Computational Physics: Simulating a Pandemic

Devansh Matham

For my final project, I simulate a pandemic by making use of classes and methods within python. This allows me to mimic a real-world scenario where 'people' objects with specific tuneable attributes can move around a confined space, interacting with one another while spreading a disease. I track various states that people are in, create a real time animation of their interactions, and collect that data to understand the spreading dynamics. I also implement a masking strategy to mimic a real-life precautionary measure in order to understand the extent that the spread of the disease can be suppressed.

1 Introduction

Most recently, COVID-19 has shown the world how a disease can quickly disrupt our everyday lives. As many countries - including first world economies - grappled with spiking cases, others that took the disease more seriously from the beginning were able to limit infections. Understanding their spread and the way in which our interactions (or lack thereof through measures such as social distancing) plays a role in their impact on societies can make us better prepared to take on future diseases of similar scales. By recreating a disease with tuneable variables and scenarios that attempt to imitate human interaction, we can also get a visual of just how many parameters go into assessing their origin, as well as their many impacts.

2 Method

As part of this project, the 'pythonic' approach to simulating our disease and its spread was to create a 'person' object across a grid. For a person,

I assumed some initial attributes in the form of binary states - they would be alive, not infected, not immune, and not masked. I then created methods within my 'person' class, detailing conditions such as under what circumstances they might die, become contagious, begin a recovery period and gain immunity if not fated to die, and etc. A crucial part of determining who dies in my population is based on the fact that the disease has a certain chance to kill over the course of the infection, not at every timestep. In the method named 'infection,' I check for ONLY when a person is infected, whether or not they will die by comparing a random number to our mortality rate. If True, I then assign a random time in their recovery window (rather than at the beginning or end of the window - that would be rather boring and unrealistic).

Some important variables that heavily influenced my simulation were the infectious rate and the mortality rate of the disease. For this project I set the infectious rate to 40% and the mortality rate to 10%.

```
def infection(self):
    if not self.immune and self.alive:
        self.infected = True
        self.fate = np.random.random() < mortality #checking their fate
        if self.fate: #if the person is fated to die...
            self.time_of_death = np.random.randint(1, recovery_time) #ra
        else: #if the person is NOT fated to die...
            self.time_of_death = None
        return True
    return False
```

Figure 1: The infection method within our person class.

Initially, I had a for loop create n x n instances of the person class based on the grid size n. This meant that if the grid size was 6 (6 units in height and 6 units in width), 36 person would be evenly spread out across the grid with a spacing of 1 unit, before the timestep advances and they move out in varying directions.

2.1 Time

An important decision to make for my simulation was the time component. For the project, advancing our time steps in terms of days would not allow us to see the many interactions that happen within a day. Since each timestep would move a person according to their velocity - an attribute assigned to them based on sampling from a standard normal distribution - I assigned a recovery period for a person who was infected to be 100 time steps from when they first came in contact with the disease, out of the total 500 time steps that goes into running the entire simulation.

2.2 Interaction Between Two People

When I initially made the handle_collisions method, it was programmed to swap velocities to simulate elastic collisions (like billiard balls bouncing off each other). They would bounce off each other in a similar manner to how I handle when a person bounces off the boundaries of our confined simulation grid. For human to human interaction. However, to have simple reflections of their velocities after coming in contact seemed rather unnatural. People don't typically reverse directions completely when they get too close; instead, they might slightly adjust their trajectory to avoid each other, such as shifting their path by a small angle to move past one another. To address this, I had the handle_collisions method to perturb the movement angles of colliding individuals slightly when they come within the collision distance that we set (0.2 units), rather than swapping velocities. This would make the simulation more realistic by mimicking how people sidestep or adjust their paths in a crowded situation.

```
dist = p1.proximity(p2)
if dist < collision_dist:
    #taking in the current angles
    angle1 = np.arctan2(p1.vel_y, p1.vel_x)
    angle2 = np.arctan2(p2.vel_y, p2.vel_x)

    #magnitudes of velocity
    speed1 = np.sqrt(p1.vel_x**2 + p1.vel_y**2)
    speed2 = np.sqrt(p2.vel_x**2 + p2.vel_y**2)

    #random angular perturbation (+/- 30 degrees) - anything more than this angle might look un-human
    perturbation = np.random.uniform(-np.pi/6, np.pi/6)
    new_angle1 = angle1 + perturbation
    new_angle2 = angle2 - perturbation  #opposite perturbation for p2

    #update velocities with new angles, preserving speed
    p1.vel_x = speed1 * np.cos(new_angle1)
    p1.vel_y = speed1 * np.sin(new_angle1)
    p2.vel_x = speed2 * np.cos(new_angle2)
    p2.vel_y = speed2 * np.sin(new_angle2)

    #resolves overlap - this moves the people apart, working in conjuction with the angle perturbation
    nx = (p2.x - p1.x) / dist #here we take the x-component of the unit vector between the two people
    ny = (p2.y - p1.y) / dist #and the y-component as well
    overlap = (collision_dist - dist) / 2 #splits it equally
    p1.x -= nx * overlap
    p1.y -= ny * overlap
    p2.x += nx * overlap
    p2.y += ny * overlap
```

