Echoes of the Spanish Lady: Navigating the Maze of Epidemic Models

Andrew Becker

Middle Tennessee State University, Murfreesboro, Tennessee, USA

***Abstract*— Cellular Automata is a mathematical system with the capability to generate complex behaviors from a simple set of rules operating on a grid of locally finite automata. Each automaton produces an output with the next state depending on its current state and the states of its neighboring cells. This process is used to analyze the SIR model of infectious diseases.**

Strategies to mitigate the impact of disease spread through mathematical modeling are critical to public health authorities, focusing on the socio-economic benefit of returning to normalcy. We develop and analyze a model for the Spanish Flu with epidemiological parameters and efficacy rate learned through a feed-forward neural network.

A hybrid approach combining a residual neural network with variants of recurrent neural networks is implemented and scrutinized for reliable and precise prediction of daily cases. Through error metrics and k-fold cross validation with random splitting, it is found that a residual neural network with a gated recurrent unit is the optimal architecture. Data-driven simulations confirm that higher efficacy in the mathematical modeling lowers the infectiousness and basic reproduction number. For this study, data for the state of Tennessee in the USA during the Spanish Flu is used.

Keywords*— *Epidemic Chain Models, Markov Chain (as alternative to), Message Passing Interface (MPI), OpenMP, Parallel Processing, Plague Inc., Pthreds, Kermack-McKendrick Model, SIR Model (a) Polyhedra, diameter, Hirsch conjecture, simplex method, simplicial complex (T) deep learning; data-driven; COVID-19; ResNet; RNN; vaccination strategy; k-fold cross validation; Infectious disease epidemiology, influenza, mathematical modelling, Susceptible-Infected-Removed (SIR) model

I. Introduction

THE spread of the H1N1 virus, known as the Spanish Flu, began in 1918. The virus spread across the globe quickly, affecting countless lives and causing an immense number of deaths, prompting a global health crisis. Studies on mitigation measures such as social distancing, quarantining, and government shutdowns for controlling the disease highlight the need for an effective intervention strategy. The rate at which protective measures are enacted and their efficacy are two main factors in an effective intervention program. Coordinated efforts by all stakeholders such as policymakers, pharmaceutical companies, researchers, and health professionals are required to effectively manage the spread of the disease. Recent studies on intervention efficacy and public confidence demonstrate the importance of an effective strategy. Mathematical models have been used to study the dynamics of an epidemic using systems of differential equations. The model used in this work is the SIR model: it divides the population into three groups, namely Susceptible (S), Infected (I), and Recovered (R). The extensions of this basic model have been applied to the Spanish Flu in the past. During a pandemic period, the main challenge in this type of model is ascertaining the model's parameters. The conventional least square method can be used to estimate reasonable constant and time-dependent parameters. Data-driven models employing deep learning architectures have been applied to epidemiological models. Raissi et al. used physics-informed neural network (PINN) to learn constant parameters of the Influenza model. Kharazmi et al. applied PINN to learn time-dependent parameters of both integer and fractional order epidemiological models. Long et al. demonstrated how PINN is used to learn the time-dependent transmission rate for the SIR and SIRD models, while Grimm et al. demonstrated the effectiveness of using PINN to learn the time-dependent transmission rate of SIR and SEIR models. Conventional forecasting techniques such as Auto-Regressive Integrated Moving Average (ARIMA) have been used to make short-term forecasts for confirmed cases of the Spanish Flu in different countries. Other studies utilized machine learning algorithms, such as Support Vector Machine (SVM), Linear regression, and Exponential Smoothing (ES), for future predictions of the Spanish Flu. Recent applications of recurrent neural networks (RNN) with its variant Long Short Term Memory (LSTM) for forecasting the Spanish Flu in Canada, Italy, and the USA showed reasonable performance in prediction. The authors in [4] applied LSTM, Bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU) for data-driven simulations of the Spanish Flu data for different countries. Uncertainties arise from the model, the parameters, and other external factors that impact the accuracy in prediction. There is a need to quantify the uncertainty associated with models for forecasting the trend analysis of the Spanish Flu. The Residual Neural Network (ResNet) is considered, which is capable of learning dynamical systems. The residual connections help to improve accuracy in training and prediction and are considered as time-stepping techniques modeled after adaptive numerical methods using the concept of integration. Chen et al., in their recent work, showed that the generalized ResNet has the capability of learning intricate unknown dynamic structures of chaotic systems and predicting results with higher accuracy than the standard ResNet structure.

