Simulating A Pandemic

Adam Abuobaid, Felix Dubicki-Piper,
Edward Emeny, Barnaby Philcox-Booth, Ashiph Rai
Y1 Engineering Mathematics, University of Bristol

May 14, 2021

1 Program Overview

The aim of our program is to simulate how a population would be affected by the COVID-19 pandemic, depending on the different measures they took against it.

We base the simulation on a grid-based SIR model, meaning each grid point represent a person, who may be susceptible, infected, recovered or dead. We develop on this by introducing the idea of 'sub-populations' and 'travelling', using multiple 'grids', which people can 'travel' between or within. These sub-populations may represent cities, countries, or any situation where there are multiple subsets of people situated in separate locations. Furthermore we also introduce 'pandemic response' factors such as social distancing, quarantining and vaccination. Finally we have a 'reinfection' factor, representing the potential loss of immunity to COVID over time, but also the rise in variants which may be resistant to our antibodies from the first strain.

Our model is also largely based on probabilities rather than absolute numbers, so while the same inputs usually produce similar behaviour, every simulation is unique, so can occasionally result in more surprising outcomes. This however is much like in real life, which can be unpredictable, given the same set of starting conditions as tiny changes can have large consequences. We base these probabilities on real world data [1] [2] unless specified differently in the README.md file.

1.1 General structure

The three main scripts in our program are `SIMULATION.py` (`SIM`), `ANIMATION.py` (`ANI`) and `MAIN.py` (`MAIN`). `SIM` generates the actual simulation, stored as a grid of 'states', which is used by `ANI` to generate the animations. `MAIN` is where the program is run from, and includes some specific pre-written scenarios. The fourth script `simulation2.py` (`sim2`), uses an different pandemic modelling technique, which attempts to track the unique probabilities for individual people, but can also be used with `ANI`.

1.2 SIMULATION.py

This file contains the classes `subPopulationSim` (`subPop`); `populationSim` (`Pop`). `subPop` is used to simulate a single grid of people, and is responsible for the majority of the simulation. It contains the following methods:

- `randomInfection`, used to initially infect the sub-population.
- `emptyLocation` randomly assigns 'empty spaces' representing social distancing.
- `randomVaccination`, used to randomly vaccinate a proportion of the sub-population.
- `updateStatus` determines the new status of an individual grid point, based on its current state and a randomly generated number.

- `updateProb` inspects the neighbours of a single grid point to determine its probability of being infected.
- `update` loops through every grid point and applies `updateStatus`.
- `collectData` and `get_Colours` are used to aid in producing the animations .

`Pop` takes a list of instances of `subPop`, and determines the probability of being infected by a traveller, using the `populationTravel` method. It similarly contains an `update` method, which updates each sub-population, and a `collectData` method, which collates all the sub-population data.

1.3 ANIMATION.py

`ANIMATION.py` uses three classes:

- `gridAnimation` generates a grid of colours using an RGB matrix output from `get_Colours`.
- `lineAnimation` retrieves population data using `collectData`, which is plotted as a line graph.
- `Animation` uses `matplotlib`'s `funcanimation` to animate a grid for each sub-population along with a combined line graph, utilising the previous two classes.
- Alternatively, `individualAnimation` animates a grid and line plot for each individual sub-population.

We also include a method to save the animations rather than play them live.

1.4 MAIN.py

This is the file which runs the whole program. There are two sections to this script, the first deals with running a custom simulations defined by user input, and the second allows the user to run some predefined scenarios, some of which are more complex so cannot be run simply with user input (as they require unique parameters for each sub-population).

1.5 Running The Program

The program can be run from terminal (or other compatible console) using bash commands. The python packages `numpy`, pandas and `matplotlib` must be installed. `ffmpeg` is required for saving plots, but is optional. To run with default settings, run `MAIN.py` from the console. This produces a live simulation animation containing a line plot and two interacting sub-population grids, using default probabilities. The main user inputs include `--cities` (number of cities), `--size` (grid width) and `--cases` (number of intial cases). A full list, which includes all the probability parameters, can be found in README.md, or using `--help`. The animation can be saved by specifying a filename using `--file`, and a numerical data mode can be run by specifying `--mode=data`. Finally, it is possible to show individual data for each sub-population by specifying `--data=individual`, which plots a line graph under each grid (or individual numerical data if specified), as opposed to the default 'total' data mode.