Figure 2: The infection method wiithin our person class.

2.3 Precautions Against Spread

After understanding how the disease proliferates among a population that freely roams around the confined space, the project sought to understand to what extent preventative measures could lower overall infection and death rates.

For each person in my simulation, I assigned them with 4 output states: Healthy, Infected, Immune, and Dead. Healthy meant that the individual has never come in contact with the disease. Infected refers to those who have come in contact with it and are in the process of being fated to either live or die, and immune are those who were previously infected, but ended up not dying. Of course, Dead are those who succumbed to the ill-fate of the disease. These 4 states were tracked at ever timestep for both the masked and unmasked scenarios using a histogram that updated in real time. It turns out that in most cases, there were very to no people left in the healthy count. However, this differed slightly in the case of the masked simulation, as it took more time for everyone to be infected. A snapshot of the animation and histogram side-by-side at a

late stage in the 500-step simulation can be seen in Fig. 7 and 8.

3 Results

Below are the results of my pandemic simulation. In Fig 3, I compare how the infections changed over time between the masked and unmasked situation. Fig. 4 is similar in that it compares death count instead. I also include a plot that compares computational runtime against the size of the population that my simulation is working with. The graph appeared to get slower as the size increased, and based off the tradeoff between wanting to see a crowded space as well as run time, I chose 15x15 = 225 people as my population size for the simulation results in Fig. 3 and 4.

3.1 Plots

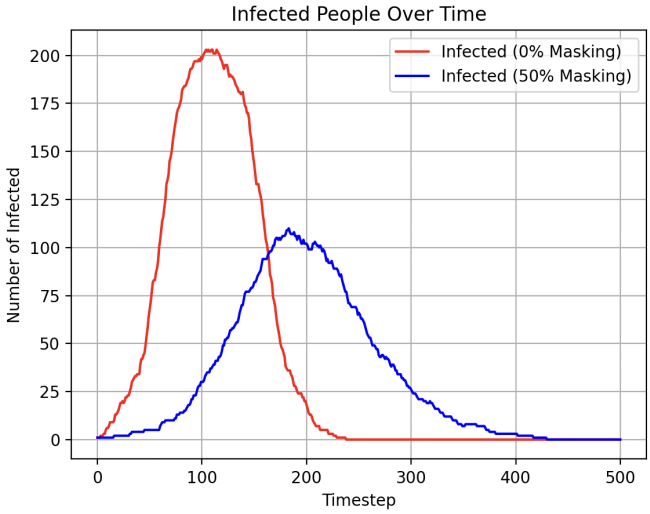

Figure 3: Number of infected people plotted against the simulation timestep - 10% mortality, 40% infectious rate, 15x15 = 225 people.