II. Background

In this paper, a hybrid approach based on residual neural network combined with variants of recurrent neural network for reliable and accurate short-term predictions is considered. The hybrid approach consists of ResNet-LSTM, ResNet-BiLSTM, and Res

Net-GRU. The impact of the intervention model with efficacy on each group of the population is analyzed. The feed-forward neural network solves the inverse problem which incorporates the epidemiological parameters in the loss function. Error metrics for data-driven simulations are discussed to support the claim of effectiveness for the hybrid approach. Furthermore, error metrics and a k-fold cross validation with random splitting are discussed. The paper is organized as follows. Section 2 gives an overview of materials and methods in terms of the mathematical model including the Susceptible, Infected, Recovered Vaccine model, non-negativity and boundedness of the model, data preprocessing, error metrics and k-fold cross validation. Section 3 presents the results and discussion of data-driven simulations along with error metrics. Section 4 summarizes the work and presents conclusions. (AI-assisted conversion)

Project Focus

Leveraging cellular automata for simulating epidemic models, specifically the SIR (Susceptible, Infected, Recovered) and SEIR (Susceptible, Exposed, Infected, Recovered) models. The goal is to compare the efficiency and accuracy of these models with traditional techniques like Markov Chains, to develop more efficient and practical tools for simulating disease spread. Additionally, the project aims to explore the use of machine learning for model refinement and seeks potential applications of these models beyond public health, such as in game development.

Project Description & Scientific Problem

The scientific problem being addressed in this project is the efficient and accurate simulation of epidemic models, specifically the SIR and SEIR models, using parallel processing techniques and cellular automata. Traditional techniques like Markov Chains can be computationally intensive and might not scale efficiently. The project aims to overcome the challenge of refining these models to better fit real-world observations without overfitting. Ultimately, the project seeks to develop practical tools for simulating disease spread that can also be applied in diverse fields such as game development.

Scientific Conclusions

Through this project, we hope to achieve several scientific conclusions: 1. Demonstrate the effectiveness of parallel processing and cellular automata in simulating the SIR and SEIR epidemic models, potentially showcasing an alternative to traditional methods like Markov Chains that can be more efficient and scalable. 2. Determine the accuracy of these simulations compared to real-world data and understand the balance between model refinement and overfitting. 3. Explore the practicality and applicability of these simulation tools beyond academic research, such as their use in public health planning or game development. 4. Provide insights into the potential integration of machine learning techniques for automatic model refinement and adaptive simulation techniques. 5. Establish a groundwork for future research, such as the inclusion of more complex factors in epidemic models (like carrier states or vaccination rates) and the potential for a reverse-engineering approach to model selection.

Simulation & Data Analysis

The simulation in this project will involve the use of cellular automata to model the spread of an infectious disease based on SIR and SEIR models. We'll represent the population as a two-dimensional grid, where each cell corresponds to an individual and their state: Susceptible, Exposed, Infected, or Recovered. The initial state of each cell will be randomly assigned to reflect a realistic distribution of population states at the start of a disease spread. This simulation will run on a parallel computing environment, leveraging technologies like MPI and OpenMP. By assigning different portions of the grid to multiple processes, we aim to enhance computational efficiency. Once the simulation is running, it will apply the rules defined by the SIR and SEIR models for each time step. Each individual's state will evolve based on these rules, and the aggregate state of the population will be captured and analyzed. In terms of data analysis, we will introduce the concept of an 'error function' that assesses the difference between the simulated data and real-world observations. This approach will allow us to refine the parameters of our model iteratively to improve its predictive accuracy while preventing overfitting. Ultimately, we aim to generate a rich dataset that provides insight into the temporal dynamics of disease spread and the comparative efficiency and accuracy of SIR and SEIR models when implemented in parallel computing environments.

