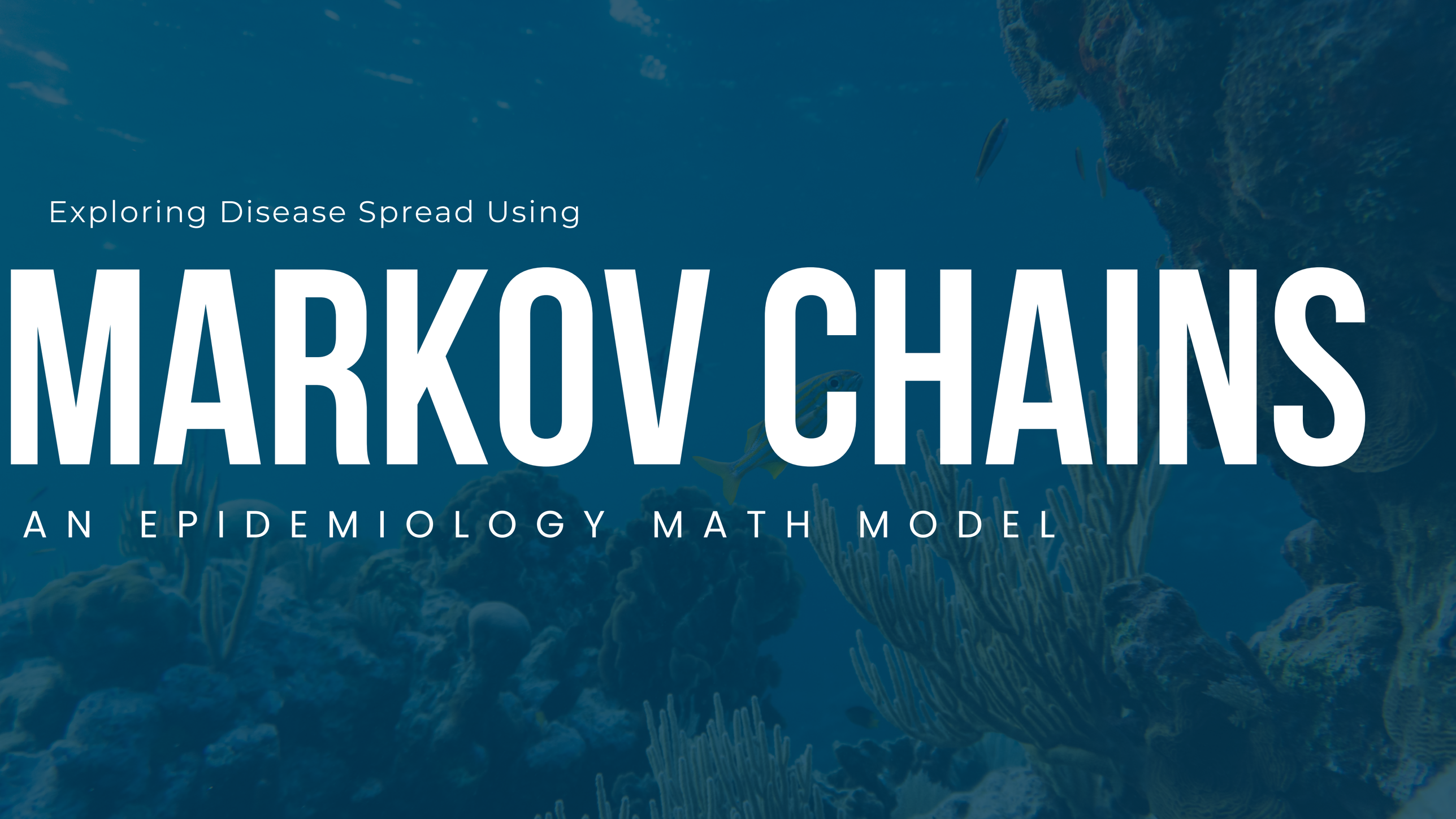
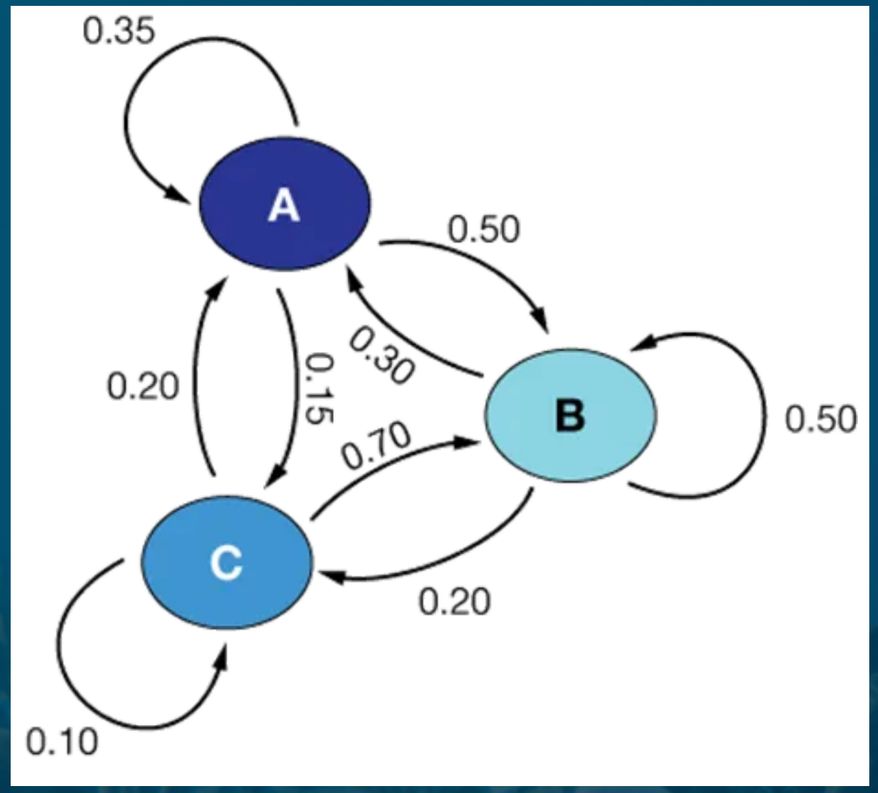


