HW 2: Agent-Based Modeling of Pandemic Spread

Objective: Develop an agent-based model to simulate pandemic spread dynamics and explore the impact of inter-

vention strategies, such as social distancing, on infection rates and recovery.

A. Building the Base Model: Infection Dynamics in a Population

i. Define the Environment and Initial Conditions

• Create a 75×75 voxel grid representing a bounded area where individuals (agents) can move and

interact.

• Populate the grid with 100 agents, initialized randomly in the following states:

– 95 agents as susceptible (S) – individuals at risk of infection.

– 5 agents as infected (I) – individuals who can transmit the infection.

– 0 agents as recovered (R) – individuals who have recovered and are immune.

ii. Define Agent Behaviors

• Movement: Each agent moves to a neighboring cell each time step (up, down, left, right, or stays

in place). Movement can be random or follow simple rules, e.g., random walk (Brownian motion) or

Levy walk.

• Transmission: If a susceptible agent shares a cell with an infected agent, there is a probability pthat

the susceptible agent becomes infected.

• Recovery: Infected agents have a probability q of recovering at each time step, after which they

transition to the recovered state.

iii. Run the Simulation

• Simulate the model over 200 time steps, recording the population counts in each compartment (sus-

ceptible, infected, recovered) at each step.

• Plot the number of agents in each state over time to visualize infection spread, recovery, and eventual

immunity.

iv. Sensitivity Analysis

• Test different values of p (infection probability) and q (recovery probability). For instance, run

simulations with p= 0.05,0.1 and q= 0.02,0.05.

• Analyze how changes in pand qimpact infection peaks, time to infection peak, and overall population

recovery.

• Rerun the simulation with other random initial conditions. Compare the plots across different runs

and plot the average of each population subgroup over time. Do you see some average trend across

the random initial conditions?

B. Extending the Model: Social Distancing and Intervention Strategies

i. Introduce Social Distancing Measures: Modify agent movement behavior to simulate social distanc-

ing. For instance:

• When agents have a reduced probability of moving at each step, limiting their interactions. Which

parameter of the model should be changed to model this?

• When agents actively move away from cells with infected individuals when possible, simulating avoid-

ance behavior. Which parameter of the model should be changed to model this?

• Run the simulation for 200 steps with social distancing measures in place, recording and plotting the

average compartment counts across multiple runs (with random initial states) over time as before.

• Which parameter impacts the maximum number of infected subjects at each time window to prevent

the healthcare system overwhelm?

ii. Analyze and Compare Results: Compare the results of the simulations with and without social

distancing. Focus on metrics such as:

The peak number of infected individuals.

• The time it takes for the infection to peak and for the population to reach a stable recovered state.

• Overall infection spread and duration.

• Discuss how social distancing impacts infection rates and the timing of infection peaks, and consider

its implications for real-world public health interventions.

iii. Additional Sensitivity Analysis

• Test varying strengths of social distancing (e.g., different probabilities of movement) to explore how

stricter or more relaxed distancing affects infection dynamics.

• Record and discuss changes in infection patterns and recovery rates with each level of intervention.

I want to finish this task. can you guide me step by step?

I want to learn to code something myself, and I want you to guide me without giving full code. Act as a coding coach.

When I describe my goal or problem:

1. Help me break it into smaller sub-problems or steps.

2. Give hints, pseudocode, or questions that lead me to think about the solution.

3. Point me to relevant documentation, functions, or libraries I should explore.

4. Ask me to try implementing small pieces and explain my reasoning.

5. Only give code if I ask for an *example of a small part*, never the full solution.

Always focus on helping me **learn and understand**, rather than just giving a copy-paste answer.

how to set up the transmission function?

can you explain this code to me?

np.unique(positions, axis=0, return_inverse=True)

what does this do?

agent_indices = np.where(inverse_indices == cell_idx)[0]

is my code correct? and interpretation?

```
def transmission(positions, states, p):
    # find out who are in the same cell
    unique_cells, inverse_indices = np.unique(positions, axis=0, return_inverse=True)
    print(unique_cells[:10], inverse_indices[:10])
    # inverse_indices tells which index does this entry first appear
    for i in range(len(unique_cells)):
        shared_cell_indices = np.where(i == inverse_indices)[0]
        shared_cell_states = states[shared_cell_indices]
```

```
        if np.any(shared_cell_states == 1) and np.any(shared_cell_states == 0):
            s_cell_indices = shared_cell_indices[shared_cell_states == 0]
            s_or_not = np.random.rand(len(s_cell_indices)) < p # still S, otherwise I
            states[s_cell_indices[s_or_not]] = 1 # the True got infected

transmission(positions, states, p = 0.5)
```