2 Program Output

All the following outputs described can be run from `MAIN.py` using the the arguments stated in brackets. The output animations are all available in this YouTube playlist or using the individual blue hyperlinks provided below. YouTube also has an inbuilt playback speed option, to view the animations at different speeds. The following plots are example frames from the animations.

For reference, the grid colours are the same as for the line plot, except for 'travelling' people which are yellow, and 'empty spaces', coloured white. Quarantining people are a more magenta shade of red, so it is still clear they are infected, but results in a subtle gradient in appearance.

2.1 Coronavirus Simulations

Standard, Large Scale Simulation (`--cities=1 --size=400 --travel=0.0001`)

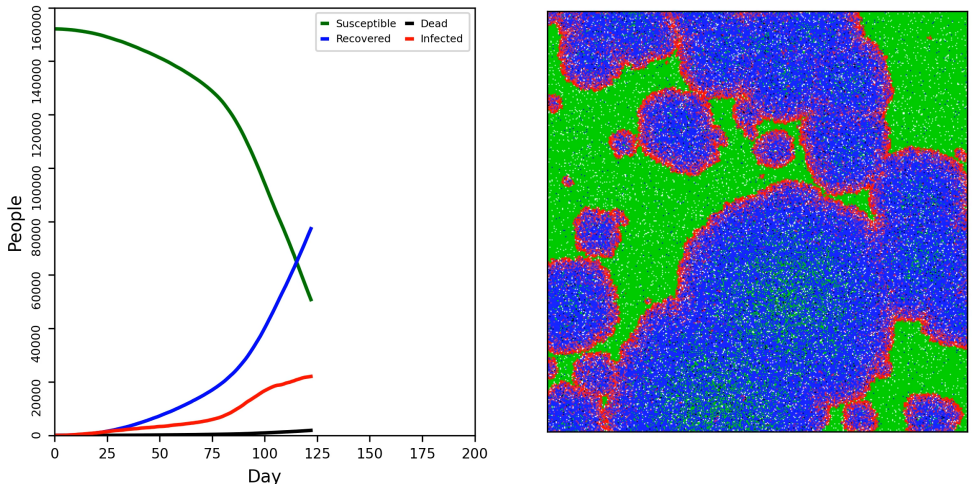
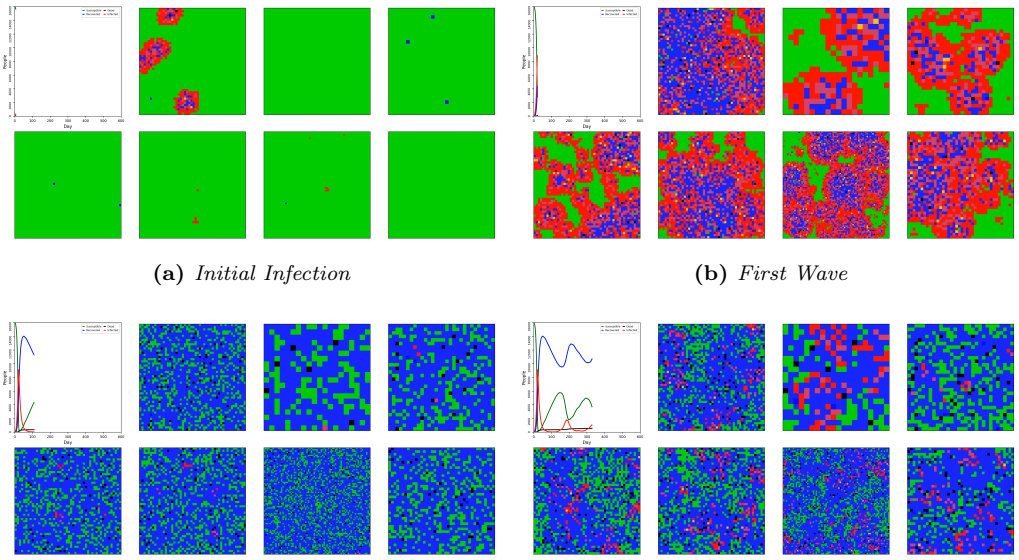


Figure 1: *Standard Animation*

This shows how COVID would spread in a city with some measures, such as individual quarantining and vaccination, but no national lockdown or social distancing. Over the 200 day period the virus spreads to almost everyone in the city.