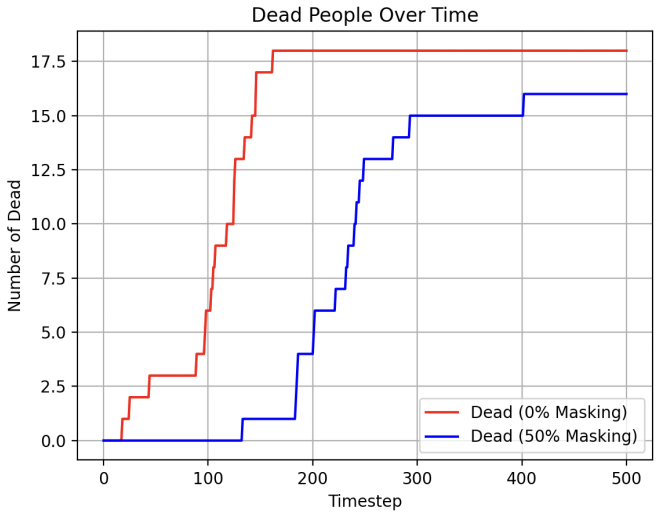

Figure 4: Number of deaths plotted against the simulation timestep - 10% mortality, 40% infectious rate, 15x15 = 225 people.

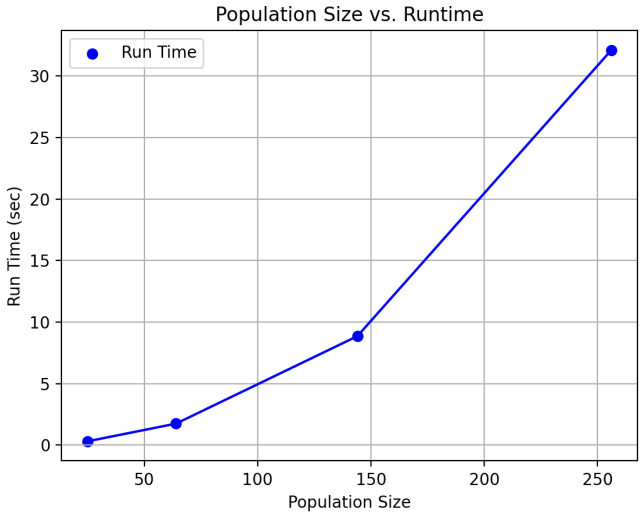

Figure 5: Four grid sizes (5x5, 8x8, 12x12, 16x16) with their run times. Their respective population sizes are 25, 64, 144, 256 people.

3.2 Animation

I decided for my output to have the masked simulation and the unmasked simulation running at the same time, in order to be able to compare both of

```
Animation computation completed in 0.30 seconds
Animation computation completed in 1.74 seconds
Animation computation completed in 8.85 seconds
Animation computation completed in 32.11 seconds
```

Figure 6: The run time output for population sizes of 25, 64, 144, 256 people.

them visually. I also thought to add a histogram next their simulations to understand

A decision I made for the animation was to keep the dead poeple (black dots) stationary on the grid at the position where they died. I ensured that they simply remained there without interacting (physically and in terms of spreading the disease) with another person, so that I could see just how many black dots accmulate over the 500 timesteps.

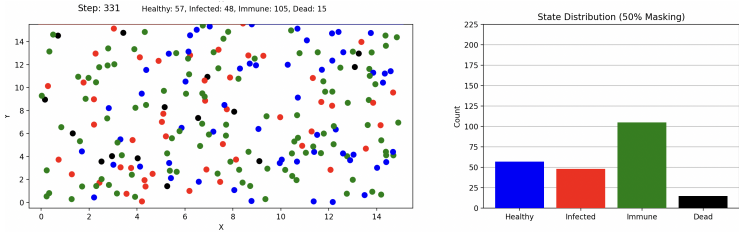

Figure 7: A side-by-side visual of the animation and histogram keeping track of the 4 states in real time. This is for the population where 50% are masked.

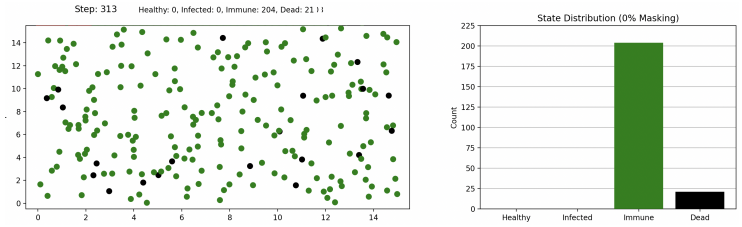

Figure 8: A side-by-side visual of the animation and histogram keeping track of the 4 states in real time. This is for the population where no one is wearing a mask. While there are more poeple who are now immune than in Fig 7, it still means that more people had to be infected and hospitalized. We can also see a higher death rate compared to Fig 7.

