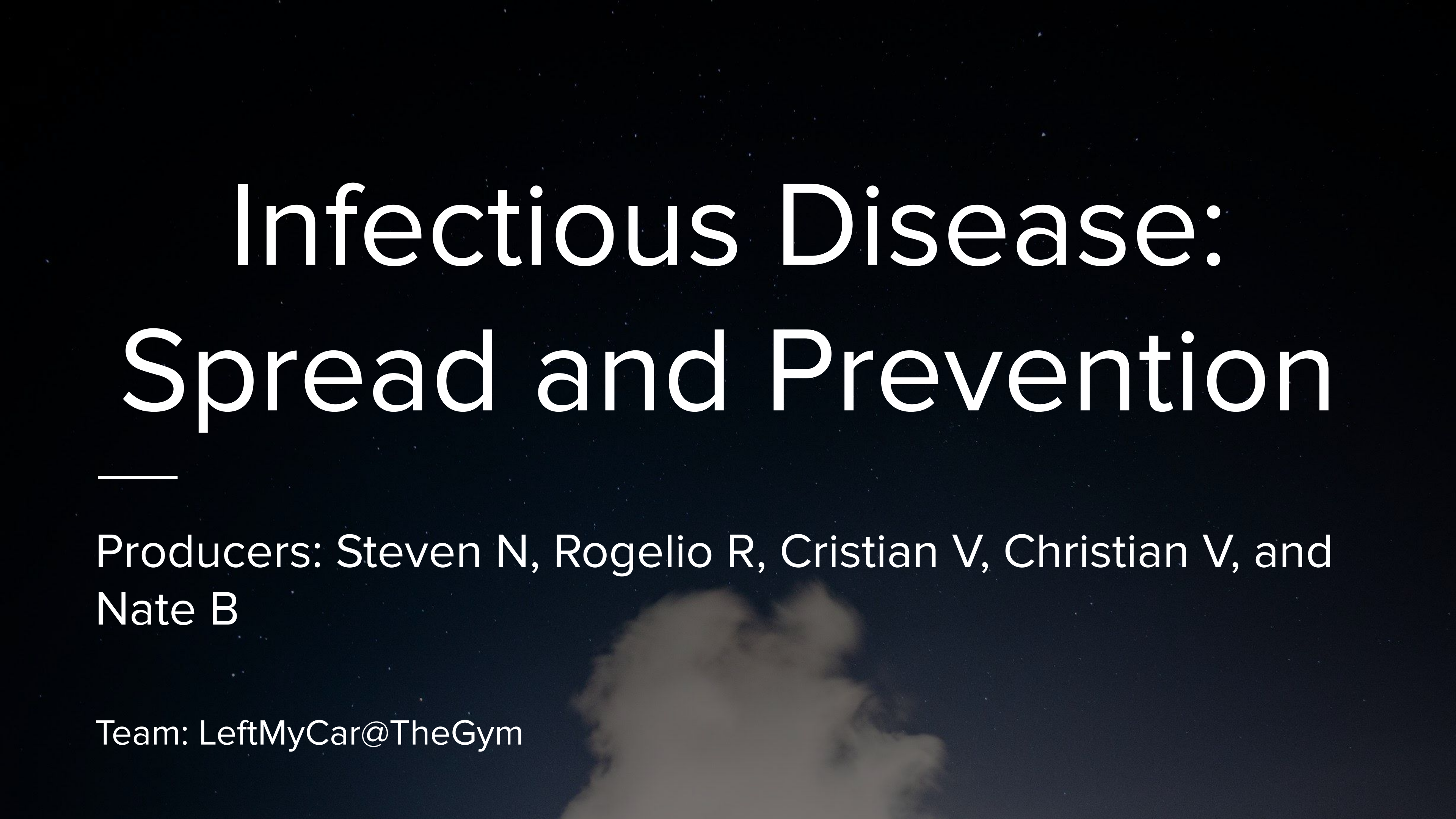
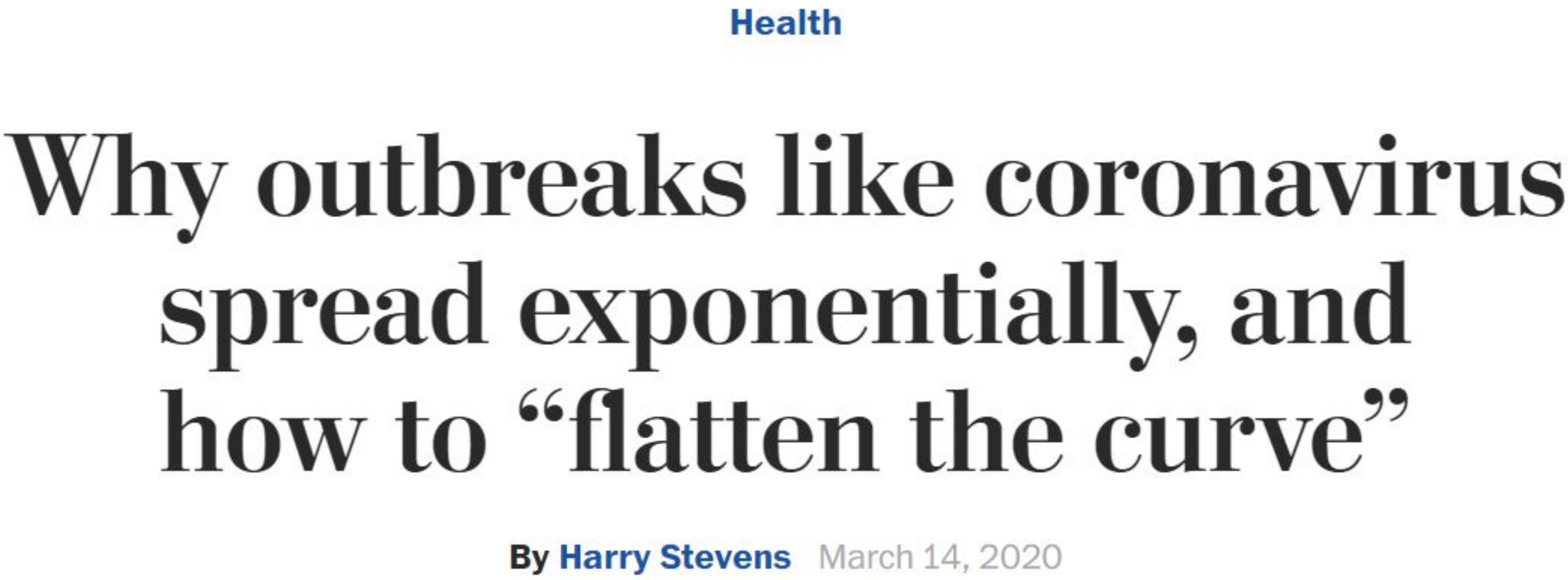


