Infectious Disease Simulation

Ethan Chang (ehc586)
University of Texas - Austin
Scientific Computation (COE 322)

Abstract

Infectious diseases will always be a public health concern that affect populations, with their severity varying depending on symptoms, transfer rate, and whether there is a vaccine available. Due to the impact they can have on the lives of people, infectious diseases are being studied thoroughly and analyzed to discover the patterns they exhibit. As such, this project seeks to do just that, albeit in a much simpler manner. How does an infectious disease spread through the population? Does anyone escape being infected? How long does the disease run? This project answers these questions using a network model to simulate how a disease spreads from an infectious person throughout a population. Starting with one infected person, the program tracks the population through the course of multiple days until none of the population is sick. As there is no re-infection, the run will always end. To simulate a greater variety of scenarios, parameters such as population size, probability of transfer, and probability of vaccination can be adjusted. Data is then gathered after simulating these various scenarios to aid in answering the proposed questions.

Code, Results, and Discussion

56.1: *Person* Class

To start, code modeling a single person is written as a way to infect and track a person's state in the population. This is done by creating a *Person* class with the methods *status_string(), update(), infect(int n), is_stable()*, and *vaccinate()*. In this class, the private variable *state* is represented with integers: 0 for healthy but susceptible, -1 for recovered, -2 for vaccinated, and a value *n* that is greater than 0 for sick with *n* days before recovery. The *state* variable is used or adjusted with every method in the class. *status_string()* reads the *state* of a person and returns a string containing a message detailing the person's condition. *update()* updates a person's status to the next day, changing their *state* to "recovered" if they were "sick" and *n* went down to 0. *infect(int n)* changes the *state* of a person that was "susceptible" to "sick" for *n* days. I prefer to use

```
void infect(int n) {
        if (state==0)
                state = n;
}
```

For this method as it is a lot simpler and combines the *is_susceptible()* bool into it. *is_stable()* checks whether a person has recovered and returns a bool indicating whether or not that is true. Finally, *vaccinated()* changes the *state* of a person that was “susceptible” to “vaccinated.”

To test that everything in the *Person* class is working properly, the main program 56_1_main.cpp is run. For the purpose of consistency, every test will have a sick period of $n = 5$, meaning a “sick” person will be sick and can infect another for 5 days before they are fully recovered. The run returned the output displayed below, demonstrating that it was successful and all methods in the *Person* class are working appropriately:

```
On day 1, Joe is sick
On day 2, Joe is sick
On day 3, Joe is sick
On day 4, Joe is sick
On day 5, Joe is sick
On day 6, Joe is recovered
```

56.2: *Population* Class

Next, code to represent a population of *Person*s is made in the form of a second class: *Population*. This class makes a population vector of length *npeople Persons*. It includes the methods *random_infection(), count_infected(), update(),* and *display(). random_infection()* generates a random number between 0 and *npeople*-1(due to C++ vector indexing) to select a random *Person* in the *Population,* by rounding the C language randomly generated number, and infects them for 5 days. *count_infected()* uses a for loop to go through every *Person* in the *Population* and checks if they are “sick,” increasing the counter *num* every time one is found. *update()* utilizes the *update()* method in the *Person* class to update the *state* of every *Person* in the *Population* with a for loop. Finally, *display()* only serves to display a model using + for sick, ? for susceptible, and - for recovered people using a for loop to go through every *Person* and if statements to check their *state* and output the appropriate symbol.

To test that everything in the *Population* class is working properly, the main program 56_2_main.cpp is run. The run returned the output displayed below, demonstrating that it was successful and all methods in the *Population* class are working appropriately:

```
size of population?
20
day 1 number infected : 1
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? + ? ? ?
day 2 number infected : 1
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? + ? ? ?
```

```
day 3 number infected : 1
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? + ? ? ?
day 4 number infected : 1
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? + ? ? ?
day 5 number infected : 1
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? + ? ? ?
day 6 number infected : 0
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? - ? ? ?
Disease ran for 6 days
```

56.3: Probability of Transfer

Now that a *Population* exists, in order to advance the simulation, a method to transfer the disease is created. Starting with a simplistic approach, the method *random_transfer_probability(float p)* is created. This method uses a for loop to go over every *Person* in the *Population* and finds a person that is "sick." It then takes the inputted probability *p* and generates a random number between 0 and 1. If that number is less than *p* then the direct neighbors of the sick person also get infected. This method assumes every person in the population only interacts with the same 2 neighbors every day.