Describe the simulation or data analysis you will be doing in this project.

Simulation models are of increasing importance within the field of applied epidemiology. However, very little can be done to validate such models or to tailor their use to incorporate important human behaviours. [12]

What algorithms and numerical methods to you anticipate applying in this work?

(GPT) In a recent incident in the virtual world of online gaming, the accidental inclusion of a disease-like phenomenon provided an excellent example of the potential of such systems to alleviate these modelling constraints. We discuss this incident and how appropriate exploitation of these gaming systems could greatly advance the capabilities of applied simulation modelling in infectious disease research. [12]

Algorithms & Numerical Methods

(GPT) Several algorithms and numerical methods are anticipated to be applied: 1. Cellular Automata: This will be the preliminary method for modeling the disease spread. Each cell in a grid represents an individual, and their state (Susceptible, Exposed, Infected, Recovered) evolves according to the SIR/SEIR model rules. 2. Numerical Integration: As the SIR and SEIR models involve ordinary differential equations, numerical integration methods might be used for their solution, especially if we wish to simulate continuous time models. 3. Random Number Generation: Some initial states of individuals (cells) in the grid will be assigned randomly, which will require a reliable method for generating random numbers. 4. Error Function and Optimization Algorithms: To refine the model parameters, we will define an error function that quantifies the difference between simulated data and real-world observations. Then, we can use optimization algorithms (e.g., gradient descent, stochastic gradient descent, etc.) to adjust parameters and minimize this error. 5. Machine Learning Algorithms: Depending on the direction of the research, we might employ machine learning techniques to automate model refinement, or even reverse-engineer the best-fitting model based on observed data.

Programming & Development

The project will be developed in Python and the programming work will likely involve the following: 1. Setting Up the Simulation Environment: This would involve setting up the cellular automata grid, initializing individual states randomly according to the SIR/SEIR model rules, and defining the transition functions for changing states. Libraries such as NumPy can be used to efficiently handle arrays representing the population grid. 2. Implementing SIR/SEIR Models: We will be writing functions or classes to represent the SIR and SEIR models, which will include the associated differential equations. These will be used to determine state transitions in the simulation. SciPy's integrate module can be utilized for numerically solving these equations. 3. Running the Simulation: This would involve iterating over the defined time steps, updating the state of each cell according to the SIR/SEIR rules at each step, and storing or outputting the state of the entire system at each step for subsequent analysis. 4. Data Analysis and Model Refinement: Here, we write code to analyze the results of the simulation, such as computing the error between the simulated data and real-world data. We also use this analysis to iteratively refine model parameters. Libraries like Pandas for data handling and Matplotlib for visualization can aid in this process. 5. Machine Learning Integration: Depending on the project's scope, we could also implement machine learning models using libraries such as Scikit-Learn or TensorFlow. These models could help automate the process of model refinement or even potentially reverse engineer the best fitting model based on real-world data. Python's extensive standard library and the availability of scientific computing packages like NumPy, SciPy, Pandas, and Scikit-learn make it a great choice for such a project.

Relevant Papers

Deep-Data-Driven Neural Networks for COVID-19 Vaccine Efficacy by Thomas K. Torku, Abdul Q.M. Khaliq & Khaled M. Furati

This paper develops a vaccination model with an efficacy rate, using a hybrid neural network approach for reliable daily case predictions, demonstrating through data-driven simulations that higher vaccination rates with more effective vaccines decrease the infectiousness and basic reproduction number, using data from Tennessee as a case study.