Infectious Disease: Spread and Prevention

Producers: Steven N, Rogelio R, Cristian V, Christian V, and Nate B

Team: LeftMyCar@TheGym

Which Article?



Health

Why outbreaks like coronavirus spread exponentially, and how to “flatten the curve”

By Harry Stevens March 14, 2020

Why this Article?

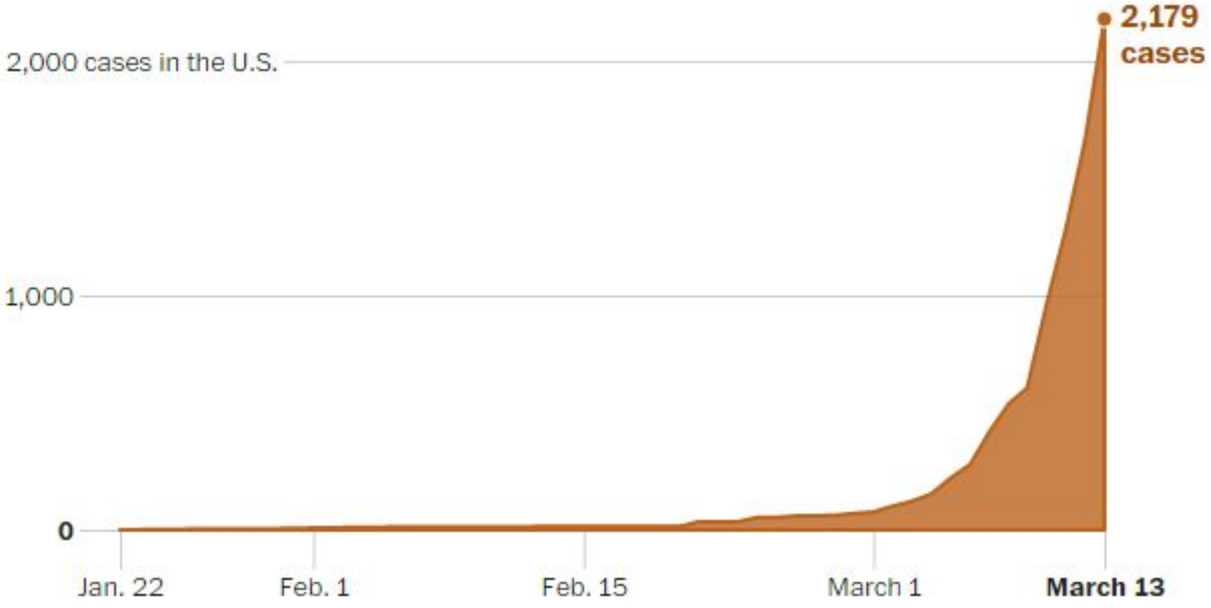
- Easy to navigate the topic
- Intuitive infectious disease simulation presented on fictitious disease, “Simulitis”
- Article’s simulations were representative of 4 likely scenarios
- Real life constraints (social distancing) on population flow that present possible real solutions

Simulation Techniques

- White box Model
- Dynamic Model (varies with time)
- Stochastic (random actions)
- Discrete State, Continuous time Simulation.

Exponential Growth in Cases of COVID-19

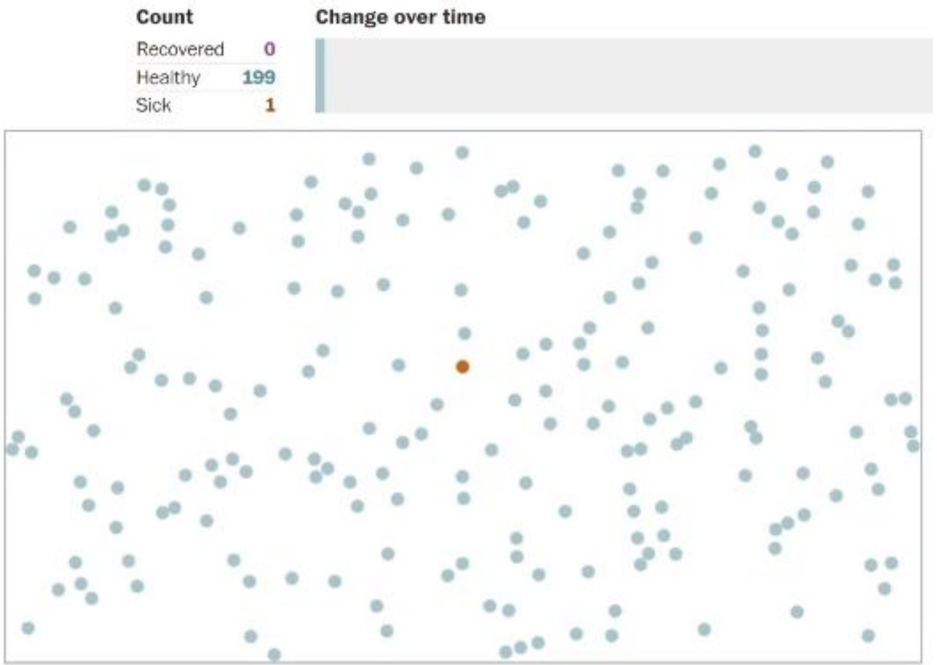
Exponential curve based on doubling of cases every 3 days or so.



