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 introduce a 'reinfection' factor, representing the potential loss of immunity to COVID over time, but also the rise in variants which may be resistant to antibodies created due to 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 potentially vary. This is much like in real life, where outcomes can be unpredictable, given the same set of starting conditions as tiny changes can have large consequences.

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, but can also be used with `ANI` and `MAIN`.

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 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 `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 an option 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 (they require different 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. A full list of user inputs can be found in `README.md`.

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`)

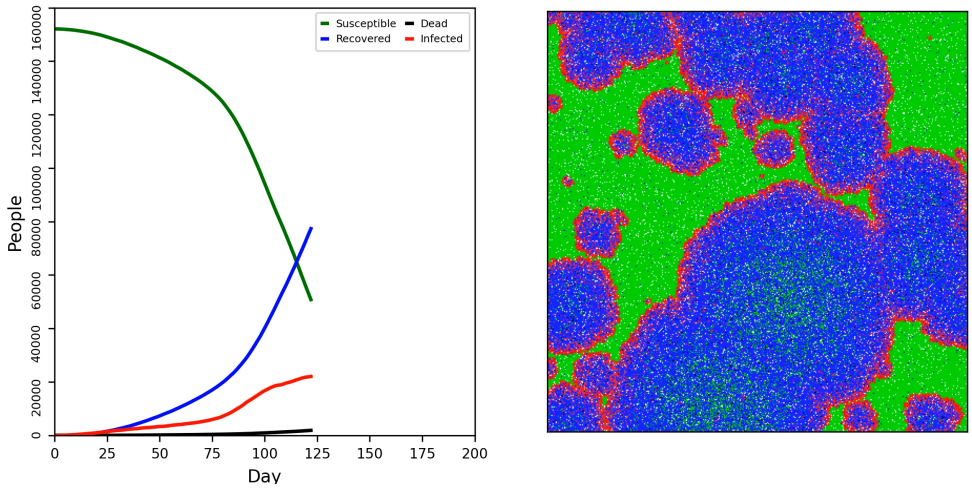
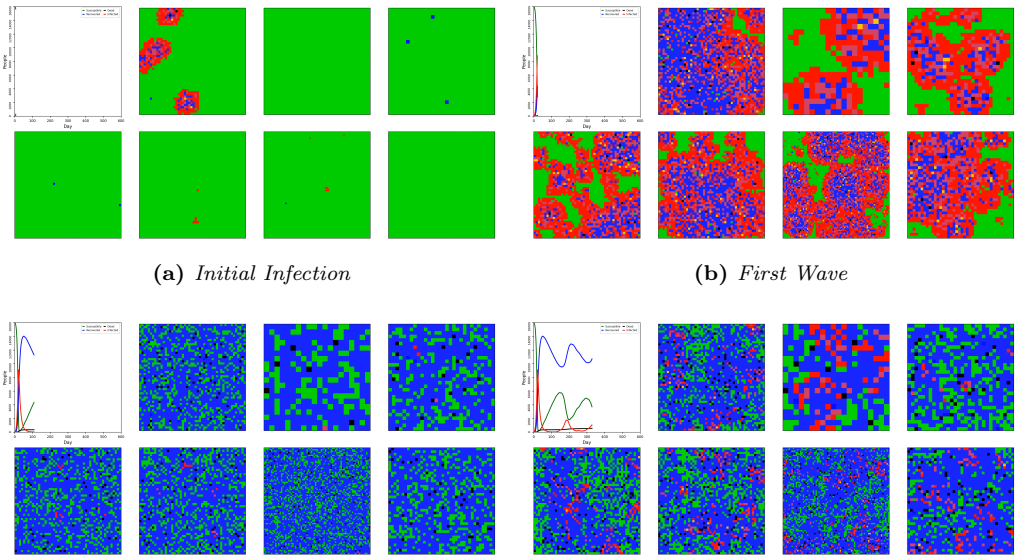


Figure 1: *Standard Animation*

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`)