Exploring Disease Spread Using

MARKOV CHAINS

AN EPIDEMIOLOGY MATH MODEL



INTRODUCTION

CREATED BY THE RUSSIAN MATHEMATICIAN ANDREY MARKOV

A Markov Chain is a mathematical model that describes a sequence of events where the next state depends only on the current state, not on the past history.

APPLICATION

HOW I WILL IMPLEMENT

I simulate how a disease spreads through a population using a Markov Chain model, where the states I will be using will include Susceptible (S), Infected (I), and Recovered (R). Transitions between states will occur based on defined probabilities.

APPLICATION

HOW I WILL IMPLEMENT

I simulate how a disease spreads through a population using a Markov Chain model, where the states I will be using will include-> Susceptible (S), Infected (I), and Recovered (R). Transitions between states will occur based on defined probabilities.

SUSCEPTIBLE (S) FIRST VARIABLE

SECOND VARIABLE **INFECTED (I)**

RECOVERED (R) THIRD VARIABLER

(Once recovered, individuals don't return to previous states.)

$$\begin{bmatrix} 1 - p_{infection} & p_{infection} & 0 \\ 0 & 1 - p_{recovery} & p_{recovery} \\ p_{loss} & 0 & 1 - p_{loss} \end{bmatrix}$$

MATRIX DESIGN

- A Markov Chain uses a transition matrix of probability to move between states.
- Key property: Memoryless — next state depends only on the current state.