Article Simulation (No constraints)

@t = 0
Pop = 200
Sick = 1

Sample is small so we could see the curve steepen greatly, leading to a much larger number infected in larger cities

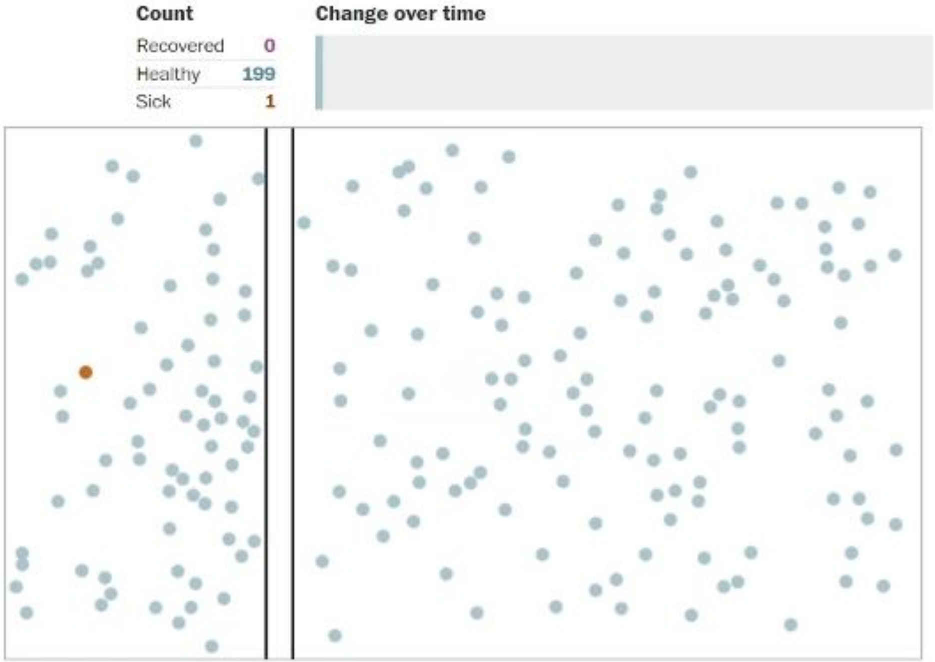


Count

Recovered	0
Healthy	199
Sick	1

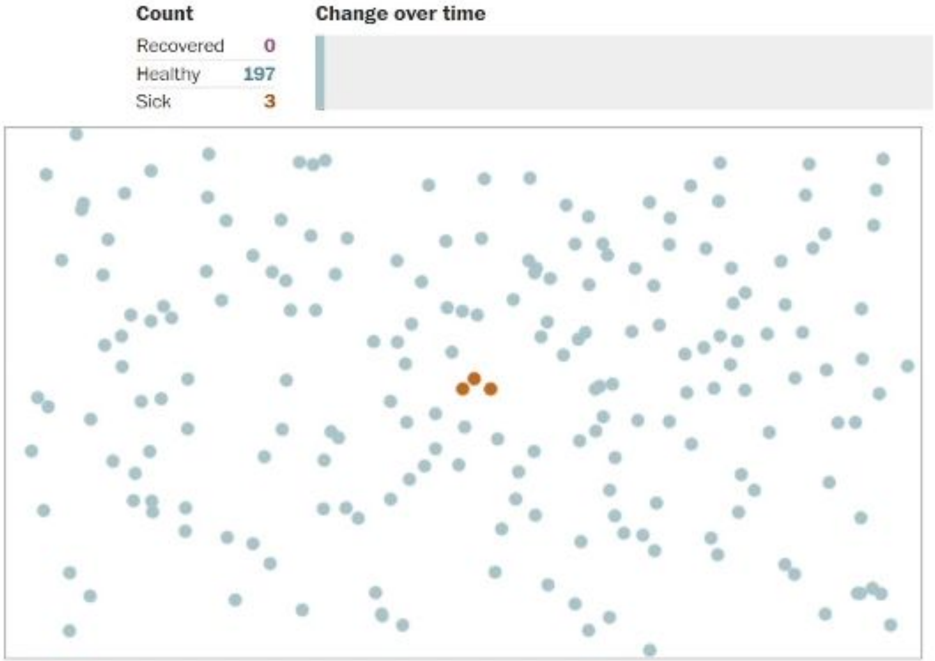
Attempted Quarantine

Attempting to force Quarantine will work initially, but people will stop complying with time.