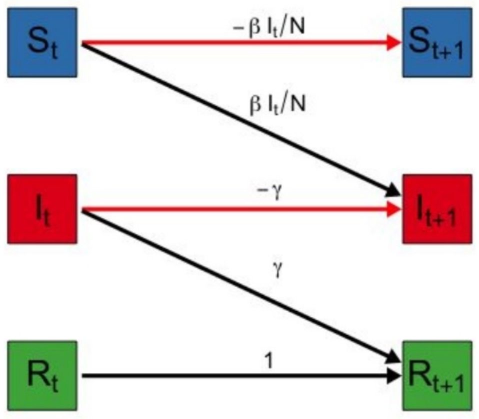


Figure 1. A block diagram of the baseline system

III. Literature Review

The Extrapolation of First Order Methods for Parabolic Partial Differential Equations J.D. Lawson and J. Ll. Morris December 1978

SPLITTING methods for parabolic partial differential equations based on (1,1) Padé approximations (Crank-Nicolson replacements) are well known to produce poor numerical results when a time discretization is imposed with time steps which are " too large" relative to the spatial discretization. In particular such numerical solutions exhibit an oscillatory behavior which increases in amplitude with a reduction in the spatial discretization, keeping the time step constant. In contrast, (1,0) Padé approximations (backwards difference, of fully implicit replacements) do not suffer from these drawbacks but are of lower order accuracy. In the present paper, a combination of fully implicit methods is used to attain second order accuracy and to retain the favorable property of the fully implicit scheme. The method is tested on a heat equation in two space dimensions which possesses a discontinuity between the initial and boundary conditions. (The Extrapolation of First Order Methods for Parabolic Partial Differential Equations J.D. Lawson and J. Ll. Morris December 1978)

Recent progress on the combinatorial diameter of polytopes and simplicial complexes Francisco Santos July 24, 2013

The Hirsch conjecture, posed in 1957, stated that the graph of d-dimensional polytope with n facets cannot have diameter greater than $n-d$. The conjecture itself has been disproved, but what we know about the underlying question is quite scarce. Most notably, no polynomial upper bound is known for the diameters that were conjectured to be linear. In contract, no polyhedron violating the conjecture by more than 25% is known.
This paper reviews several recent attempts and progress on the question. Some work in the world of polyhedral or (more often) bounded polytopes, but some try to shed light on the question by generalizing it to simplicial complexes. In particular, we include here our recent and unpublished proof that the maximum diameter of arbitrary simplicial complexes is in $n^{\Theta}(d)$ and we summarize the main ideas in the polymath 3 project, a web-based collective effort trying to prove an upper bound of type *nd* for the diameters of polyhedral and of more general objects (including, e.g., simplicial manifolds). Keywords Polyhedra, diameter, Hirsch conjecture, simplex method, simplicial method. (Recent progress on the combinatorial diameter of polytopes and simplicial complexes Francisco Santos July 24, 2013) [11]

Corrupted Blood Virus

The Corrupted Blood incident (also known as the World of Warcraft pandemic[1][2]) took place between September 13 and October 8, 2005, in World of Warcraft, a massively multiplayer online role-playing game (MMORPG) developed by Blizzard Entertainment. When participating in a boss battle at the end of a raid, player characters would become infected with a debuff that was transmitted between characters in close proximity. While developers intended to keep the effects of the debuff in the boss's game region, a programming oversight soon led to an in-game pandemic throughout the fictional world of Azeroth.

World of Warcraft introduced the game region of Zul'Gurub on September 13. The boss of the region, Hakkar the Soulflayer, cast Corrupted Blood on raid participants; the debuff's effects expired when players defeated Hakkar. Corrupted Blood soon spread beyond Zul'Gurub as players reacted to the infection with panic, either fast traveling to heavily-populated game regions or deactivating their animal companions. When those companions were reactivated, they still carried the debuff, becoming disease vectors, while non-player characters became asymptomatic carriers. Player reactions to the Corrupted Blood pandemic varied: some provided aid by healing players or warning them of outbreak zones, while griefers intentionally contracted the debuff to spread it across World of Warcraft. After several failed hotfixes, Blizzard ended the pandemic by performing a hard reset, and a later patch prevented companions from contracting Corrupted Blood entirely.