Multiple Cities with Some Travelling (`--sim=1`)



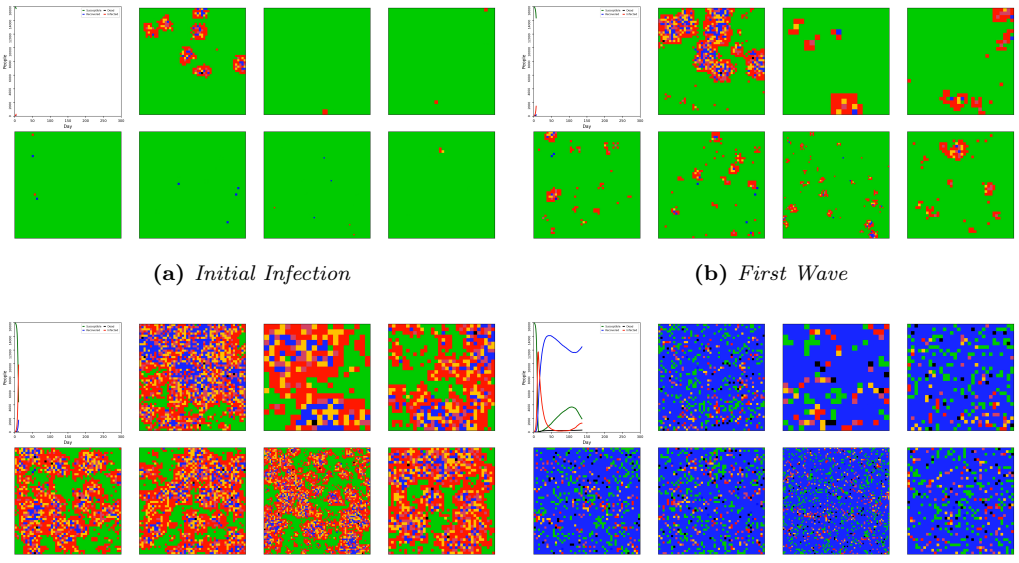
(a) *Initial Infection* **(b)** *First Wave*

(c) *Period of Low Activity* **(d)** *Third Wave*

Figure 2: *Low Travel Animation*

This demonstrates how travelling people spread COVID across and between cities. The probability of travelling while infected is 0.01. It shows that even for small amounts of travelling, there are still multiple waves of the virus.

Multiple Cities with Excessive Travelling (`--sim=2`)



(a) *Initial Infection* **(b)** *First Wave*

(c) *Peak Infection* **(d)** *Second Wave*

Figure 3: *High Travel Animation*

In contrast, lots of travelling results in a larger initial wave, yet less overall deaths. Perhaps this is due to herd immunity, as more people were infected in a short space of time. Here we set the probability of a person travelling while infected is 0.3.

This scenario is particularly relevant to the fact that for COVID-19, 80% of cases present with mild symptoms or asymptomatic [3], which may lead people to travel more often. This is why travelling between tiers during December 2020 in the UK was prohibited.

Multiple Cities with Social Distancing (`--sim=3`)