Moderate Social Distancing

Constraint added: 75% of the population practices social distancing

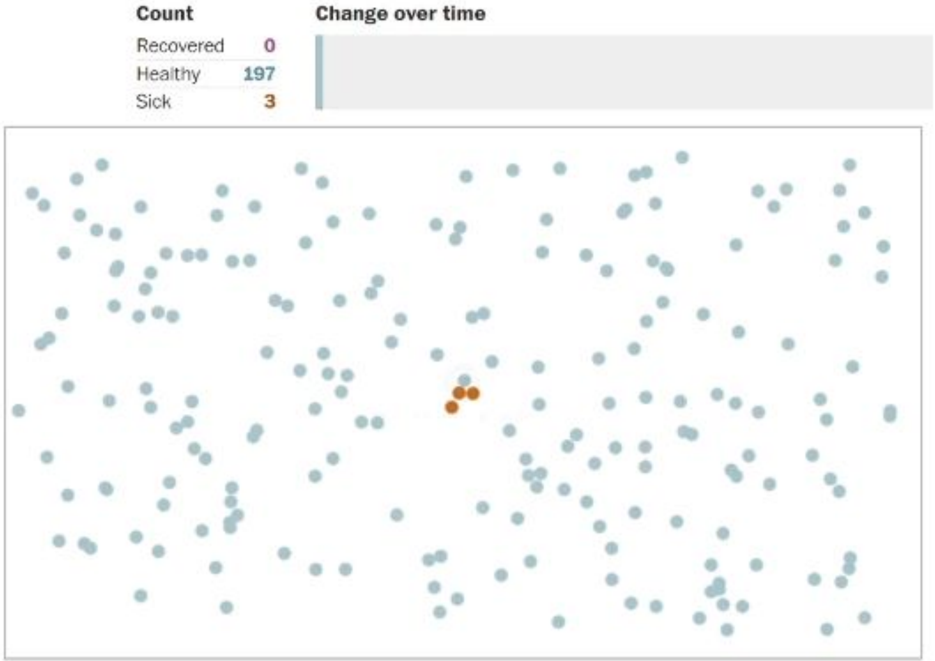


Count	
Recovered	0
Healthy	197
Sick	3

Change over time

Extreme Social Distancing

Constraint tightened: 87.5% of population practices social distancing methods that equate to current “shelter in place” methods.



The Challenge of Simulation

Accurately modeling real world phenomena while putting in place statistically accurate constraints.

Simulation is like analogy, imperfect but it gets the point across.

Agent-based simulation of a hypothetical disease spread virus

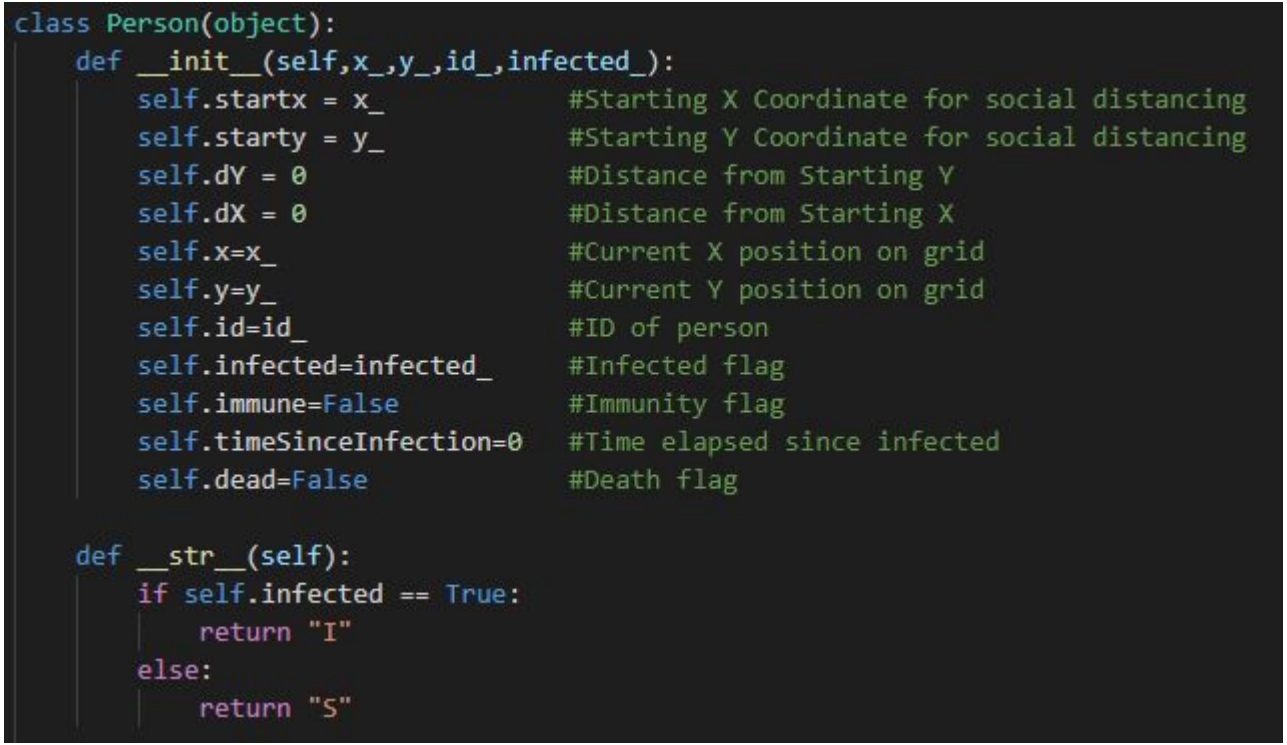
- Four parts to this simulation
- 100x100 grid with 200 people
- Simulation repeated 1, 10, 100, and 1000 times to show averages and outliers in data.
- People randomly placed in grid
- 1 infected at t=0.