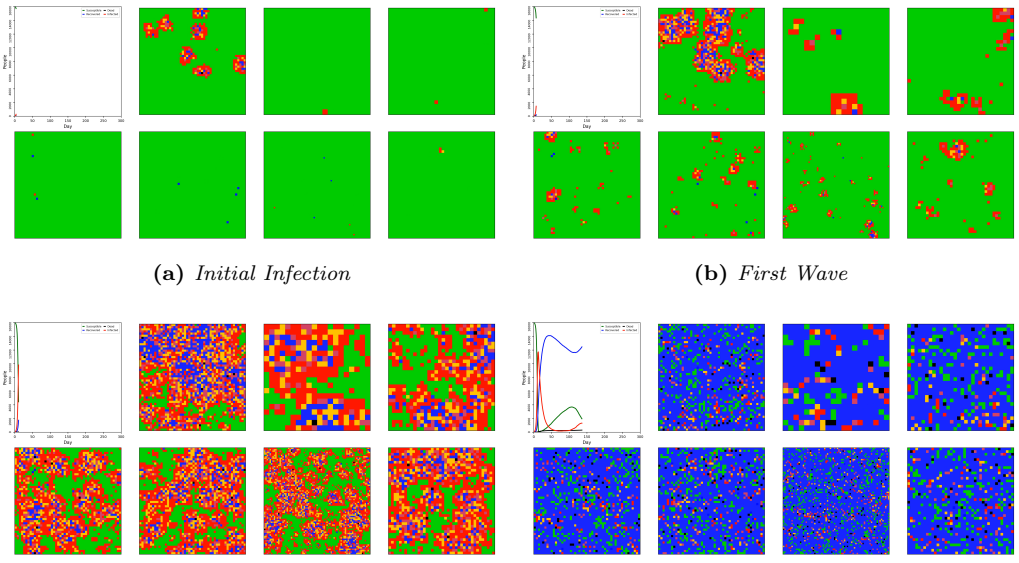
(a) *Initial Infection* **(b)** *First Wave*

(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.
Travelling between tiers during December 2020 in the UK was prohibited, the importance of this is proven through this demonstration showing cities with large amounts of travelling with the infected community which highly relates to COVID-19 as 80% of cases present with mild symptoms or asymptomatic(https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200306-sitrep-46-covid-19.pdf?sfvrsn=96b04adf4#). In this simulation the probability of a person travelling while infected is 0.3.

Compared to the previous simulation this shows the significance of travel restrictions between cities and hence why the tier system was introduced to the UK.

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

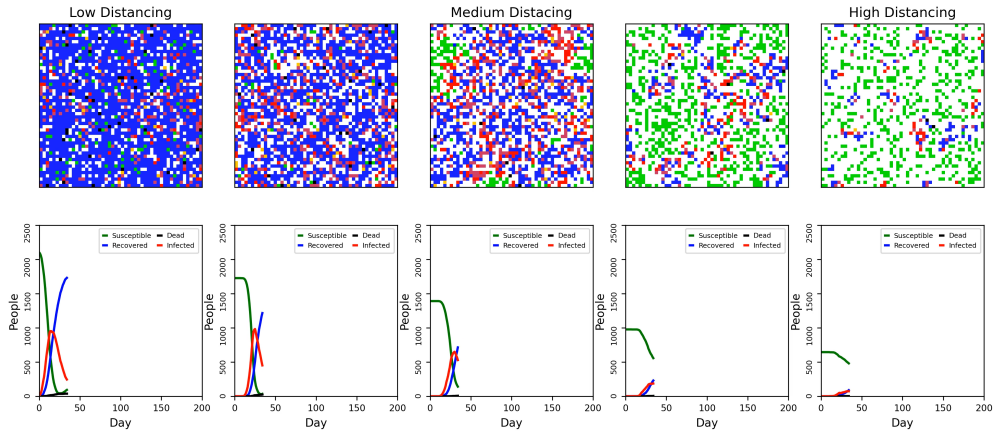
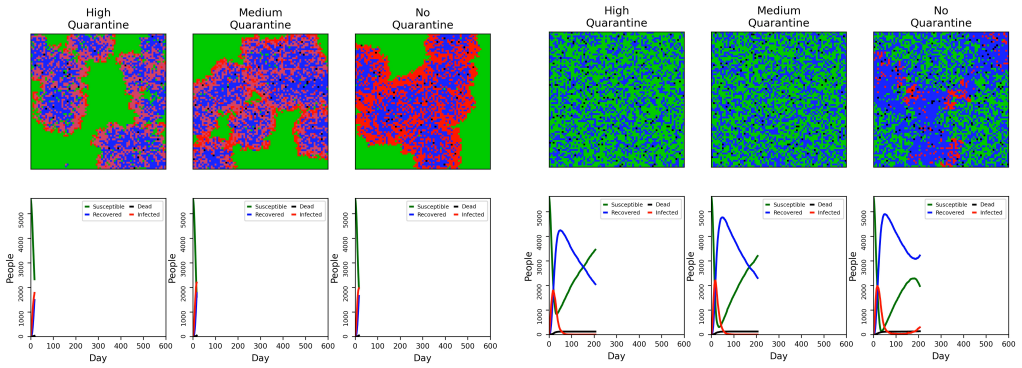


Figure 4: *Social Distancing Animation*

Looking around the world during the pandemic its clear to see that not every country has the same restrictions and more often than not we see countries get taken over by COVID-19 due to their restrictions being too light and social distancing not being enforced. The program shows us 5 cities, all with the same travel probability of 0.1 however with different social distancing levels ranging from 0.15 to 0.75 empty space in the grid.