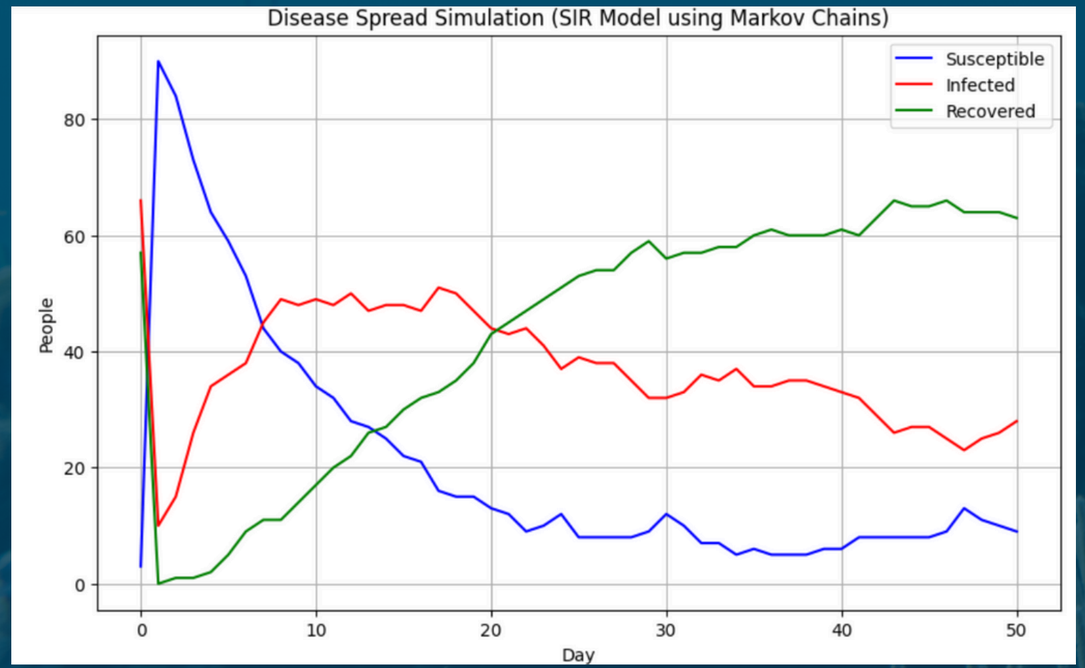
From \ To	S (Susceptible)	I (Infected)	R (Recovered)
S	0.90	0.10	0.00
I	0.00	0.85	0.15
R	0.00	0.00	1.00

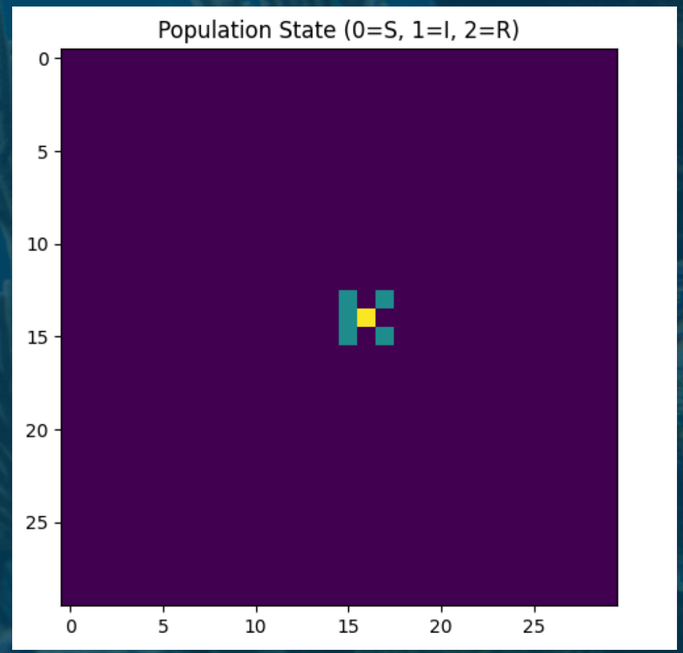
MATRIX EXAMPLE

- A Markov Chain uses a transition matrix of probability to move between states.
- Key property: Memoryless — next state depends only on the current state.