Python Libraries Used

- Random
- Pandas
- Matplotlib

Code Snippet

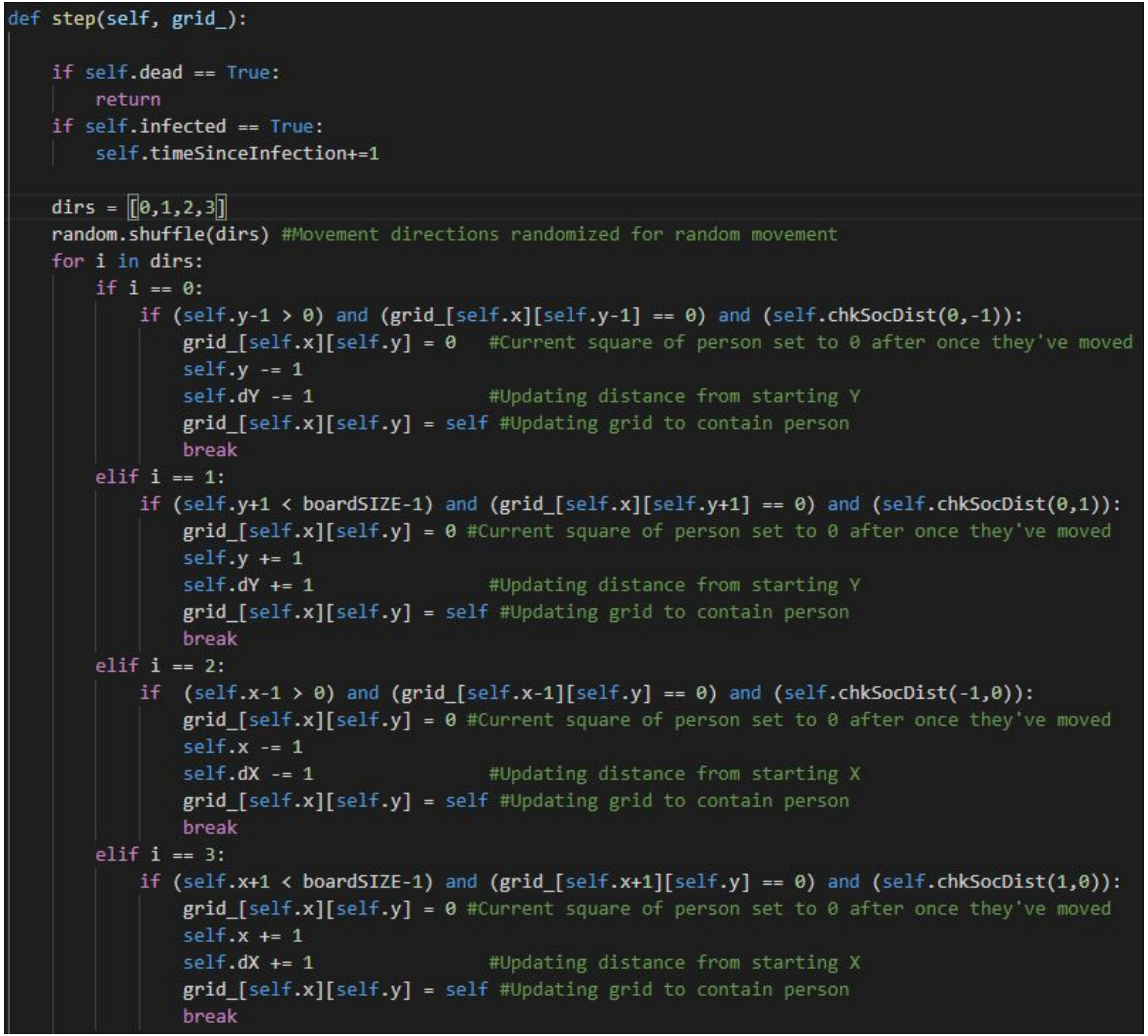
Person Class



```
class Person(object):
    def __init__(self,x_,y_,id_,infected_):
        self.startx = x_            #Starting X Coordinate for social distancing
        self.starty = y_            #Starting Y Coordinate for social distancing
        self.dY = 0                 #Distance from Starting Y
        self.dX = 0                 #Distance from Starting X
        self.x=x_                   #Current X position on grid
        self.y=y_                   #Current Y position on grid
        self.id=id_                 #ID of person
        self.infected=infected_     #Infected flag
        self.immune=False           #Immunity flag
        self.timeSinceInfection=0   #Time elapsed since infected
        self.dead=False             #Death flag

    def __str__(self):
        if self.infected == True:
            return "I"
        else:
            return "S"
```

Code Snippet

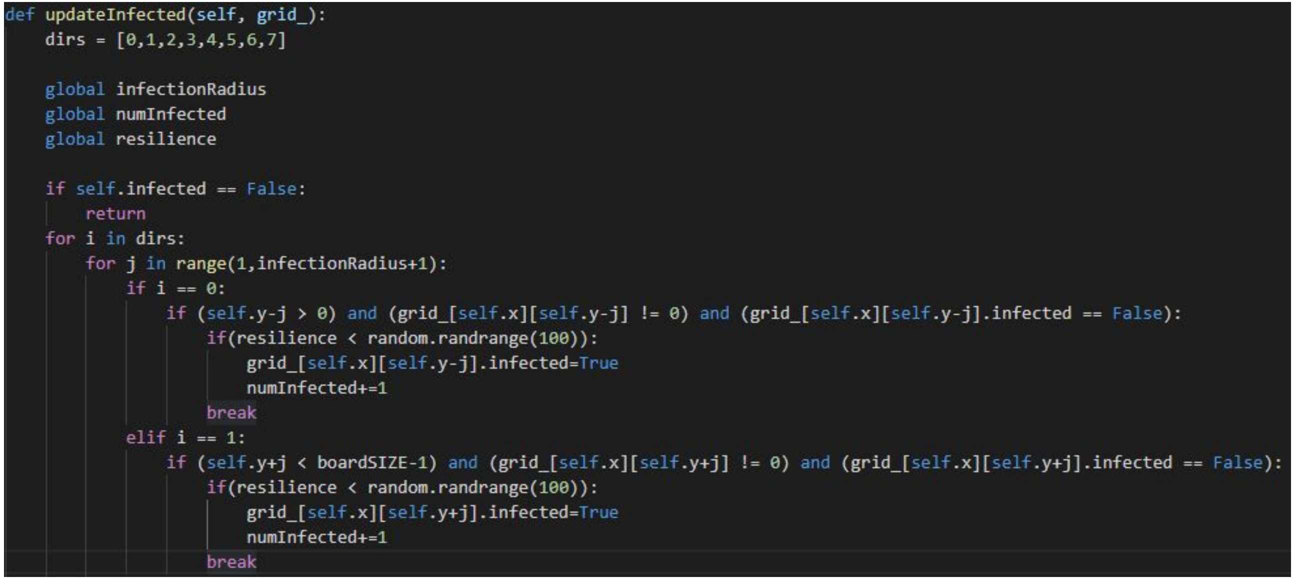
Step function



```python
def step(self, grid_):

    if self.dead == True:
        return
    if self.infected == True:
        self.timeSinceInfection+=1

    dirs = [0,1,2,3]
    random.shuffle(dirs) #Movement directions randomized for random movement
    for i in dirs:
        if i == 0:
            if (self.y-1 > 0) and (grid_[self.x][self.y-1] == 0) and (self.chkSocDist(0,-1)):
                grid_[self.x][self.y] = 0   #Current square of person set to 0 after once they've moved
                self.y -= 1
                self.dY -= 1                #Updating distance from starting Y
                grid_[self.x][self.y] = self #Updating grid to contain person
                break
        elif i == 1:
            if (self.y+1 < boardSIZE-1) and (grid_[self.x][self.y+1] == 0) and (self.chkSocDist(0,1)):
                grid_[self.x][self.y] = 0 #Current square of person set to 0 after once they've moved
                self.y += 1
                self.dY += 1                #Updating distance from starting Y
                grid_[self.x][self.y] = self #Updating grid to contain person
                break
        elif i == 2:
            if  (self.x-1 > 0) and (grid_[self.x-1][self.y] == 0) and (self.chkSocDist(-1,0)):
                grid_[self.x][self.y] = 0 #Current square of person set to 0 after once they've moved
                self.x -= 1
                self.dX -= 1                #Updating distance from starting X
                grid_[self.x][self.y] = self #Updating grid to contain person
                break
        elif i == 3:
            if (self.x+1 < boardSIZE-1) and (grid_[self.x+1][self.y] == 0) and (self.chkSocDist(1,0)):
                grid_[self.x][self.y] = 0 #Current square of person set to 0 after once they've moved
                self.x += 1
                self.dX += 1                #Updating distance from starting X
                grid_[self.x][self.y] = self #Updating grid to contain person
                break
```

