Abstract

The purpose of this paper is to investigate how different conditions and disease properties can affect large populations of people. By simulating a disease outbreak in a sample population, it is possible to statistically determine how different properties of real-world diseases might affect large populations of people in the real world. To investigate this, an outbreak is simulated (originating from 5 infected individuals) under a sample population of 100. The results show that a disease's performance is dependent on the environments from which it originates; as well as the behaviour of infected individuals. The results demonstrate that the deadliest diseases are those in obstacle-filled environments that have the highest probabilities of being transferred, the lowest probabilities of recovery and lowest probabilities of death. Furthermore, the results indicate that the number of obstacles and chances of death of a disease are inversely related; implying that as obstacles decrease, the disease can kill its host faster without hindering its rate of spread. Therefore, it is demonstrated that in order to maximise total casualties within a given time period, a disease needs to have high rates of infection, low rates of recovery, and death rates that are dependent on its rate of spread.

Background

The purpose of this simulation is to examine the results of outbreaks of different diseases (with different characteristics) and how they behave with a specified sample population. The results of how the disease interacts with the population sample will facilitate extrapolatory predictions on how similar diseases interact with larger populations. In this section a basic background of the main functions and calculations used to generate the simulation data are provided as well as a definition of the choice of parameters being investigated.

NumPy. The python library NumPy (hereafter referred to as 'np') adds support for large multi-dimensional arrays and matrices [SciPy.org 2019]. The inclusion of this python library in the simulation allows us to run calculations for multi-dimensional arrays and values that are contained within them. The function `np.zeros(row,col)` returns a new array of the given shape filled with zeros and was used to define the barriers, infected, uninfected and immune 'grid spaces'.

Random. The python module Random is used in the simulation to generate random integers to 'randomly distribute' the sample population to different index locations of the np multi-dimensional array.

Matplotlib. This is another python library which enables the production of publication-quality figures in a variety of hardcopy formats and interactive environments [Matplotlib development team 2018]. It is incorporated into the simulation to graphically represent and allow visual inspection of the disease's behaviour on the simulation's population across 'timesteps' as it spreads.

Sys. This module provides access to variables used by the system interpreter [Python Software Foundation 2019]. It is incorporated into the simulation to facilitate a parameter sweep; the effect of changing the characteristics of the disease.

barrierdis(). This is another crucial function that is utilised in the program. This function replaces a np array index with a 1 integer if it matches the user-specified barrier layout. If this layout is not defined from a csv file, it will default to replacing the outside rows and columns (surrounding the array) with 1s.

distribute(). This function adds one to a random index for its specified np array. It will keep adding one until it matches the sample population's size. If the x and y values match an index of the barrier np array that contains a 1, the function will generate a different random index.

movePeeps(). This is the main function of the simulation that determines the behaviour of the sample population. It runs either a Von Neumann or Moore neighbour calculation to the current np array and ultimately creates a new one.

- The Moore calculation (based on the previous np array) will either add one or subtract one or do nothing to the separate x and y index values in the np array.
- The Von Neumann calculation (based on the previous np array) will have the same logic as the Moore calculation, except it will not add or subtract one to both the x and y value of the current cell's index. This means that it will not consider the cells diagonally of the current cell's index.

After this function runs, if it matches the index of the barrier np array (that contains a 1) it will recalculate the random integers until it no longer matches. This gives the barriers their 'solid' properties.

The user-defined parameters that are being investigated in this report are the probabilities of infection, recovery and death. These parameters have been chosen because they are similar to real life disease characteristics. For example, the common cold is more infectious than leprosy and rabies is more fatal than chickenpox.

The characteristics of the disease are defined by the infect(), recover() and die() functions.

infect(). This function cross-references the indexes of the diseased np array with the indexes of the uninfected np array. If both array's indexes contain values greater than 0, a random float between 1 and 0 is generated for each index. If this float is greater than the user-defined chance of infection, one will be subtracted from the uninfected np array while one is added to the infected np array at the specified index.

recover(). This function behaves the same way as infect() except in the opposite way. Specifically, if the random float (between 1 and 0) is greater than the user defined probability of recovery, the uninfected np array will add one to the current index and the infected np array will subtract one.

die(). This function will generate a random float between 1 and 0 for every index in the infected np array that contains a value greater than 0. If this random float is greater than the user defined probability of death, one will be subtracted from the given index on the infected array.