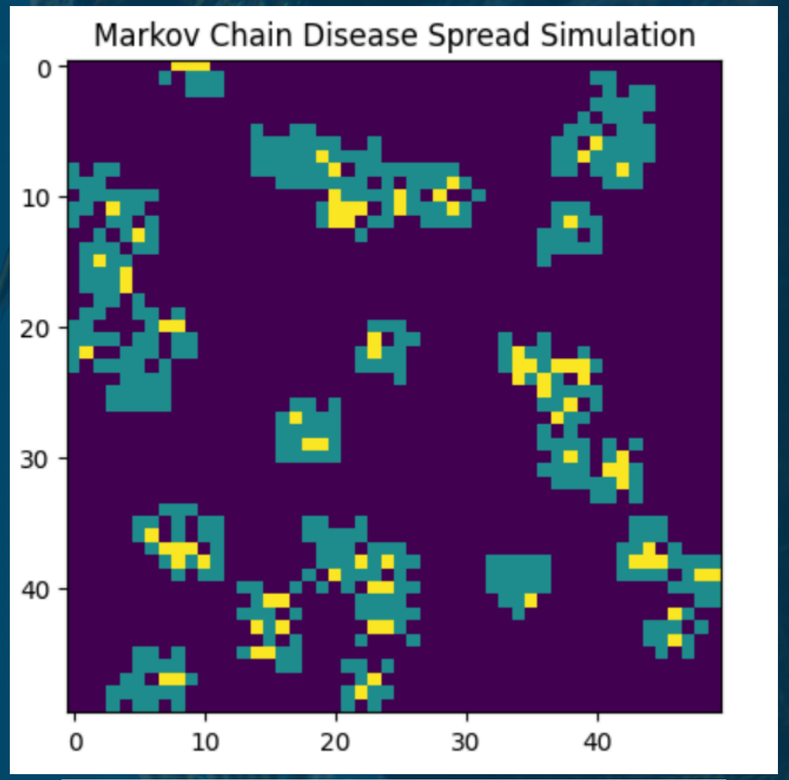
```python
for day in range(days):
    new_states = states.copy()

    for i in range(population_size):
        if states[i] == 0:  # Susceptible
            # Check if there's any infected person around (simplified)
            if np.any(states == 1) and np.random.rand() < p_infection:
                new_states[i] = 1
        elif states[i] == 1:  # Infected
            if np.random.rand() < p_recovery:
                new_states[i] = 2
        elif states[i] == 2:  # Recovered
            if np.random.rand() < p_loss_immunity:
                new_states[i] = 0
```

```python
    states = new_states

    # Record for graphing
    S = np.sum(states == 0)
    I = np.sum(states == 1)
    R = np.sum(states == 2)

    history_S.append(S)
    history_I.append(I)
    history_R.append(R)

# Plot results
plt.figure(figsize=(10, 6))
plt.plot(history_S, label='Susceptible', color='blue')
plt.plot(history_I, label='Infected', color='red')
plt.plot(history_R, label='Recovered', color='green')
plt.xlabel('Day')
plt.ylabel('People')
plt.title('Disease Spread Simulation (SIR Model using Markov Chains)')
plt.legend()
plt.grid(True)
plt.show()
```