Code Snippet

UpdateInfected function

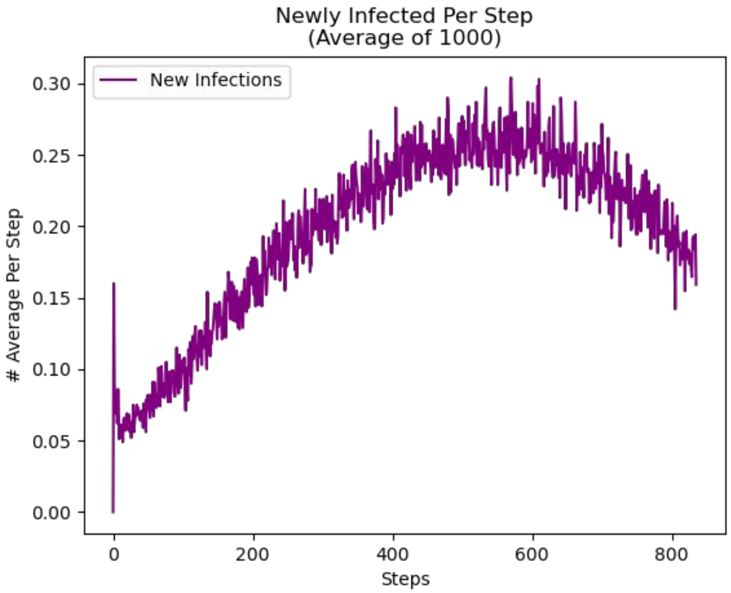


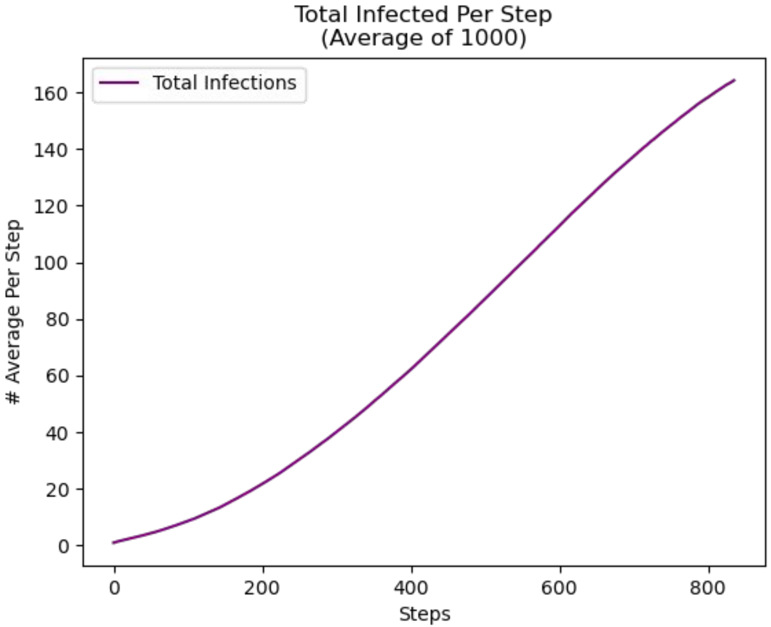
```
def updateInfected(self, grid_):
    dirs = [0,1,2,3,4,5,6,7]

    global infectionRadius
    global numInfected
    global resilience

    if self.infected == False:
        return
    for i in dirs:
        for j in range(1,infectionRadius+1):
            if i == 0:
                if (self.y-j > 0) and (grid_[self.x][self.y-j] != 0) and (grid_[self.x][self.y-j].infected == False):
                    if(resilience < random.randrange(100)):
                        grid_[self.x][self.y-j].infected=True
                        numInfected+=1
                    break
            elif i == 1:
                if (self.y+j < boardSIZE-1) and (grid_[self.x][self.y+j] != 0) and (grid_[self.x][self.y+j].infected == False):
                    if(resilience < random.randrange(100)):
                        grid_[self.x][self.y+j].infected=True
                        numInfected+=1
                    break
```

Simulation A

- Infection chance = 100%
- Infected people infect those immediately adjacent to them
- People cannot die or recover.
- Simulation runs until everyone is infected