Although it was the result of a software bug, the Corrupted Blood incident gained attention from World of Warcraft players and disease researchers. Blizzard developed intentional in-game pandemics in two expansion sets: Wrath of the Lich King in 2008 and Shadowlands in 2020. Epidemiologists, meanwhile, took interest in how MMORPGs, unlike mathematical models, could capture individual human responses to disease outbreaks rather than generating assumptions about behavior.

Corrupted Blook incident - Wikipedia

IV. Methods • Games and Graphs

MATHEMATICAL modeling an active process, rather than a static object of study. We practice modeling as a systematic approach use techniques and structures of mathematics to make the pursuit of abstract mathematical knowledge. For example, John Conway's game of life is an early example of an emergent system of computers, where a process, once activated, can become a special phenomeonon.

Draw Conclusions based on Probabilistic Models

Modeling offers the power of mathematics to the needs of a society, and so many other areas of human flourishing. When we play games that are designed with emergent systems, we consider parallels to our own world. While the map can never be the territory, the game isn't reality, the metaphor of models can be explored in games like SimCity, Civilization or Plague Inc., even if we are only passively aware of this.

Did the Spanish Flu originate in Kansas?

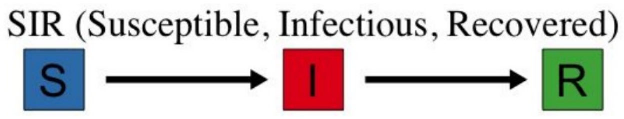
After the Spanish Flu, mathematical models have been used to place the beginning of the disease in rural Kansas. However the pandemic in 2020 has shown us that models can be cumbersome and difficult to implement, and also become more sophisticated over time. Can we use modern games to produce models that return the same conclusion? One game in particular which has become popular, Plague Inc., where a player is tasked with destroying every last human with an epidemic disease, in a noir-style race against the humans and their cure. Indeed it is a race against competing model curves.

Epidemic Models for Public Health

The model we will be using includes, as a particular case, the epidemic chain model corresponding to the stochastic Kermack-McKendrick model and, as a limiting case, the Reed-Frost chain binomial mode. The more general model are illustrates with an application to household data for the common cold. Finally, it is shown how the coefficient of variation of the duration of the infectious period may be estimated without any direct observations on this duration [1].

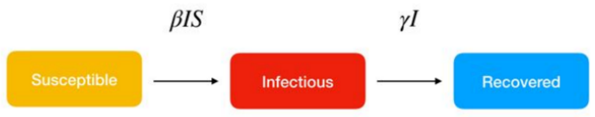
Abbreviations and Acronyms

The Model will be discussing has some common terminology, which we introduce here, as well as some alternative letters typically used alongside these letters. As pictured below, we see the chain in the simple SIR model. We might also consider alternate states, such as deaths, those who aren't born yet (in the case of a longer-term model, or people vaccinated [4]. When we add up all of the people in the SIR model at every given moment, everyone in the model will be represented ($S+I+R=100\%$). Thus we are considering the movement from S to I and I to R, which we represent with differential equations and whose sum must equal zero (since everyone is in the model when they aren't transitioning.)



Overview of Relevant Literature

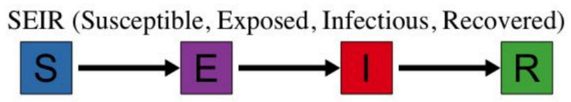
An epidemic chain model is developed by assuming a beta distribution for the probability of being infected by contact with a given infection from the same household. We will be using a probabilistic generator as a simpler approach that simplifies from techniques like the Markov Chain.



Epidemic Chain Models

Model fitting is a procedure that takes three steps: First you need a function that takes in a set of parameters and returns a predicted data set. Second you need an 'error function' that provides a number representing the difference between your data and the prediction for any given set of parameters.

SIR Implies More Complicated Models



SIR lends itself toward Complicated Models. There are many other compartmental models that extend SIR. The closest is the SEIR model that splits out the infected population into two sub-groups, those who are infected but not yet contagious and those who are infectious.