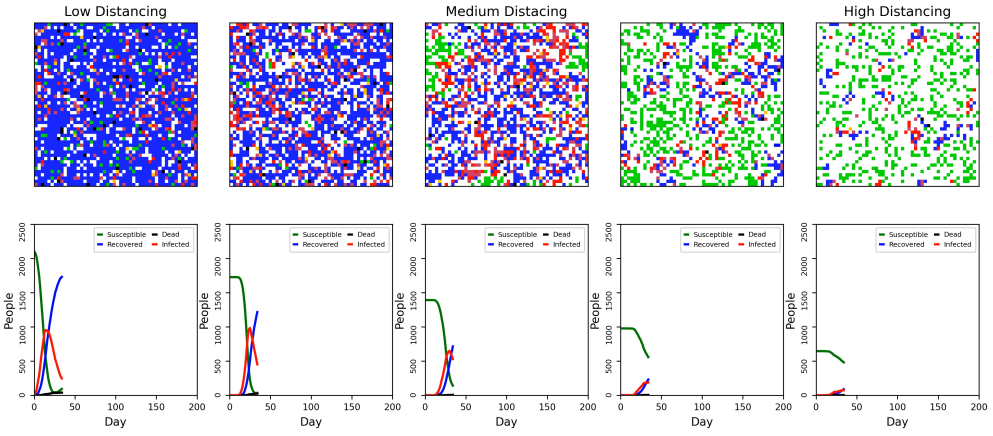


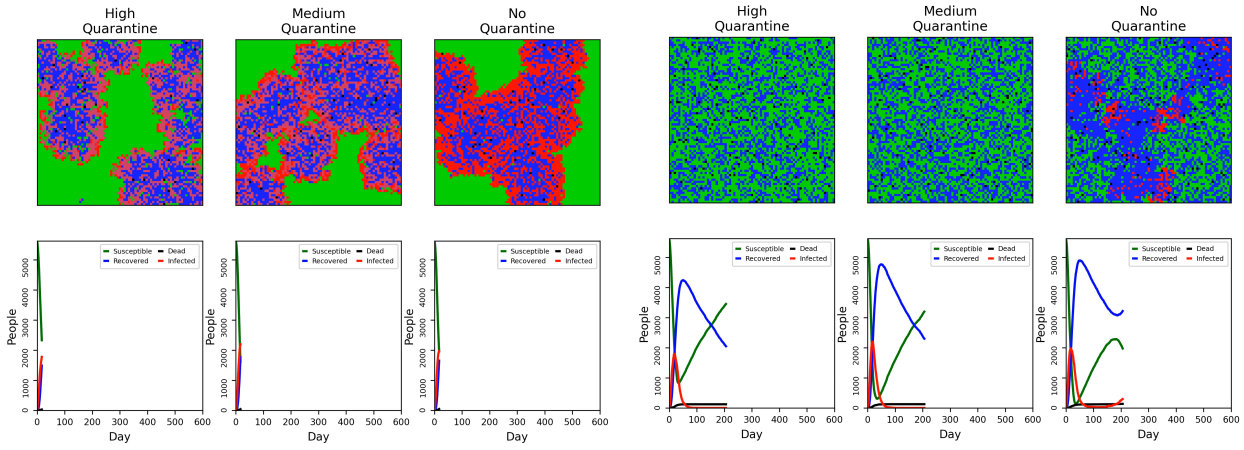
Figure 4: *Social Distancing Animation*

The simulation shows us how the peak of infections varies as social distancing increases, which could represent the UK's Government's measures just prior to, and during lockdown. We see, as expected that increased

social distancing slows the spread of COVID and hence flattens the curve.

Here we use the same travel probability of 0.1 however with different distancing levels ranging from a 0.15 to 0.75 proportion of 'empty space' in the grid.

Quarantine (`--sim=4`)

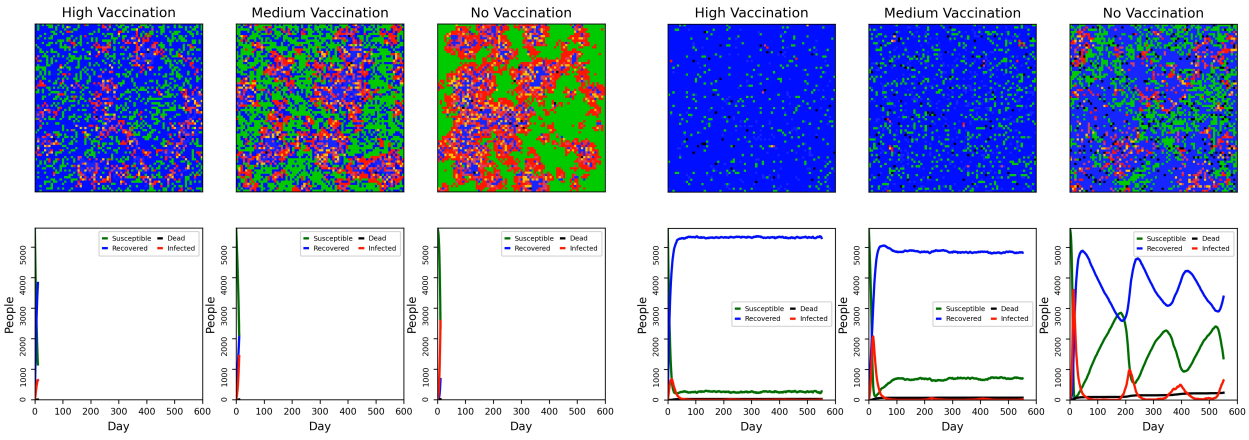


(a) *Initial Infection* **(b)** *Second Wave*

Figure 5: *Quarantine Animation*

This simulation shows that when people do not isolate after contracting the virus, it stays in the city for longer, causing multiple waves. The range used is from immediate quarantine after infection to a probability of 0.4 (i.e. after testing positive/showing symptoms) and finally no quarantine at all.