Results of Running Sim A

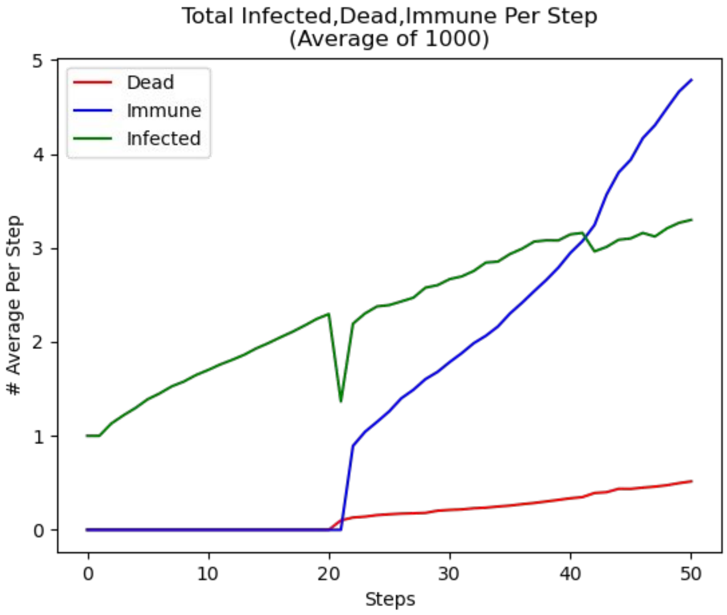


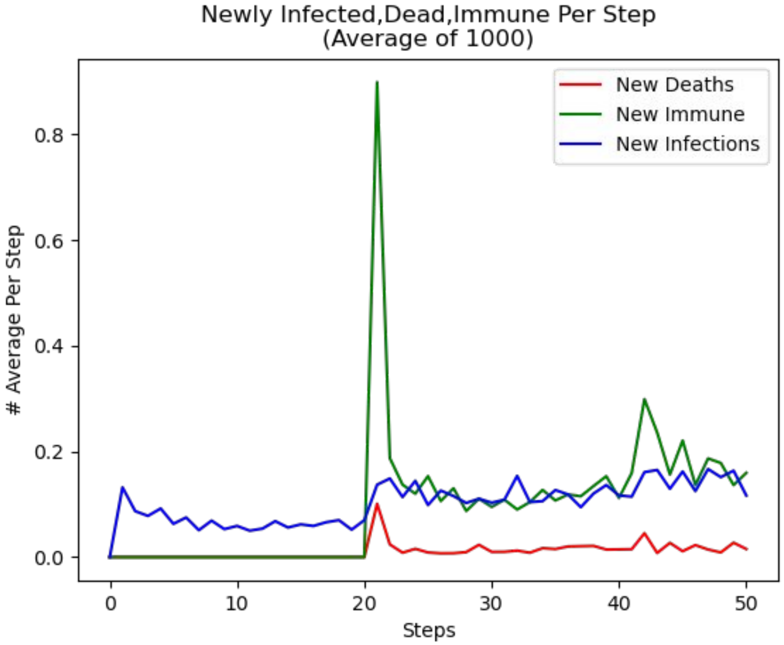


Simulation B

- Infected people now have a 90% chance to recover and a 10% chance to die after 20 time steps.
- Other parameters remain the same as Simulation A.
- Simulation runs until there are zero infected.