The simulation shows us how the peak of infections varies as social distancing increases, one of the instructions given during lockdown by the government was that we should 'flatten the curve' this animation has shown us that increasing social distancing prevents the spread of coronavirus and hence flattens the curve compared to no social distancing that peaked far higher.

Quarantine (`--sim=4`)

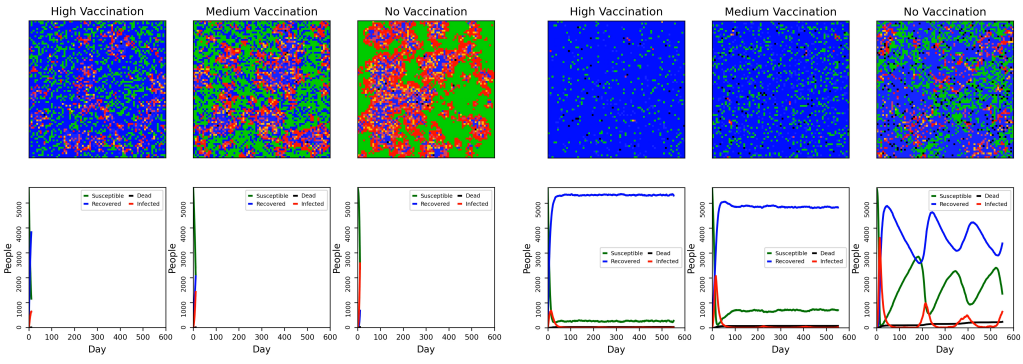


(a) *Initial Infection* **(b)** *Second Wave*

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) and finally no quarantine at all.

Vaccination (`--sim=5`)



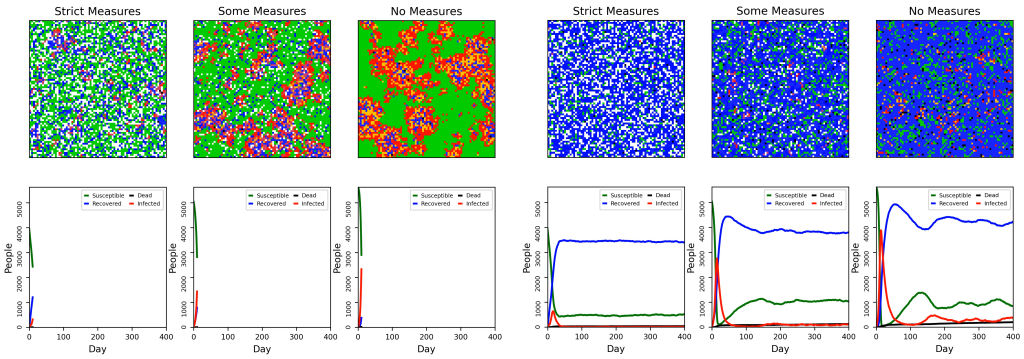
(a) *First Wave* **(b)** *Fourth Wave*

Figure 6: *Vaccine Animation*

Here we see vaccines play a large role in preventing second waves. As immunity begins to wear off, or resistant strains emerge, vaccines prevent 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*

This simulation contains three cities, the first having probability of travel, quarantine and social distancing as 0, 0.95, 0.3 respectively. the second having the probabilities as 0.1, 0.3, 0.1 and the final city with no restrictions has its probabilities as 0.3, 0, 0.

The Simulation shows that no restrictions allows for the virus to spread far more quickly and again shows it continuing peaks as peoples immunity fall away.