It should be noted that this is not a list of all the functions and/or libraries that have been implemented in the simulation. They are however required to be defined for the methodology of this investigation to be sufficiently understood.

Methodology

Since this investigation involves the comparison of different disease characteristics and how they behave on a sample population, a parameter sweep will be utilised to sweep over incremental floats that will be used to define the probabilities of infection, recovery and death.
The probabilities for each characteristic will be confined within the bounds of 0.1 through to 0.5. The reason the probabilities are restricted to these boundaries is so that the simulation can have realistic characteristics. For example, if the probability of infection is 1.0 and the probability of recovery is 1.0, the simulation would not mirror real life as accurately since the entire population would immediately recover after they have become infected and no one would die. Hence, the

spread of disease and not as much data would be generated. Since this is not a realistic scenario (i.e. nobody's immune system can instantly eradicate an infection) the selected boundaries are more appropriate.

Ten parameter sweeps of both Moore and Von Neumann cell behaviours were conducted in order to generate our statistics. Furthermore, five of each trial was conducted using different grid layouts that altered the sample population's movement ability in a ten by ten grid. The first grid layout (figure 2) contained no obstacles whereas the second layout (figure 3) contained large obstacles in the centre of the grid space. This was conducted to get an indication of how the spread of the disease changes if the sample population's movements are restricted by boundaries.

In the figures at the end of this report, blue dots represent the uninfected population, red dots represent the infected population, green dots represent the immune population and squares represent the boundaries and/or obstacles. The dots are all slightly opaque. This has been done so that the grid will display if dots of different colours occupy the same index location.

For each trial, ten 'timesteps' were implemented to give the simulation time to run. This number was selected because it was a sufficient middle-ground between generating enough data to draw conclusions from, as well as maintaining a fast data generation time.

A sample population of size 100 was utilised to generate the results. Of these 100 people, 5 started the simulation as 'contaminated infectants'. Furthermore, 1 person in each experiment was given an immunity to the disease. This was implemented to mirror real life scenarios where a chance of an individual possessing a genetic immunity to a disease is a real possibility. This means that a maximum of 94 people could have become infected with the disease and hence, a maximum of 99 people could have died.

Results were calculated by taking the average of 5 trials (separate parameter sweep's greater_statistics.txt files) for the different trial conditions. Each sweep stepped up each characteristic by 0.1 for each probability.

Results

The table below (figure 1) contains the average statistics for the 5 parameter sweeps for each type.

Statistics	Moore (No Obstacles)	Von Neumann (No Obstacles)	Moore (Obstacles)	Von Neumann (Obstacles)
Most Infections	33	34	40.8	39.4
Most Recoveries	33	34	40.8	39.4
Most Deaths	19.6	16.8	20.6	21.6
Most Left Infected	13.6	11.2	15.8	15.8
Most Left Uninfected	100	100	100	100
Most Left Alive	100	100	100	100
Lowest Infections	0	0	0	0
Lowest Recoveries	0	0	0	0
Lowest Deaths	0	0	0	0
Least Left Infected	0	0	0	0

Least Left Uninfected	75	74.8	71.2	69.8
Least Left Alive	80.4	83.2	79.4	78.4
Average Infections	4.6128	3.9696	5.616	5.4176
Average Recoveries	5.3392	4.9696	5.768	5.7504
Average Deaths	3.832	3.584	4.272	5.3648
Average Left Infected	0.4416	0.416	0.576	0.5024
Average Left Uninfected	95.7264	96	95.152	95.3382
Average Total Left Alive	96.168	96.416	95.728	95.8352

(Figure 1)

From the results in figure 1, several observations can be made.

