Abstract

The purpose of this paper is to investigate how different conditions and disease properties can affect large populations of people. Through the simulation of a disease outbreak it is possible to determine how different properties of different dieseases can affect large populations of people.

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 (1). 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 (2). 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 (3). It is incorporated into the simulation to facilitate a parameter sweep; the effect of changing the characteristics of the disease.

**barrierdis().** This function replaces a np array index with an integer of one if it matches the user-specified barrier layout.If this layout is not specified from a csv file, it will default to replacing the outside rows and colums (surrounding the array) with ones.

**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 index values match those of an index in the barrier np array that contain an integer of one, the function will generate a different random index and try again.

**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 only add or subtract one to either the x or y value of the current

cell’s index. This means that it will not consider the cells diagonally of the current cell’s index.

If this new index matches the index of the barrier np array (that contains a one) 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 simulated disease are defined by functions explained below.

**infect()**. This function crossreferences 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 of the functions and/or libraries that have been implemented in the simulation. They are however required to be defined in order for the methodology of this investigation to be sufficiently understood.

Methodology

Bibliography

1. https://www.numpy.org/

2. https://matplotlib.org/  
3. https://docs.python.org/2/library/sys.html