Death Rates (`--sim=7`)

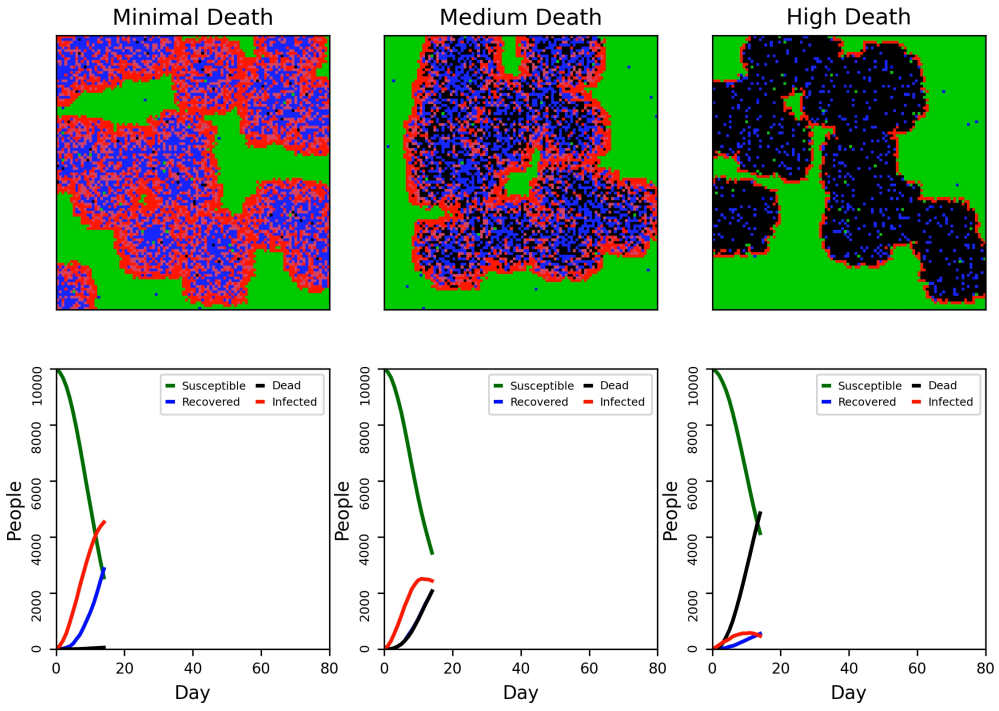


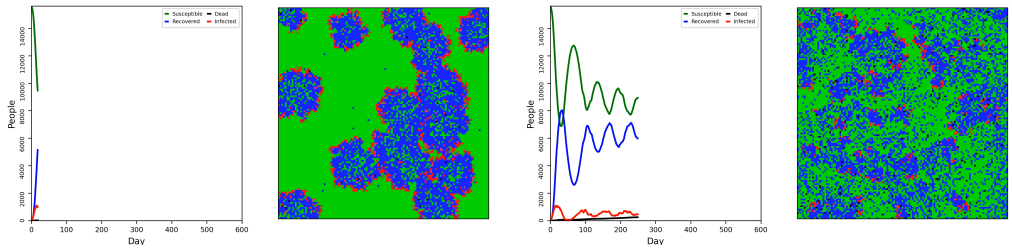
Figure 8: *Death Rates Animation*

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, and may represent other kinds of diseases.

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

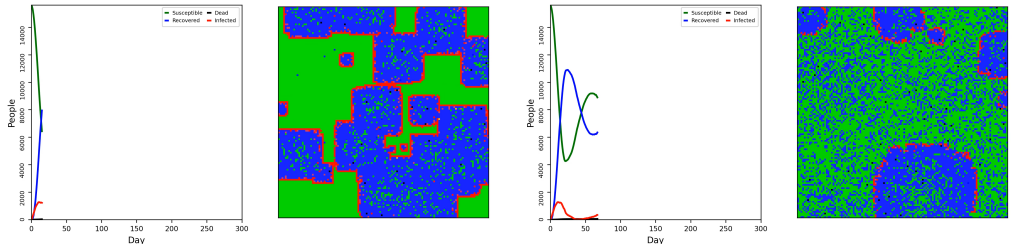


(a) *Initial Wave* **(b)** *Circulation Later On*

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

In these simulations, the infection rates appear to oscillate about an average, with the virus continuously circulating throughout the grid, rather than appearing in large bursts. The 'high recovery' version is very similar to the former, just appearing more 'sparse' on the grid. This could represent something more like the common cold, which circulates and mutates on a yearly basis.

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 every so often from a small number of areas, rather than spreading more chaotically.