3.3 Anomalies

When running my simulation, sometimes there was a chance that the masked population had a higher death and infected rate than the unmasked population, as seen in Fig 7 and 8. Since the nature of our simulation includes some level of randomness, we should expect to have a non-zero, but minuscule chance of this occurring. For example, it is possible that the first infected person was immediately surrounded by those who were not masked and this could lead to higher infected rates than the unmasked scenario if all those who were masked were clumped initially in the opposite corner, unable to play a role in suppressing a spread early on.

I noticed that the bigger the initial population size, the chances of this anomaly from occurring was greatly reduced, helping us reach our goal of showing that the 50 % masking simulation consistently shows lower infections and deaths (or at least a very high probability of doing so) to reflect the protective effect of masks. This anomaly also reveals the somewhat random things in the real world, such as where people move and who they (intentionally or unintentionally) interact with can be a contributing factor in helping precautionary measures make an effective impact.

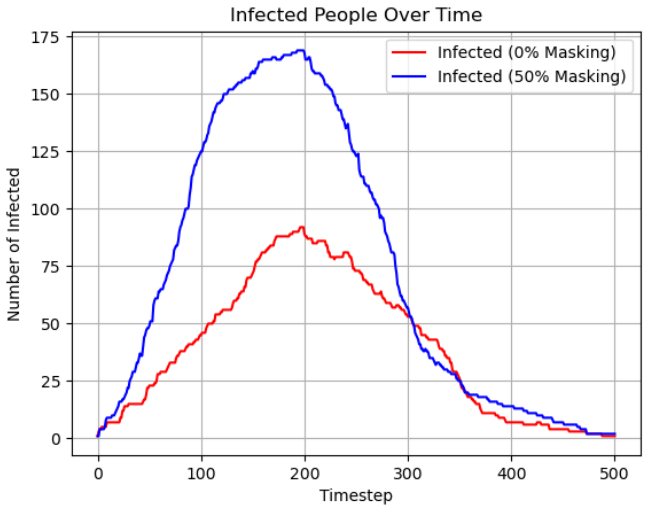

Figure 9: A rare and unexpected result where the masking situation resulted in more infections than the unmasked.

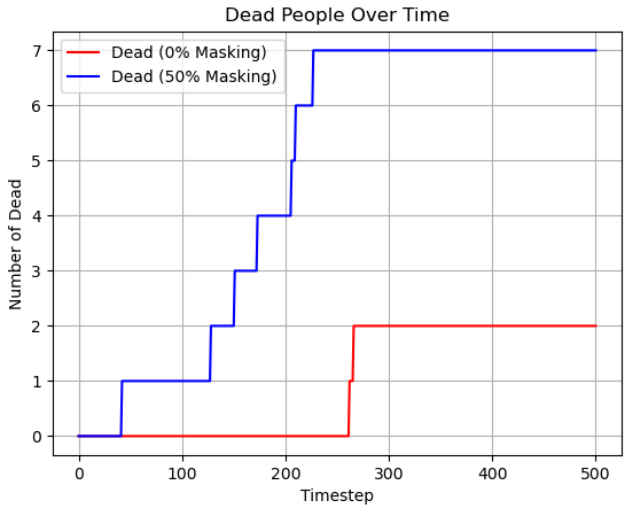

Figure 10: A rare and unexpected result where the masking situation resulted in more deaths than the unmasked.

4 Discussion

When assessing the graphs and their results, we are able to notice a defining trend. In both Fig. 3 and Fig. 4, the maxima of the masked plotted points

is less than the unmasked graph - for Fig. 3, it is on the order of half the population that is not exposed to the disease when half of the population wears a mask.

Additionally, we can also notice that there is a delayed response in both deaths and infection rates when 50% of the population is masked, i.e, the disease is slowed down. However, for infections, Fig 3 also reveals that the infection numbers takes longer to reach back to the initial level compared to the unmasked situation.

At the height of the COVID-19 pandemic, scenes of hospitals in major cities around the world were overwhelmed by the magnitude of infected cases. Often having to deny those who required immediate medical attention due to limited resources, preventative measures play a key role in allowing access to provide aid to those who require it in desperate times. A further outlook to this project would be to implement other preventative measures, such as social distancing, or perhaps even investigating different types of masks and comparing their effectiveness against a disease.