Results of Running Sim B

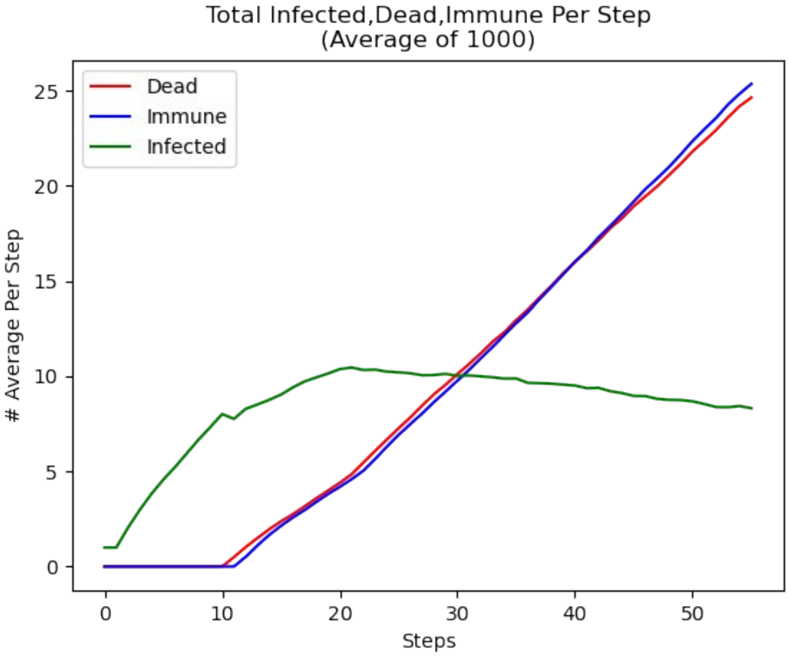


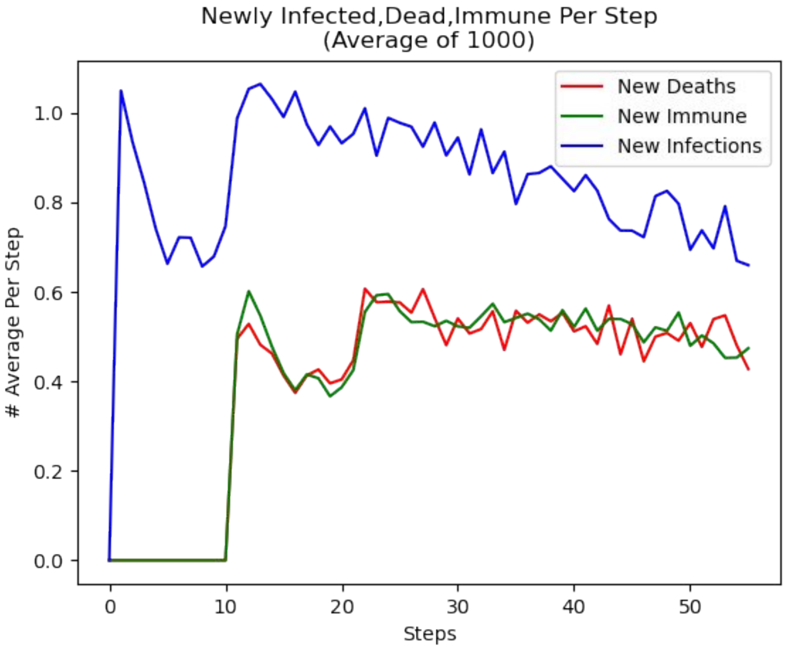


Simulation C

- Infected people now have a 50% chance to recover and a 50% chance to die after 10 time steps.
- Infection Radius = 3 instead of 1
- Grid Size 50x50 with 100 people

Results of Running Sim C

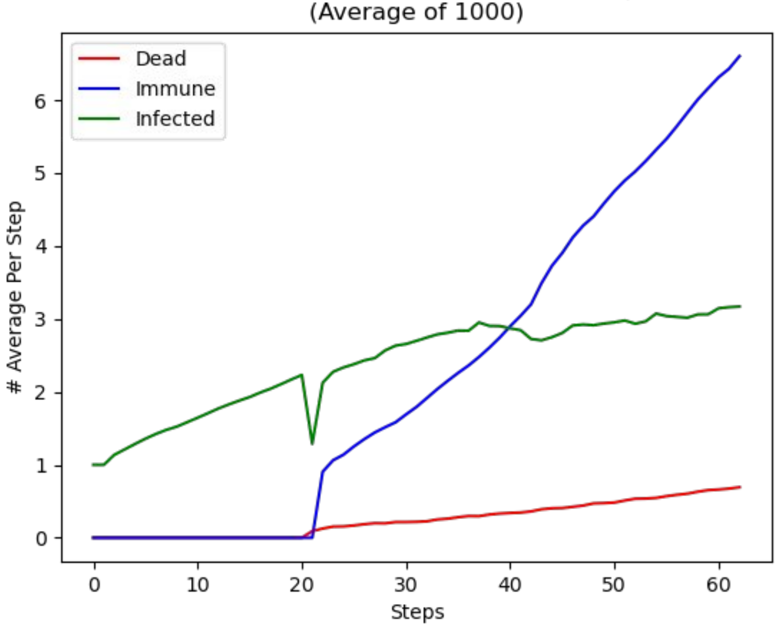




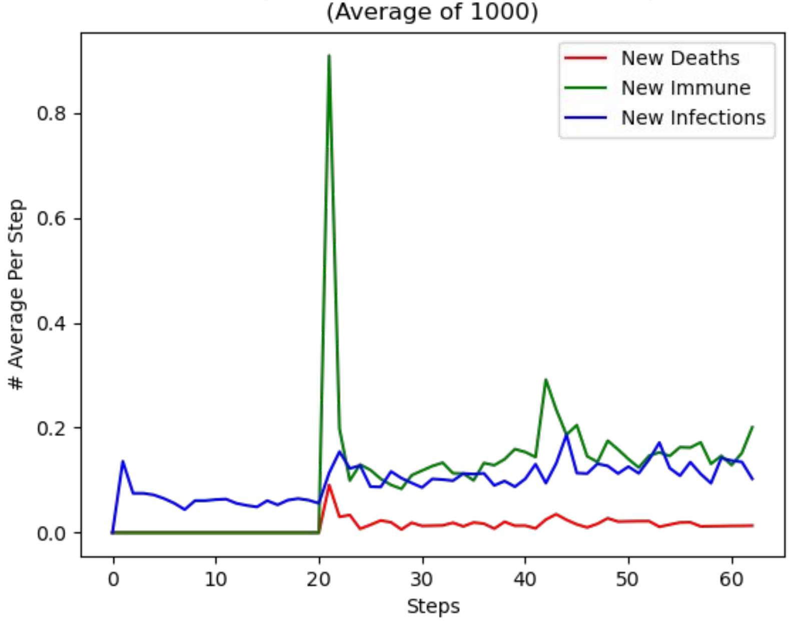
Simulation D

- Using parameters from Simulation B
- Resilience to virus added
- Social Distancing function added
- Wearing masks will increase resilience to virus.
- Some people will have higher resilience than others

Results of Running Sim D



Total Infected,Dead,Immune Per Step
(Average of 1000)



Newly Infected,Dead,Immune Per Step
(Average of 1000)

Adjustable Attributes

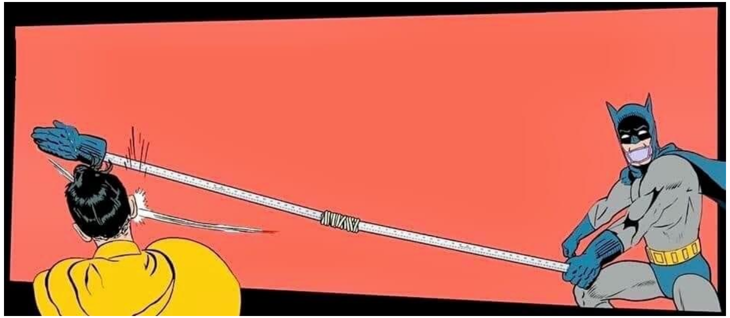
- Population density
- Social distancing/mobility measures
- How resilient one is to illness
- Preventative measures such as masks and hand washing

Us and Them

Differences in Simulations

- Handling of Social Distancing
- Sample Size
- Expertise
- White Box Model

What has this simulation taught us?



What we learned

1. Socially Distance
2. Wear a mask and wash your hands.
3. Disease spreads in direct proportion to population density
4. Disease can spread quickly if no preventative measures are in place

If you have any question about the code, feel free to ask it on stackoverflow

THANK YOU ;)