Vaccination (`--sim=5`)

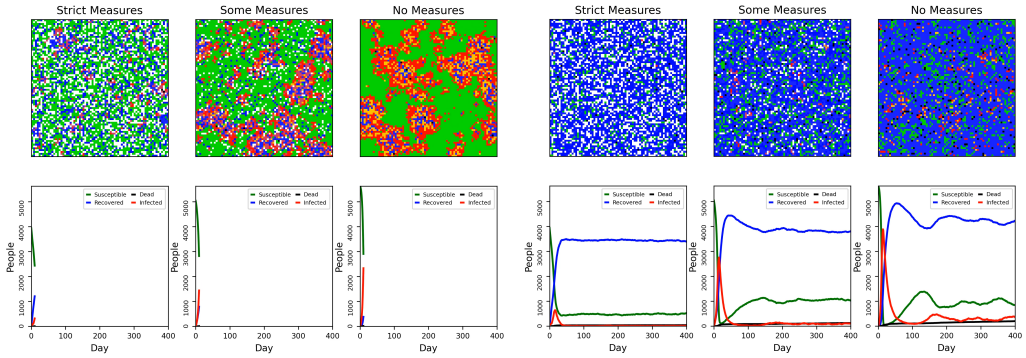


(a) *First Wave* **(b)** *Fourth Wave*

Figure 6: *Vaccine Animation*

Here we see vaccines can prevent multiple waves. As immunity begins to wear off, or resistant strains emerge, vaccines help slow the spread. The range of vaccination rates (per day) here are, 0.1, 0.033 and 0.

Combining Measures (`--sim=6`)



(a) *First Wave* **(b)** *Fourth Wave*

Figure 7: *Combined Measures Animation*

Here we have three cities, the first having probability of travel, quarantine and social distancing as 0, 0.95, 0.3 respectively. the second has 0.1, 0.3, 0.1 and the last, with no restrictions has 0.3, 0, 0.

We see that no restrictions allows for the virus to spread far more quickly and resulting multiple waves as peoples immunity fall away (or new variants emerge).

Death Rates (`--sim=7`)