Multiple simulations were run varying population sizes and contagion probabilities. One such simulation consisting of a population size of 10 and a contagion probability of .3 is displayed below using the main program 56_3_main.cpp:

```
size of population?
10
probability of transfer? (between 0 and 1)
.3
day 1 number infected : 1
? ? ? ? ? ? ? ? ? +
day 2 number infected : 1
? ? ? ? ? ? ? ? ? +
day 3 number infected : 1
? ? ? ? ? ? ? ? ? +
day 4 number infected : 2
? ? ? ? ? ? ? ? + +
day 5 number infected : 3
? ? ? ? ? ? ? + + +
day 6 number infected : 2
? ? ? ? ? ? ? + + -
day 7 number infected : 2
```

```
? ? ? ? ? ? ? + + -
day 8 number infected : 1
? ? ? ? ? ? ? + - -
day 9 number infected : 0
? ? ? ? ? ? ? - - -
Disease ran for 9 days
```

As seen in this example, there are definitely cases where people can escape getting sick depending on the contagion probability.

56.4: Vaccination

Another factor that affects the spread of a disease is whether a vaccine exists. Vaccines serve to mitigate the spread of diseases by decreasing the maximum number of people that can get infected, and thus spread the disease. Vaccination is incorporated into the code via the method *vaccine_probability(float v)*. This method uses a for loop to go through the entire population and generates a random number between 0 and 1 for each person. If the number is less than v, then the person gets vaccinated. This serves as a way to randomly vaccinate people in the population.

To view the effect vaccinated people have on the spread of the disease, a simulation using the main program 56_4_main.cpp is run with population size 20, contagion probability .5, and vaccination probability .3.

```
size of population?
20
probability of transfer? (between 0 and 1)
.5
probability of vaccination? (between 0 and 1)
.3
day 1 number infected : 1
? v + ? ? ? v ? ? v v ? v ? v v ? ? v v
day 2 number infected : 2
? v + + ? ? v ? ? v v ? v ? v v ? ? v v
day 3 number infected : 4
? v + + + + v ? ? v v ? v ? v v ? ? v v
day 4 number infected : 4
? v + + + + v ? ? v v ? v ? v v ? ? v v
day 5 number infected : 4
? v + + + + v ? ? v v ? v ? v v ? ? v v
day 6 number infected : 2
```

```
? v - - + + v ? ? v v ? v ? v v ? ? v v
day 7 number infected : 0
? v - - - - v ? ? v v ? v ? v v ? ? v v
Disease ran for 7 days
```

As seen in this simulation, the vaccinated people essentially serve as a barrier that prevents the spread of disease to the susceptible people behind them. This is a very unrealistic model when representing the effect vaccinated people have in a population as it is still possible for susceptible people to get infected even if a vaccinated person is between the susceptible and sick people. It also does not accurately represent the type of spread a disease has as people do not only interact with the same 2 people.

56.5: Spreading

To fix this issue, the *random_transfer_probability(float p)* method is adjusted to allow an infected person to interact with 6 random people every day as opposed to only the people next to them. This is done by adding a new variable *ri* to represent the 6 random people each person interacts with and replacing the if statements after checking whether a person is sick with a while loop to choose a random person in the population and check whether they are susceptible. It then generates a random number and compares it to p like before to determine whether that random person gets infected or not. An example of how this new transfer method works is displayed below using the main program 56_5_main_1.cpp:

```
size of population?
20
probability of transfer? (between 0 and 1)
.5
probability of vaccination? (between 0 and 1)
.3
day 1 number infected : 1
v ? ? v v ? ? + ? ? ? v ? ? ? ? ? ? ? v
day 2 number infected : 5
v ? + v v ? ? + ? ? + v + + ? ? ? ? ? v
day 3 number infected : 13
v + + v v + + + + + + v + + + + ? + ? v
day 4 number infected : 14
v + + v v + + + + + + v + + + + + + ? v
day 5 number infected : 15
v + + v v + + + + + + v + + + + + + + v
day 6 number infected : 10
v + - v v + + - + + - v - - + + + + + v
```

```
day 7 number infected : 2
v - - v v - - - - - - v - - - - + - + v
day 8 number infected : 1
v - - v v - - - - - - v - - - - - - + v
day 9 number infected : 0
v - - v v - - - - - - v - - - - - - - v
Disease ran for 9 days
```

NOTE: For the rest of the tests, a realistic population size of 100,000 people will be used

Using this new *random_transfer_probability(float p)* method, 10 simulations, using the main program 56_5_main_2.cpp, are run varying the vaccination percentage by .1 in order to acquire how long the disease runs through the population. The data acquired for these simulations is represented in the table below comparing vaccination probability and number of days. A fixed probability of transmission of .5 is used for every simulation.

Vaccination Probability	**Number of Days**
0	13
0.1	14
0.2	14
0.3	15
0.4	16
0.5	18
0.6	22
0.7	27
0.8	39
0.9	71

When analyzing the above data it can be seen that as the vaccination probability increases, the number of days that a disease remains present in a population increases. This is most likely due to the fact that as the percentage of vaccinated people go up, the probability that someone gets infected in a day decreases. As such, depending on the probability of transmission and randomly generated number, each iteration can either go for a really long time as the times of infection are more spread out due to there being less people to infect per day or end really fast as the disease never gets transmitted to a susceptible person. Ultimately, all of these data are dependent on the randomly generated number.