I thought if the randomly generated p is less than p, then it remained suseptible, then it's True. they should be uninfected

is this correct?

```
def transmission(positions, states, p):
    res_states = states
    # find out who are in the same cell
    unique_cells, inverse_indices = np.unique(positions, axis=0, return_inverse=True)
    print(unique_cells[:10], inverse_indices[:10])
    # inverse_indices tells which index does this entry first appear
    for i in range(len(unique_cells)):
        shared_cell_indices = np.where(i == inverse_indices)[0]
        shared_cell_states = states[shared_cell_indices]
        if np.any(shared_cell_states == 1) and np.any(shared_cell_states == 0):
            s_cell_indices = shared_cell_indices[shared_cell_states == 0]
            i_or_not = np.random.rand(len(s_cell_indices)) < p # True = get infected
            res_states[s_cell_indices[i_or_not]] = 1
    return res_states
```

is this interpretation right?

inverse_indices tells which index does this entry first appear

is my code correct?

```
def recovery(positions, states, q):
    res_states = states.copy()
    infected_agents_indices = states[states == 1]
    r_or_not = np.random.rand(len(infected_agents_indices)) < q
    res_states = res_states[infected_agents_indices[r_or_not]]
    return res_states
```

is there a better way to use np arrays? so that I don't have to run a for loop?

```
states, positions = initialize()

def pop_count(states):
    s = len(np.where(states == 0)[0])
    i = len(np.where(states == 1)[0])
    r = len(np.where(states == 2)[0])
    res = [s, i, r]
    return res

pop_counts = []
for i in range(200):
```

```
    positions = movement(positions)
    states = transmission(positions, states, p=0.1)
    states = recovery(positions, states, q=0.05)

    pop_counts.append(pop_count(states))
```

I want to set seed, how to do that?

```
def initialize(seed = 1024):
    # initialize parameters
    grid_w, grid_h = 75, 75
    N = 100

    # initialize and set susceptible people
    states = np.zeros(N, dtype = int)
    # set infected people
    infected_indices = np.random.choice(N, size=5, replace=False)
    states[infected_indices] = 1

    # initialize their position on the grid.
    positions = np.random.randint(0, grid_w, size = (N, 2))

    return states, positions
```

```
states, positions = initialize()

print('S', np.sum(states == 0))

print('I', np.sum(states == 1))

print(positions[:5])
```

is this correct coding?

```
def reduced_movement(positions, move_prob = 0.3, grid_w = 75, grid_h = 75):
    directions = np.array([[0,-1],[0,1],[-1,0],[1,0], [0,0]]) # left, right, up, down, stay
    movement_indices = np.random.randint(0, 5, size = len(positions))
    moves = directions[movement_indices]

    mask = np.random.rand(len(moves)) < move_prob
    positions[mask] = positions[mask] + moves[mask]

    # keep agents within grid bounds (clip to [0, 74])
    positions[:, 0] = np.clip(positions[:, 0], 0, grid_w - 1)
    positions[:, 1] = np.clip(positions[:, 1], 0, grid_h - 1)

    return positions

print('before', positions[:5])

print('after', movement(positions)[:5])
```

could you help me with the second avoid infection coding?

def avoid_infected(positions, states):

neighbors = np.array([[0,-1],[0,1],[-1,0],[1,0], [0,0]]) # left, right, up, down, stay

neighbors_states = positions + heighbors

infected_states = np.where(neighbors_states == 1)

does this work?

def avoid_infected(positions, states):

infected_states = np.where(positions == 1)

for i in range(len(positions)):

if states[i] == 0:

diffs = positions[i] - positions[infected_states]

I don't get your if statement, why would length of infected positions == 0?

it's not that I only move away from when the distance is less or equal to 1. it that I'd prefer moving away at every step. trying to increase the nearest distance right?

the reduced movement and avoid movement are all modification towards movements. should there be a way to unite them?

when I run this code grid_h, grid_w = 20, 20 p = 0.9 q = 0.02 states, positions = initialize(seed = 1024, grid_w = grid_w, N = 100, size = 20) pop_counts_avoid2 = simulate_avoid(states, positions, p = p, q = q, grid_h = grid_h, grid_w = grid_w, move_prob = 0.5) plot(pop_counts_avoid2, p = p, q = q) pop_counts_avoid3 = simulate_avoid(states, positions, p = p, q = q, grid_h = grid_h, grid_w = grid_w, move_prob = 0.8) plot(pop_counts_avoid3, p = p, q = q) The second graph, have a starting number of infected agents near 40. adn the fisr graph has a starting number of infected agents near 25. they are not the 20 I set. Why

here’s what I wrote, without changing what I wrote.

Summarize parameter sensitivity analysis in comparing p and q

In Fig 1, p=0.5 and q=0.02. The peak is high, reaching its infection peak at approximately 36 infected agents in 100 time steps. With low recovery probability q, individuals stay sick for a

long time. The recovered cover grows slower than the infected curve till 100 time steps, and the susceptible population drops significnatly.

In Fig 2, p=0.1 and q=0.02. The infection peak is very low, peaking aroung 7 agents in 15 time steps. Because of the low recovery probability, the infected population didn't drop to zero until 180 time steps, and the recovered population only increased by a small amount. Most people stay susceptible throughout the whole experiment.