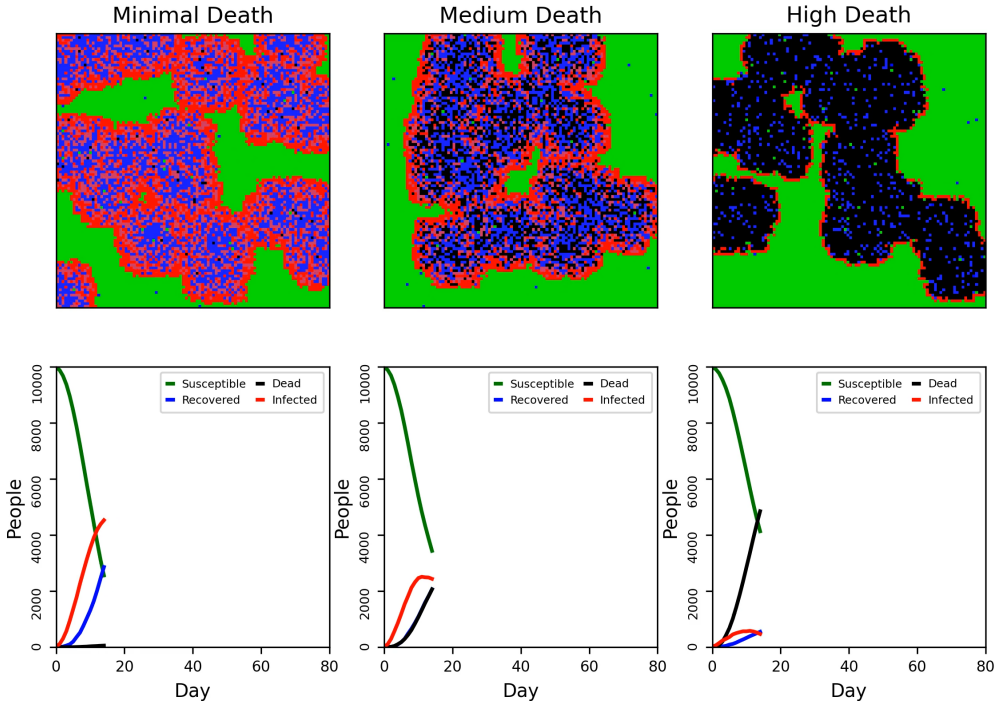


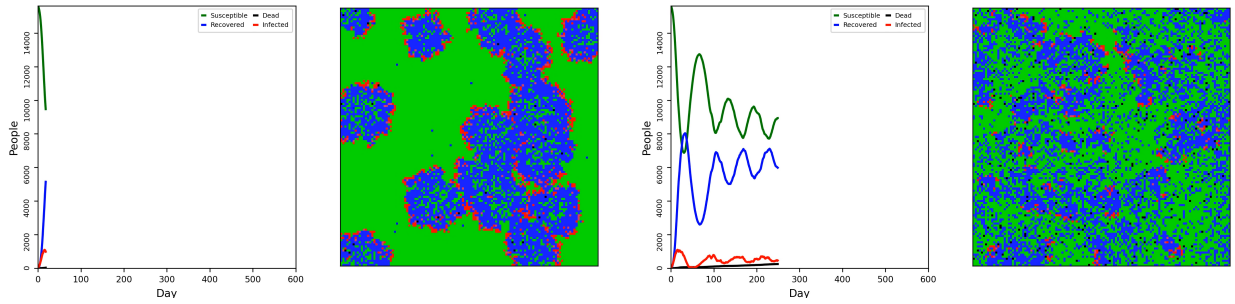
Figure 8: *Death Rates Animation*

As expected, a higher death probability (per day) result in more deaths. Higher deaths rates could represent older populations, or populations with poor access to sufficient medical equipment.

2.2 Experimental Simulations

The following simulations use parameters which are not particularly representative of coronavirus but are interesting nonetheless, as they may represent other kinds of diseases.

Medium & High Recovery Rates, with Immunity Loss (`--sim=101` and `--sim=102`)

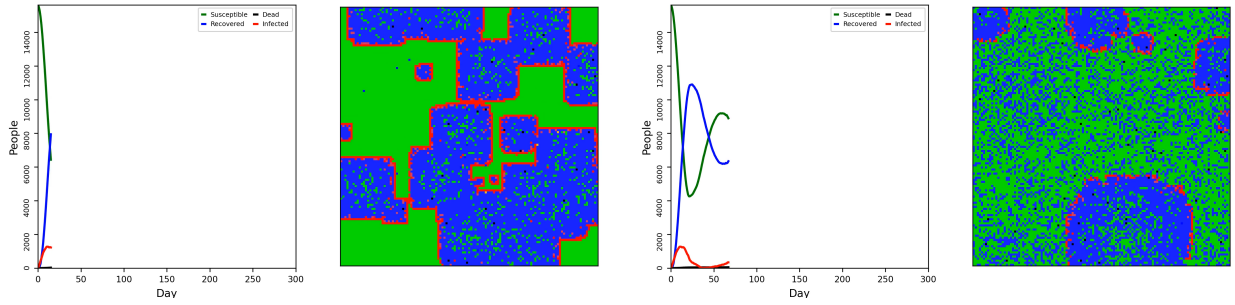


(a) *Initial Wave* **(b)** *Circulation Later On*

Figure 9: *'Continuous Circulation' Animation 1 (above) and Animation 2 (not shown)*

In these simulations, the infection rates appear to oscillate steadily (though slightly dampened over time), with the virus continuously circulating throughout the grid, rather than appearing in sudden waves. The 'high recovery' version is very similar to the former, just appearing more 'sparse' on the grid. These could represent something more like the common cold, which circulates and mutates on a yearly basis, but has a fairly short recovery time.

High Recovery and Infection Rate, with Immunity Loss (`--sim=103`)



(a) *First Wave* **(b)** *Second Wave*

Figure 10: *'Pulsating' Animation*

In this case, the long term behaviour is similar to the previous two, however in the short term the waves of the virus appear to pulsate regularly from a small number of areas, rather than spreading more chaotically. There is also a very small amount of travel for this case, otherwise this behaviour stops very rapidly, due to 'herd immunity' preventing spread to neighbours.

Slow but Guaranteed Death (`--sim=104`)