Firstly, the total left alive on average decreases when more obstacles are introduced. This an enlightening observation since one would expect the obstacles to somewhat prevent the spread of disease. What could be explain this property however is that as the obstacles take up more room of the grid space, there are less possible indexes for distribute() to distribute people too. This would result in an overall higher concentration of people throughout the grid space and hence it can be inferred that as the grid space decreases due to more obstacles being introduced, (depending on the arrangement of those obstacles), people are more likely to die.

Another observation that can be seen in the table is that the average deaths, average recoveries and average infections all increase the more obstacles there are. This makes sense considering if the overall concentration of people is greater, the more chance any one individual person has of becoming infected, recovering or dying.

It can also be determined from the results that the number of timesteps was (on average) not long enough for all the individuals to either die or recover from infection. This can be seen by the 'average left infected' statistics being greater than 0.

If a 'successful' disease is one that causes the most deaths in a population...

By drilling down into the results used to calculate the averages of the 5 trials for each state, it can be determined that the most successful disease overall was one that occurred in the Von Neumann experiments with obstacles. By looking at the raw data used to calculate this column in figure 1, it can be observed that the most deaths that occurred was 29 and it occurred in trial 2, experiment 101. This disease had probability characteristics of 0.1 for death, 0.1 for recovery and 0.5 for infection which implies that under Von Neumann conditions (with obstacles) high infection rate is favourable to high death rate.

There were several 'least successful' diseases as can be determined by the most left alive column being equivalent to 100 (or the same size as the sample population). What is inferred by this statistic is that these diseases did not have good enough characteristics to cause any deaths. By drilling down into the raw data, it can be determined that the 'least successful' diseases had characteristics where the chances of recovery were (on average) greater than 0.4 and the chance of death and infection were (on average) less than 0.3. Hence, this can be determined as the limit for disease viability in a grid space with obstacles.

The most ‘successful’ disease in a grid space with no obstacles occurred in the Moore experiments. By looking at the raw data the most ‘successful’ disease with no obstacles can be determined to have characteristics of 0.5 for infection, 0.1 for recovery and 0.4 for death. Interestingly, by comparing this to the characteristics of a disease with 0.5 infection, 0.1 recovery and 0.5 death, these diseases weren’t as successful. What can be concluded from this is that if infection rate is higher than the death rate, the disease is more ‘successful’ in an environment with no obstacles.

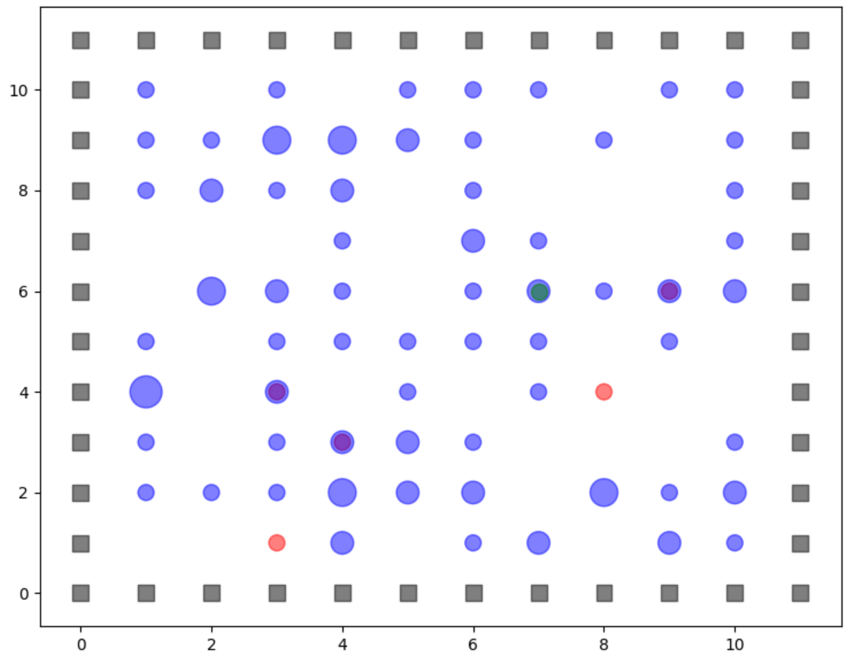
Likewise, the ‘least successful’ disease benchmark in an environment with no obstacles can be determined as one that has probability characteristics of (on average) less than 0.2 infection and 0.2 death, provided they have at least (on average), 0.4 chances of recovery. This makes sense considering obstacles hinder the of disease’s rate of spread. Therefore, by lowering the amount of obstacles in a given environment, the disease does not have to have as low of a death chance in order to maximise its chances of infection across a given time period.

