rand() / double(RAND_MAX));
                    if (chance_inf < probs) {
                            condition.at(b) = -1;
                    }
            }
    }
}
```

3 Analysis

For the contagion model that includes vaccinated people and probability of infecting the neighbor to the left and the probability of infecting the neighbor to the right, the following conclusions about the model can be drawn.

- The addition of randomly choosing a percentage of vaccinated people in the population reduces the number of people that could randomly get chosen as the first one sick.
- If both the first person and the last person in the population are susceptible from the start, the code loops so that the first and last person are neighbors. The first person, condition.at(0), has neighbors condition.at(1) and `condition.at(pop_size)`, and the last person, `condition.at(pop_size)`, has neighbors `condition.at(pop_size - 1)` and condition.at(0). Therefore, the population will loop around and every person has two neighbors.
- After the first day, the neighbors who are susceptible (values of '0') and next to an infected people are the most at risk of getting infected out of the entire population. Although, even if a susceptible person ('0') has an infected neighbor both to the left and to the right of them, there is still a possibility of escaping infection. However, with increasing days for the infection to last, it is highly unlikely that the susceptible person will remain susceptible throughout the entirety of both its neighbors infection days.
- With this specific model, the spread of infection is bounded by two vaccinated people. This can be seen in small and large populations. An example shown below uses a small population size equal to 9 with about 22 percent of the population vaccinated.

day 1	0	2	0	0	-1	0	0	2	0
day 2	0	2	0	-1	-2	0	0	2	0
day 3	0	2	0	-2	-3	-1	0	2	0
day 4	0	2	-1	-3	-4	-2	0	2	0
day 5	0	2	-2	-4	-5	-3	-1	2	0
day 6	0	2	-3	-5	1	-4	-2	2	0

  It can be seen that a higher percentage of the population that gets vaccinated will decrease the possibility of large numbers of susceptible people separating the vaccinated people. As a result fewer people will get infected out of the total population.
- In the model where an infected person has a 100 percent chance of infecting both of their neighbors, a chevron shape forms in the vector output for each day between two vaccinated people (i.e. every person next to a '-2' will be infected, so susceptible '0' will be '-1'). The vector output for this model looks similar to the 100 percent chance of infection, however, the chevron is chipped on the outer diagonal when a susceptible person escapes infection from their infected neighbor(s).
- The bound of vaccinated people on the population makes this model unrealistic because only a certain section of the population actually gets infected and the population outside of the infected bound never even randomly gets the possibility of infection after the first day.

The spreading model has a random spread of the infection through the population as the infected person spreads the disease to random people. The number of people is limited to six contacts per day and can include any variation of susceptible, already infected, recovered, or vaccinated people. With further analysis:

- After the first day, every person in the population, that is susceptible, is at an increased risk of getting infected with an increasing of the days.
- The more people who are vaccinated equates to less people have the possibility of being infected.
- The model finishes very quickly for its population size because each person has their six (possibly infecting) contacts with other people, which are mutually exclusive to each individual person. This means that any one person can have contact with six random people, however, an unlimited amount of random people can have contact with that one person. For example, if person two touches six other people, this model does not account for the other people touching person two at the same time and track both sides of the connection.
- From the way this model calculates infection, there is a high likelihood that every person in the population who starts the simulation as susceptible will get infected at low vaccination rates. Also, the infection spreads rapidly and because there is no reinfection, the simulation of the disease spread ends very quickly.
- Testing the code of the spreading model,

for a population of 1,000,000 people
and probability of transmission = 0.53 or 53 percent
(i.e. percentages in decimal form: 0.2 = 20 percent, etc.)

Simulation	Vaccination Percentage	Infected	Recovered	Susceptible	Vaccinated	Final Step(Day)
1	0.2	0	800,000	0	200,000	17
2	0.4	0	600,000	0	400,000	19
3	0.5	0	499,999	1	500,000	22
4	0.6	0	399,992	8	600,000	21
5	0.8	0	198,818	1,182	800,000	34

The table shows:

- with an increase in vaccination percentage, there will be an increase in the number of susceptible people once everyone in the population is recovered.
- that for a large population size of 1 million, the simulation is over between 17 and 34 days for an initial vaccination of the population of 20 percent to 80 percent, respectively. For a population of that size, in reality the simulation should run for much longer.
- that once between 50 percent and 60 percent of the population are vaccinated, some people start to remain susceptible throughout the entire population. This means that at the very minimum, over half of the population must be vaccinated before results show an exponential increase of susceptible people at the end of the simulation.

4 Possible Extensions

Both the contagion model and spreading model described above are very basic in terms of modeling the reality of an infectious disease. There are many other factors that need to be taken into account including;

- A change that would drastically impact the results of the spreading model would be to fix the problem of the mutually exclusive contacts. By creating unique vectors, of the size of the amount contacts (six), for each individual in the population array to hold the values of the positions of those they have been in contact with would track the number of new and old contacts. So, one person chosen randomly would get into contact with six other random people. The unique vectors would be checked for the first random person: if that random person has already been in contact with six others or previously with the first person. If the random person has space in its unique vector to come in contact with another person it has not previously had contact, then both the initial and the random person of the six will populate both of their vectors. If one of them is then infected, the disease spreads, otherwise no change.
- The models for the infectious disease spread above only look at a single population. A possible extension would be bringing in other populations as well and mixing in infected and vaccinated people. This is thinking about people flying into a neighboring country which has a lower/higher portion of its population infected compared to the originating country with a higher/lower infected population.
- These models only have someone infected once and then not ever again. Also, the vaccinated can never get sick. However, with certain infectious diseases, it is possible to get infected twice. There have been cases with COVID-19 where people have been infected again with a different strain of COVID-19. Also, vaccinations are not 100 percent effective, so the model should account for the inefficiencies.
- The models also have people vaccinated from the start of the virus. However, COVID-19 came up really quickly and there were no ready vaccinations at the beginning of the pandemic. To extend the models, vaccinations can be implemented on both infected and susceptible in the middle of the simulation.
- Another extension would be to take into account self-isolation and quarantining. Those who go into self-isolation have no contact with anyone for a certain amount of days, so they would not be able to spread the disease. However, those quarantining could still get in contact with a lesser number of people.
- It can be assumed that anyone in these models who are labeled as 'infected' are either carriers or sick, sick from having flu/cold like symptoms to being in hospital. A distinction could be made for those that get sick and those that just carry the disease and spread without symptoms, to those that actually die from the disease.
- One last addition to the models would be exposure time to the disease. An increased exposure to an infectious disease like COVID-19 has a higher likelihood of contracting it. So factoring in random exposure times separate from the probability of infection would produce more realistic results.

5 Summary

The contagion and spreading models are basic structures for an infectious disease spread; however, they are not realistic in terms of actual human reactions. Only four different states can occur for each person, but in the real world, there are many more states within an infectious disease that a person can be. The spreading uses six random interactions, but during the time of COVID-19, varying groups of people of different ages and different factors have had contact and disease hot spots formed. From the table of simulations for the spreading model, even with an unsatisfactory methods for contact, it can still be seen that with an increase in vaccinated people within a population, there will be an increase of uninfected people. This is

why vaccinations are so important for treating an infectious disease as they flatten the curve of the rate of infected people. With an even better model of spreading, more people will be able to escape getting infected. So the created models are just a beginning on structuring a true simulation of a human population, but there are many additional varying factors that need to be added to strengthen the precision of the results.