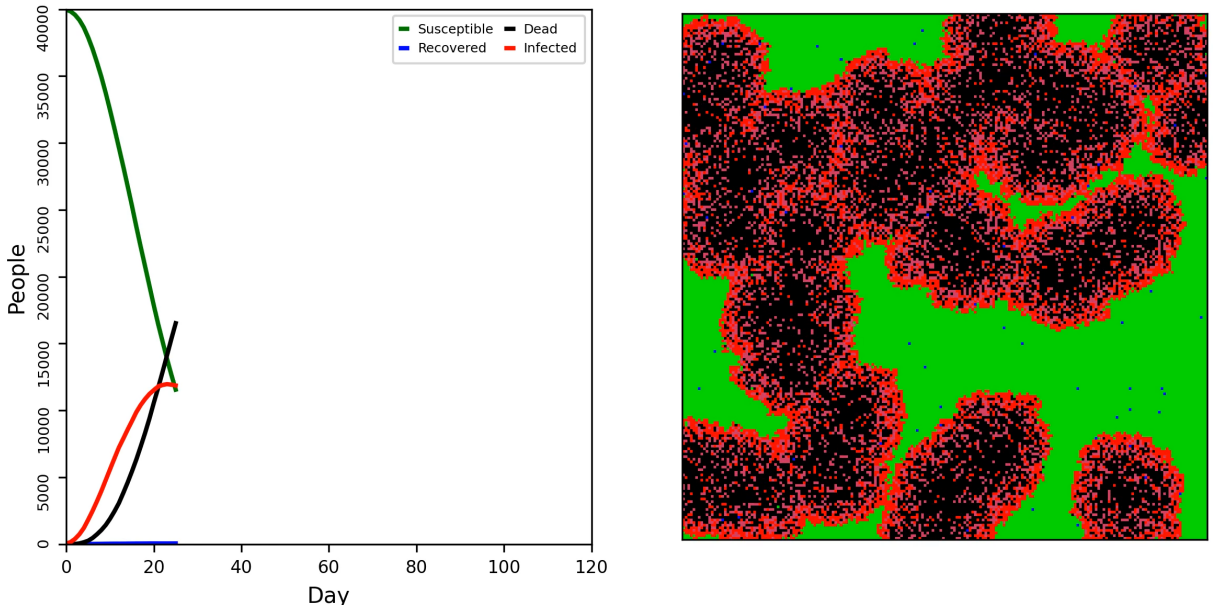


Figure 11: *'Gradual Death' Animation*

Here the death rate is somewhat low (0.1), but the recovery rate is 0, resulting in a more gradual rate of death.

Rapid Spread and Death Rates (`--sim=105`)

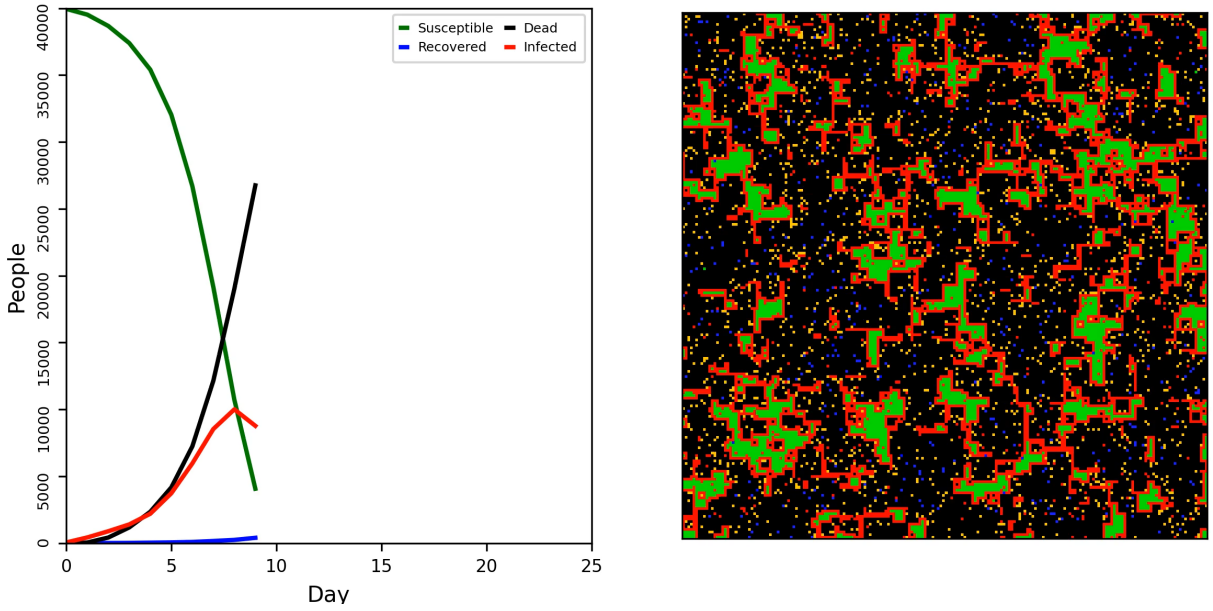


Figure 12: *'Instant Death' Animation*

The infection and death rates here are as high as possible, causing a very rapid decline of the entire population. Note the 'boxiness' of the pattern, which is a quirk of using a grid to determine neighbouring nodes.