Conclusion

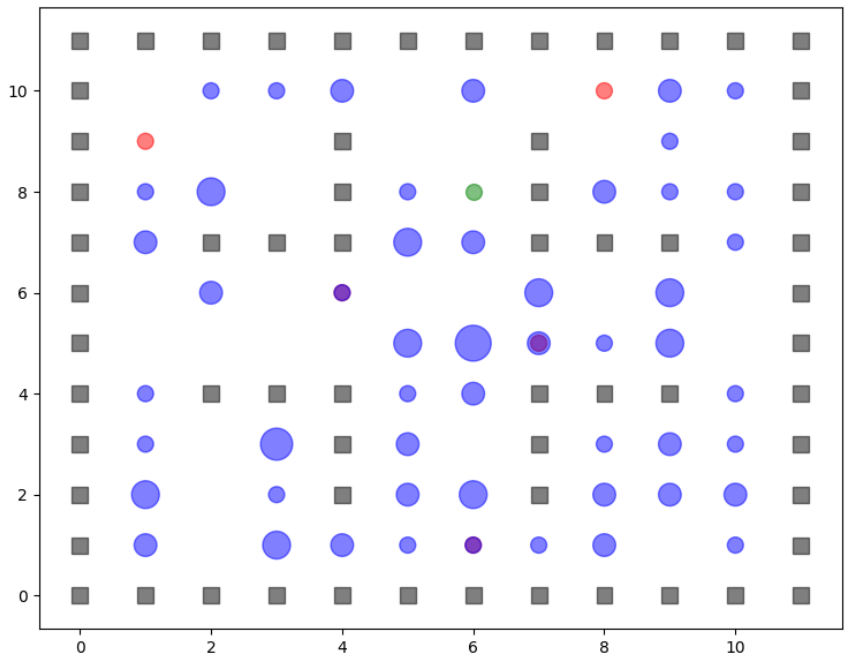
What can be determined from this investigation is that a disease’s effectiveness ultimately depends on its environment. If a disease originates in an environment where there are lots of people and it is difficult for these people to become exposed to the disease, then the chances of death after an individual becomes infected should be lower in order to maximise the disease’s effectiveness. From the results generated it can also be determined that if a disease does not have sufficient rates of infection or death, it can be easily fought off by an individual’s immune system and will eventually be eradicated. Hence it is only possible to increase these rates if it becomes easier for the disease to spread, either from the removal of obstacles or increasing its infectability. One example of a disease that mirrors these characteristics particularly well is HIV/AIDS. Although HIV does not have a particularly high mortality rate (approximate five percent) it is highly infectious and accounts for approximately three percent of all global deaths every year [Carter 2008]. Further investigations could utilise larger parameter sweeps to improve the accuracy of results as well as the inclusion of more statistical data; like variance and margin of error.

Appendix

(Figure 2)



(Figure 3)



Bibliography

Carter, Michael. 2008. "Risk of death for people with HIV now similar to that seen in the general population." *NAM Aidsmap*. 2019. http://www.aidsmap.com/Risk-of-death-for-people-with-HIV-now-similar-to-that-seen-in-the-general-population/page/1430739/.

Hunter, John, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team. 2018. "Matplotlib." The Matplotlib development team. https://matplotlib.org/.

NumPy developers. 2019. "NumPy." SciPy.org. https://www.numpy.org/.

Python Software Foundation. 2019. "sys – System-specific parameters and functions." The Python Software Foundation. https://docs.python.org/2/library/sys.html.