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

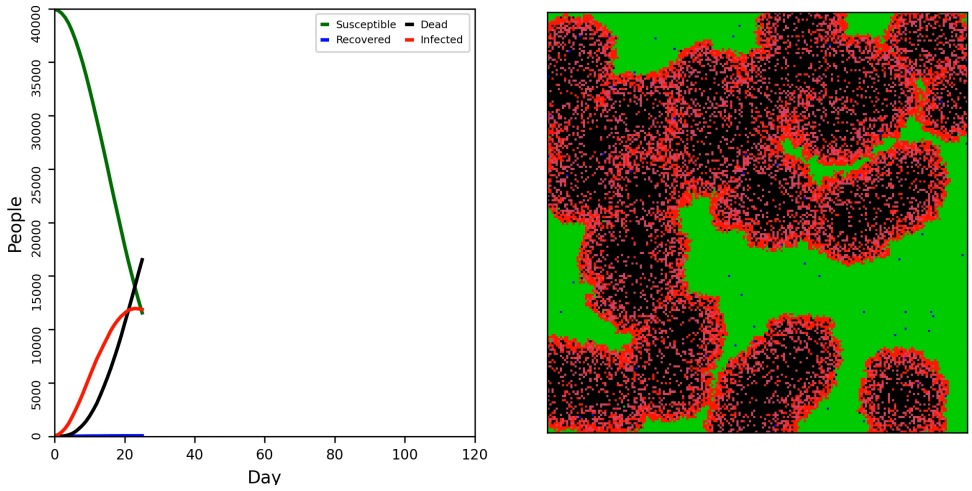


Figure 11: *'Gradual Death' Animation*

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`)

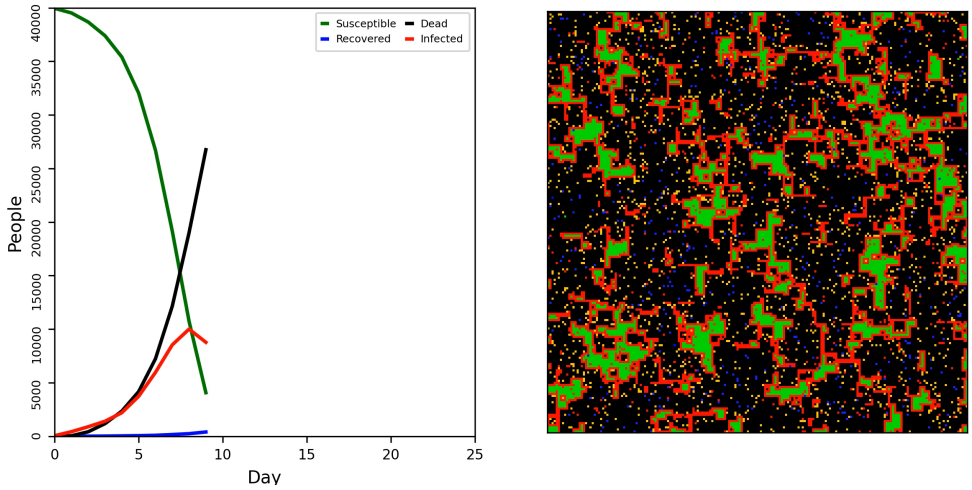


Figure 12: *'Instant Death' Animation*

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`)

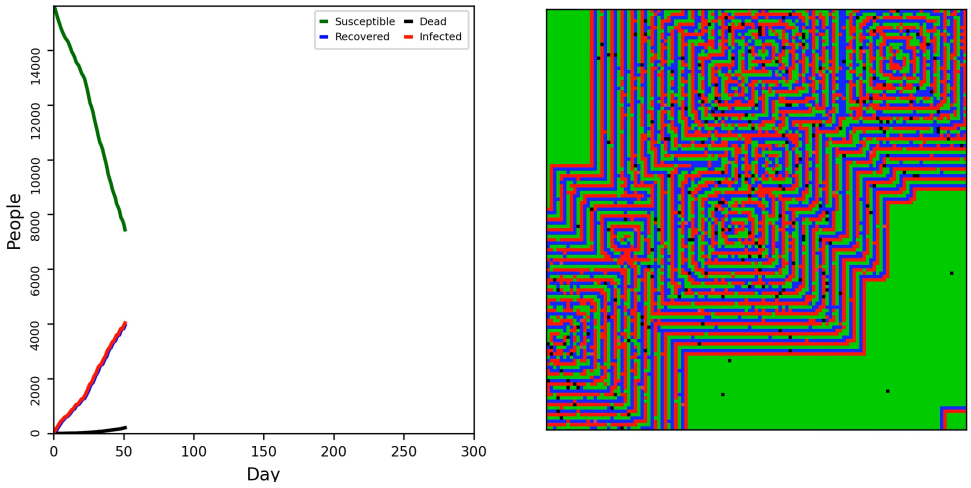


Figure 13: *'Rapid Turnaround' Animation, WARNING: Flashing Imagery*

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.