Rapid Spread, Recovery and Loss of Immunity (`--sim=106`)

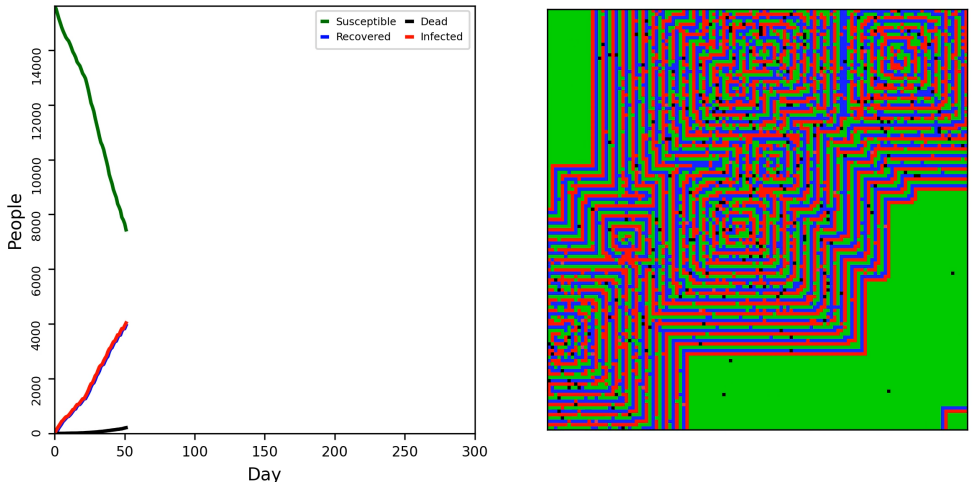


Figure 13: *'Rapid Turnaround' Animation, WARNING: Flashing Imagery*

There may not be any diseases which have these characteristics, but it yields unusual behaviour for sure. Due to the instant recovery, there is a 'protective barrier' on the inner radius of the infected ring, causing the spread to occur radially outward, initially. Over time, the rings superpose, and the pattern of spread becomes very noisy.

2.3 'Person Tracking' Model

Here we have an example output using the 'Person Tracking' model in `simulation2.py` (this file can be run directly with bash). This is much more computationally heavy, so is optimal for a 20x20 grid size. Though incomplete, this model has the potential to track unique probabilities for each person using a `Person` class, depending on factors like age and how long they've been infected, meaning their probability of recovery and death etc. can change over time.

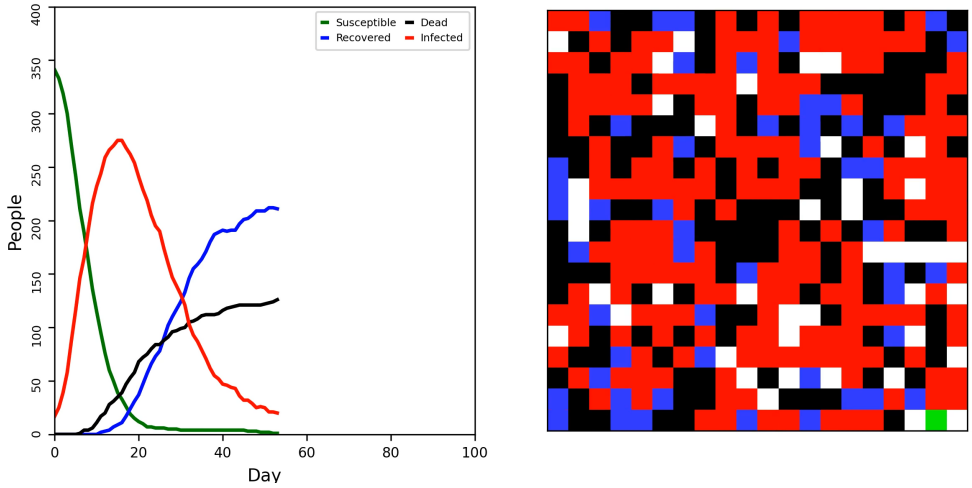


Figure 14: *'Person Tracking' Animation*

References

[1] COVID-19 Death Data, Our World in Data, updated daily, https://ourworldindata.org/covid-deaths
Date Accessed 01/05/21

[2] 'Coronavirus Worldometer', updated hourly, https://www.worldometers.info/coronavirus/
Date Accessed 01/05/21

[3] Coronavirus Disease Situation Report, World Health Organisation, 26/03/20, https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200306-sitrep-46-covid-19.pdf?sfvrsn=96b04adf4#
Date Accessed 08/05/21