The SEIR model is very similar to the SIR model, but Susceptible people who become infected move first move into the Exposed group. There is one additional parameter that controls how long a person stays in the Exposed group before they move into the Infectious state.

A model cannot perfectly map what it is trying to represent, and in the case of games or simulations, we are looking to fit models and data as close as we can without sacrificing either over-fitting or (with games) exploration.

V. Results • Project Overview

We are going to be using some simple values to practice using parallel processing techniques inside the SIR model. The picture below is a rough approximation of what we expect to see over time, with different inputs producing slightly different curves.

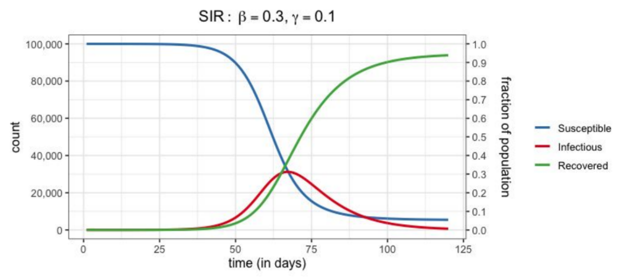
These curves come from the (continuous time) SIR model which specifies a set of three ordinary differential equations:

Datasets

The Kermack-McKendrick epidemic model of 1927 is an age of infection model. That is, a model in which the infectivity of an individual depends on the time since the individual became infected. A special case, which is formulated as a two-dimensional system of ordinary differential equations, has often been called the Kermack-McKendrick model.

One of the products of the SARS epidemic of 2002-2003 was a variety of epidemic models including general contact rates, quarantine, and isolation. These models can be viewed as age of infection epidemic models and analyzed using the approach of the full Kermack-McKendrick Model.

All of these models share the basic properties that there is a threshold between disappearance of the disease and an epidemic outbreak, and that an epidemic will die out without infecting the entire population. [2]



VI. Result and Discussions

Population is set up as a grid (2D array); each cell in the array represents one person (current population limit is 10,000)

The values of the array are randomized – each cell is assigned a state in the model: S, I, or R. The totals of these values will be used for i0, the first day of the observation period.

Using MPI, the master process creates a certain number of worker processes (inputted at compilation) and assigns a portion of the array to each process.

The worker processes calculate the totals of each stage of the model in their portion of the array and add to the running totals of the stages. These are used for the initial values at the start of the observation period.

For the purposes of this parallelization demonstration, we will stick to a simplified SIR model.

• S - the fraction of the population that is able to be infected

• I - the fraction of the population currently infected

• R - the fraction of the population that no longer has the infection

Other factors for the moment are not considered by the model. For this part we are simply practicing parallelizing a sample 2D model like the Kermack-McKendrick Model.

One purpose for completing this exercise is to explore alternative methods to Markov Chains that might produce faster results that are closely accurate with more intense traditional data methods. Games don't have the same burdens of exactitude as public health models, and can operate in the exploratory realm.

S-I-R Model Simulation

People that are recovered, people that are vaccinated, and people with natural immunity.

Our model sets the start of the observation period at a point in which the distribution of the population across the model reflects one closer to the middle of the simulated infectious disease spread.

Assumptions

Assumptions used for the simulation are adapted from Sullivan, Brian; SIR Model for Disease Spread 1: Introduction

Population Size Fixed

In the model we assume no births, nobody new enters the model, and nobody leaves the model for any reason.

Recovery Gives Full Immunity

Once a person is recovered, they cannot become infected or susceptible again and remain in the recovered column through the conclusion of the simulation.