Furthermore, when plotting these data using function approximation with regression analysis, a function of $numDays = 270(vacPer)^3 - 242.54(vacPer)^2 + 62.28(vacPer) + 11.32$ is obtained. The function of best fit is this cubic function, obtained through cubic regression with a correlation coefficient of 0.9893, relating the vaccination percentage and how long a disease runs in a population.

56.6: Herd Immunity

Finally, another aspect of the relationship between these parameters is the idea of herd immunity. This essentially means that if enough people are vaccinated, then some susceptible people will still never get sick. However, the percentage of people that need to be vaccinated to achieve a specific herd immunity varies depending on the probability of transmission of the disease. In order to model this, the method *count_susceptible()* was created with a function similar to the method *count_infected()*. This new method is then used in order to count the number of susceptible people in a population before and after the disease runs its course in order to take the quotient and obtain a probability representing the herd immunity. This probability is then compared to .95, the probability we want the herd immunity over to investigate the data of.

In order to find the percentage of vaccination needed to reach a herd immunity probability of .95 as a function of the contagiousness of the disease, the main program is edited to input 10 different p and v values with a margin of .1 and find the herd immunity probability of all cases, resulting in 100 data points displayed in the table below.

Vaccination Percentage	**Herd Immunity (p>.95 highlighted green)**									
.9	1	1	1	.998	1	.842	.8	.749	.727	.708
.8	1	1	1	.787	.723	.703	.688	.676	.675	.682
.7	1	1	1	.715	.69	.678	.673	.672	.676	.673
.6	1	1	.725	.687	.677	.672	.673	.675	.673	.676
.5	1	1	.697	.679	.675	.671	.674	.671	.673	.673
.4	1	1	.689	.675	.672	.673	.673	.673	.674	.672

.3	1	1	.681	.675	.673	.673	.673	.673	.67	.672
.2	1	.727	.677	.673	.671	.673	.673	.672	.673	.672
.1	1	.712	.674	.673	.672	.672	.671	.673	.673	.672
0	1	.701	.674	.672	.672	.672	.672	.672	.672	.672
	0	**.1**	**.2**	**.3**	**.4**	**.5**	**.6**	**.7**	**.8**	**.9**

Probability of Transmission

The data presented in the table demonstrate that as the probability of transmission increases, the vaccination percentage necessary to maintain a herd immunity probability above .95 increases until it eventually plateaus at approximately a probability of transmission of .5. To obtain points to approximate a function, the minimum vaccination percentage value necessary to obtain a herd immunity probability above .95 for every probability of transmission value would be used. When plotting these data using function approximation with regression analysis, a function of $vacPer \geq 3.38(probTran)^3 - 6.99(probTran)^2 + 4.7(probTran) - 0.03$ is obtained. The function of best fit is this cubic function, obtained through cubic regression with a correlation coefficient of 0.9937, relating the probability of a disease being transferred and the vaccination percentage.

Ethics Discussion

In a real world setting, there will be situations where people's actions affect the impact they have on the spread of a disease in a population. Some of these actions include getting vaccinated and interacting with many people. Depending on the situation, as well as the scale and transmissivity of the disease, this could lead to disastrous consequences. As seen in the simulation, although the probability of vaccination increased, the time before a disease finished its course demonstrated an increasing pattern, although some of this was largely affected by random number generation. This pattern demonstrated in the simulation shows just how disastrous of an effect even a few unvaccinated people can have on a population. By interacting with many people per day and not vaccinating, the spread of a disease would tend to last a bit longer as the time in between infection widened, making it slightly easier for more people to get infected since there is a larger period of time where there are infected people. As such, the safest and most assured method of preventing the spread of disease and dampening the consequences would be to get as many people (close to max) vaccinated as possible.

Conclusion

After multiple tests and simulations, it can be seen that this simple model of representing the spread of an infectious disease, while with its flaws, functions appropriately. It is capable of predicting the spread of a disease throughout a population if given enough parameters to an extent. Do bear in mind that this is merely a simple network model, and thus does not work in a large variety of situations containing parameters not present in this simulation. As such, this project could be expanded upon to cover situations not covered in this one. For example, it could go over situations where social distancing exists, limiting the amount of contact between people. It could also cover situations where some people are asymptomatic, meaning they are infected and can spread the disease, but do not show any symptoms, resulting in less overall social distancing. Situations where a vaccination period, re-infection period, or variants that bypass old vaccines are introduced could also be covered. These types of simulations have endless amounts of possibilities that can be explored solely to predict the spread of infectious diseases, demonstrating the impact computational methods can have in aiding and predicting real world situations.