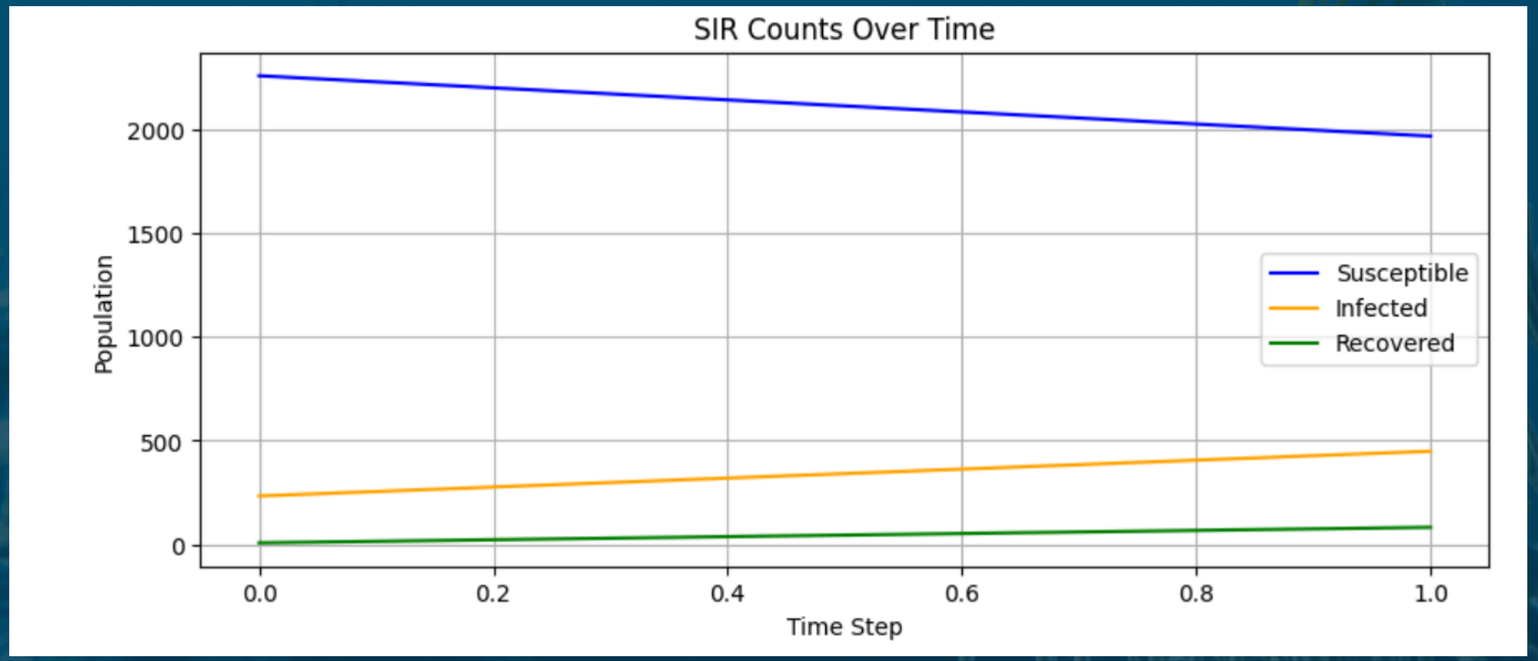


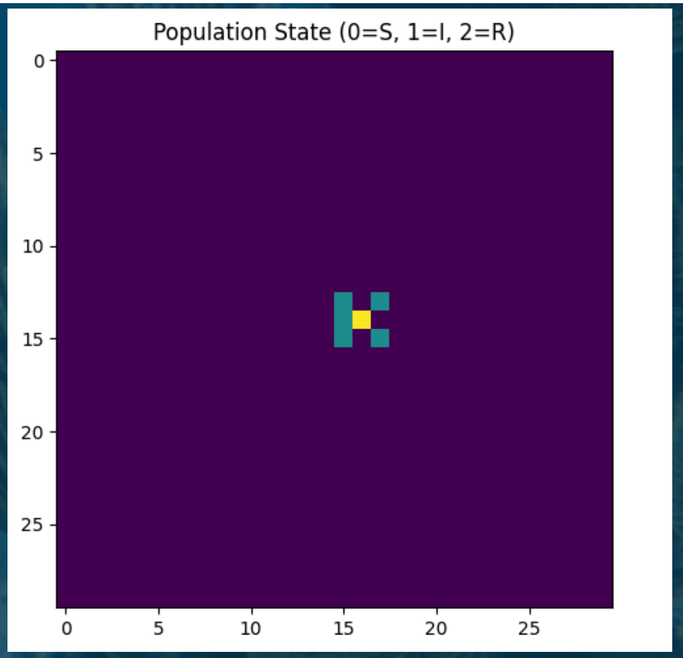


```
plt.figure(figsize=(10, 4))
plt.plot(S_history, label='Susceptible', color='blue')
plt.plot(I_history, label='Infected', color='orange')
plt.plot(R_history, label='Recovered', color='green')
plt.xlabel('Time Step')
plt.ylabel('Population')
plt.title('SIR Counts Over Time')
plt.legend()
plt.grid()
plt.show()
```



from nmatplotlib.animation we imported FuncAnimation





CONCLUSION:

- There is scope to include more elements to this project
- There is possibility for making this project more realistic(immunity)
- Can be made more specified to the way different outbreaks happen
- Can incorporate more realistic maps.


THANK YOU