**Simulation of Disease Outbreak of the SIR model with C++ programming**

Author:

Jakob Long,

UTEID: Jrl4725

TACC Username: Kebabo

Jakob.long@utexas.edu

**Introduction:**

Within this project the scope was to develop a program that can simulate the effects of an infectious disease on a population depending on the changed parameters. Of which we based our program from a simplified version of the SIR (Susceptible, Infected, Recovered) model. The SIR model is an epidemiological model that approximated the number of people within a closed population that would be infected when exposed to a contagious disease. Where each state of the SIR model is a compartment: a percentage of the total population.

As such we created a model where:

* If a person is susceptible, they’re healthy but able to get sick.
* If a person is sick, they can make others get sick.
* If a person is recovered, they’re no longer sick and cannot get sick again.
* If a person is vaccinated, they’re unable to get sick.

More complicated models can be created, such as only allowing infection during a set period during illness, or even having multiple diseases at once. However, with our model we stuck with only one disease running through a population, and always beginning with only a single person being infected. Having multiple patient zeroes could be an interesting improvement upon this project and would likely yield even more interesting results.

With this we were able to implement a simple simulation of how disease can spread within a population, depending on different parameters: such as how long a person can be sick for, the rate of transmission between sick and susceptible people, and the percentage of the population that is vaccinated.

Now with a goal in mind of modeling a simple version of the SIR model, it was time to implement the idea. To achieve this goal, I planned on the usage of object-oriented programming in C++. Of which would allow great variability of the parameters in order to test for specific situations and obtain more clear and meaningful data.

It was from this data; we were able to see how specific percentages of a population being vaccinated impacts the spread of a disease overall. This allows for conclusions to be made about herd immunity, and about how long a disease will stay within a population.

To begin, we initially needed to create a way to infect a single person and allow him to recover after an n number of days. Additionally, a way to change their status from susceptible, to sick, to recovered or even vaccinated was required as well. To achieve this a class named Person in conjunction with an enum class named Status.

After ensuring that the single susceptible person could become infected, recover, or even be vaccinated a way to implement this across a large population was needed. Thus, an additional class was necessary, and was named Population for a well-fitting name. Within the new class, a vector of people using the Person class was created to simulate a large community of people. With this, the population could then be manipulated in ways to simulate the spread, recovery, and blocking of a disease.

**Discussion:**

As mentioned earlier, I began with the construction of a Person class, which was able to simulate the infection, recovery, and vaccination of a singular person. This was achieved with by writing a method to infect randomly, infect, based off the C language random number generator. Additionally, a method to update, update, the status of the person was needed as well. This method was created in conjunction with the enum class Status, and use of switch statements. As depending on the Person’s Status, the output and behavior of the program changed. As such here are some examples of sample output:

On day 1, Joe is susceptible

On day 2, Joe is susceptible

On day 3, Joe is susceptible

On day 4, Joe is sick (5 to go)

On day 5, Joe is sick (4 to go)

On day 6, Joe is sick (3 to go)

On day 7, Joe is sick (2 to go)

On day 8, Joe is sick (1 to go)

On day 9, Joe is recovered

Additionally:

On day 1, Joe is vaccinated

From the sample outputs, we’re able to see that the code is working properly as the recovery time used throughout the entire project was five days. However, this is only the beginning, and not where the fruits of our project begin to bloom.

With the construction of the Person class being finished, the Population class is now able to be fully constructed. As such I was able to implement the Person class, into the Population class for use as a vector, to store each Person. Through this I was able to construct an output something along the lines of:

Day # 22: #sick: 1 | #rcvrd: 0 # susceptible: 29999

Day # 23: #sick: 1 | #rcvrd: 0 # susceptible: 29999

Day # 24: #sick: 1 | #rcvrd: 0 # susceptible: 29999

Day # 25: #sick: 1 | #rcvrd: 0 # susceptible: 29999

Day # 26: #sick: 1 | #rcvrd: 0 # susceptible: 29999

Day # 27: #sick: 0 | #rcvrd: 1 # susceptible: 29999

From this point forward, I had to think about how to implement the random transmission between sick and susceptible individuals. This initially began with those who were sick infecting their direct neighbors, but was further evolved into setting the population to ‘interact’ with one another, and having a pre-set probability that one would become infected after interacting with someone who was sick. This is demonstrated within the interactions method:

 void interaction(Person\* person, vector<Person\*> &new\_infections)

        {

            for (int i = 0; i < contacts; i++)

            {

                double bad\_luck = (double) rand() / (double) RAND\_MAX;

                if(bad\_luck <= probability)

                {

                    int randnum = rand()%peoples.size();

                    new\_infections.push\_back(peoples.at(randnum));

                }

            }

        }

Where the probability, is manually input within the main program and obtained by a sub function probability\_of\_transfer(double probability) , where the private variable probability within the Population class then points at the input parameter given. Again, like the random infection method, infect, the new person infected was depended on a ‘bad luck’ and was completely random, and looped for the duration of the max # of contact that was provided in a similar fashion to the probability.

From here I was able to then do some real analysis of the results I was getting. As the model had evolved from the simplistic form of just stating when someone was susceptible to when they became sick, and to when they recovered, to that of modeling simple interaction of individuals within a population.

I created a couple of graphs displaying the lifespan of the disease, and the changes in those who are susceptible, infected, and recovered with 20% of the population being vaccinated. While keeping the population that was vaccinated constant, the probability or rate of transmission was changed.

Chart

Description automatically generated

Figure

Chart

Description automatically generated

Figure

Within figures one and two, the vaccination population was twenty percent of 30,000, so 6,000 people. However, the transmission probability was larger in figure 2, than in figure one. From here we’re able to see that the total infected had a much larger peak than in figure one; however, the overall duration of the duration of the disease was shorter, due to more people becoming infected and recovering. Despite the disease having a shorter lifespan within the population this is not necessarily a good thing.

As within a non-perfect scenario, that would a much larger count of those who were not able to properly recovery and passed due to the disease. In comparison, figure one has a smaller peak, and upward trend, thus giving more time to create safeguards or receive medical attention to combat the spread of the disease.

**Conclusions:**

As discussed earlier, within figures one and two we’re able to see the impact of vaccinations on a varying range of transmission rates. As a result, I determined that diseases with a higher transmission rate, aren’t affected as heavily when a portion of the population is vaccinated.

However, there are some limitations within my project. As the way those who were vaccinated were completely immune to the disease is not realistic. While vaccinations due provide an effective means of defense against specific diseases, no vaccine is 100% effective, thus; even those who were marked as vaccinated should have had a chance to become infected. Additionally, within my modeling of the disease spreading multiple patient zeroes could be used to fully implement the emergence of an epidemic, rather than stemming from one singular person every time.