In Fig 3, p=0.5 and q=0.05. The infection peak is moderate, peaking aroung 17 agents in 27 time steps. With high infection rate and high recovery rate, infected individuals recover much faster, leading to a much smaller and earlier infection peak compared to Fig 1. The slope for recovered curve is much steeper and leaves a higher number of susceptible population.

In Fig 4, p=0.1 and q=0.05. The infection peak is also very low, peaking aroung 10 agents in 15 time steps. Similar to Fig 2, the infection rate is too low to spread the disease. With a high recovery rate, the infection drops to zero quickly, and most people remain susceptible.

Summarize parameter sensitivity analysis in comparing different initialization parameters

- To compare different grid size, which is a proxy for population density:

Fig 5 (Small Grid Size = 25): In the simulation with a small grid size (grid_w = 25), the high population density facilitates a major outbreak. The infection curve shows a significant peak, rising to approximately 20 agents around time step 75. This strong spread leads to a substantial decline in the susceptible population, which drops from ~95 to below 40 agents by the end of the simulation.

Fig 6 (Large Grid Size = 175): In the simulation with a large grid size (grid_w = 175), the low population density stifles the pandemic. The infection peak is very low, reaching only ~6 agents around time step 25 before quickly declining. The low probability of agent interaction prevents sustained transmission. Consequently, the infection dies out, and the vast majority of the population remains susceptible.

- To compare different population size:

Fig 7 (Small Population = 10): In the simulation with a small total population (N = 10, starting with 5 infected), the dynamics are rapid. The infection peak is its starting value of 5 agents, which then rapidly drops toward zero. With such a small pool of agents, the susceptible population remains almost unchanged, while the recovered curve rises quickly to 5, reflecting the recovery of the initial infected group.

Fig 8 (Large Population = 200): In the simulation with a large total population (N = 200, starting with ~5 infected), the infection fails to gain traction. Despite a large susceptible pool (~195 agents), the proportion of initial infectors is too small to initiate a widespread outbreak. The infection peak is extremely low, barely rising from 5 to 9 agents, and the susceptible population remains almost entirely untouched.

- To compare different initial infection population size:

Fig 9 (Small Initial Infection = 2): In the simulation with a small initial infection size (2 agents), the infection fails to launch. The infected peak is negligible, hovering near the starting value of 2 agents before declining. With such a small initial seed, the virus likely dies out before significant transmission can occur. As a result, the recovered population remains near zero, and the susceptible population is almost entirely unaffected.

Fig 10 (Large Initial Infection = 50): In the simulation with a large initial infection size (50 agents), the pandemic begins at its peak. The infected curve starts at 50 and immediately declines as agents recover, showing no initial growth phase. The recovered curve rises steeply, surpassing the infected curve around time step 40. This large initial outbreak burns through a portion of the population, causing the susceptible curve to drop from 50 to ~42 agents.

Compare the results of the simulations with and without social distancing.

Fig 11 (Original Dynamics - No Intervention): In the baseline simulation with no interventions, the infection spreads rapidly and widely. The infected curve shows a significant outbreak, rising from 5 agents to a high peak of approximately 25 infected agents

around time step 80. This uncontrolled spread leads to a steep decline in the susceptible population, which falls from ~95 to ~18.

Fig 12 (Reduced Movement - Social Distancing): In the simulation with reduced movement, the curves are flatten. The infection peak is significantly lower, reaching only ~16 agents. The rate of infection is slower, leading to a broader, less severe peak. This intervention successfully slows the pandemic's spread, leaving a much larger susceptible population (~44) at the end of the simulation compared to the baseline.

Fig 13 (Avoid Movement - Social Avoidance): In the simulation with "avoid movement," the intervention is effectively suppresses the outbreak. The infection fails to spread in the population, showing a small peak of ~6 agents before quickly declining to zero by time step 100. The virus is unable to spread. Thus, the susceptible population remains almost entirely unaffected, staying at ~94 agents.

These graphs demonstrate the direct trade-offs of public health interventions. The "Original" scenario (Fig 11) represents an uncontrolled outbreak that would likely overwhelm public health systems. The "Reduced Movement" (Fig 12) shows how social distancing lowers the peak, buying time for healthcare response and reducing the maximum simultaneous burden, even if many people still get sick over a longer period. "Social Avoidance" (Fig 13) illustrates how strict lockdowns can suppress the virus, but this strategy leaves a large, non-immune (susceptible) population vulnerable to a resurgence of the virus if the interventions are lifted.

I want to display bosth text and images in the README file. here are the file names:

ls

ave_grid175.png ave_inf2.png ave_popu10.png avoid0.2.png
reduce0.2.png simu1.png simu3.png social_avoidance.png
social_original.png

ave_grid25.png ave_inf50.png ave_popu200.png avoid0.8.png
reduce0.8.png simu2.png simu4.png social_distance.png

identify which are fig 1 through 17. based on naming. here are some help:

```
 ![Fig 1](Figs/simu1.png)
```

can you display the fig name in the readme? I only want Fig # not the description

what is model based machine learning