Fixed Infection Rate

Measured in people infected per day.

```
import math   # used for mathematical functions
import random # to generate random numbers
import time   # to seed the random number
```

Constants and Global Variables: This section defines constants (MASTER and Max) and declares global variables (x, y, Scount, Icount, Rcount, and Population) which are used throughout the program.

```
MASTER = 0
Max = 100

x, y = 0, 0
Scount, Icount, Rcount = 0, 0, 0

Population = [['' for _ in range(Max)] for _ in
     range(Max)]

def initPopGrid(pop):
    global x, y
    dbPop = float(pop)
    popRoot = math.sqrt(dbPop)
    x = math.ceil(popRoot)
    y = math.ceil(popRoot)

    # initialize people
    for i in range(x):
        for j in range(y):
            random.seed(time.time())
            randNum = random.randint(1, 3)

            if randNum == 1:
                Population[i][j] = 'S'
            elif randNum == 2:
                Population[i][j] = 'I'
            elif randNum == 3:
                Population[i][j] = 'R'
```

Function calculate: This function simulates the spread of the disease over a given number of days using the SEIR model. It uses the infection rate and recovery rate to update the population's state and prints the fractions of the population in each state for each day.

```
def calculate(days, population):
    dt = 1          # time step in days
    beta = 1 / 5    # infection rate
    gamma = 1 / 14  # recovery rate

    S = [0]*Scount
    I = [0]*Icount
    R = [0]*Rcount

# Initial Populations
    I[0] = Icount/population # init infective
    S[0] = Scount/population - I[0] # initial S
    R[0] = Rcount/population # init recovered

    # print initials
    print(f"Fraction of population susceptible
    to infection at beginning of observation :
     {S[0]}")
    print(f"Fraction of population infected at
    beginning of observation : {I[0]}")
    print(f"Fraction of population recovered at
     beginning of observation : {R[0]}")

    for i in range(days):
        S[i+1] = S[i]-beta * (S[i] * I[i]) * dt
        I[i+1] = I[i] + (beta * S[i] * I[i] -
    gamma * I[i]) * dt
        R[i+1] = R[i] + gamma * I[i] * dt

        # print values
        print(f"Fraction of population
    susceptible to infection at day {i+1}: {S[
    i]}")
        print(f"Fraction of population infected
     at day {i+1}: {I[i]}")
        print(f"Fraction of population
    recovered at day {i+1}: {R[i]}")
```

Function main: This is the main function which is the entry point of the program.

• It begins by checking the number of tasks. If it's less than 2 or greater than 10, it prints an error message and exits.

• Next, it asks for user input for the population size and number of days.

• It initializes the population grid, and divides the population among the tasks.

• If the task is the master task, it prints out a bunch of predetermined values (which doesn't seem to serve any real purpose), sends population portions to worker tasks and receives processed portions back, then runs the calculate function.

• If the task is not the master task, it receives a portion of the population, counts the number of 'S', 'I', and 'R' states in its portion, then sends this data back to the master task.

```
def main():
    global Scount, Icount, Rcount
    numtasks = 2  # set to 2 as a default

    if numtasks < 2:
        print(f"ERROR: Number of tasks set to {
    numtasks}")
        print("Need at least 2 tasks! Quitting
    ...")
        return

    if numtasks > 10:
        print(f"ERROR: Number of tasks set to {
    numtasks}")
        print("Number of tasks must be below
    10. Quitting...")
        return

    popPortion = ['']*x*y

    if MASTER == 0:
        population = int(input("Enter a
    population size between 4 and 100"))
```

Fixed Recovery Time

In this case, we set exactly two weeks as the amount of time to recover from the illness (I to become R).

Not Taken into Account

People exiting the simulation through death count as recovered. In the SIR model, people exit the model in all 3 phases.

The amount of people that exit the simulation are denoted for Susceptible μS, Infected μI and Recovered μR.

$$\begin{tikzcd}
{} \arrow[r, "\mu N"] & S \arrow[r, "\beta SI"] \arrow[d, "\mu S"'] & I \arrow[r, "\gamma I"] \arrow[d, "\mu I"'] & R \arrow[d, "\mu R"'] \\
& {} & {} & {}
\end{tikzcd}$$

VI-A. The Population Model

The population to be used is represented as a grid (2D array), each cell representing one person in the population. The values of the array are randomized so that each cell is randomly assigned a state in the model.

The values in the array will be used in calculations made over the defined observation period. The initial values would be higher; in order to more easily demonstrate the difference in the stages over time, we decided to randomize the values.

The master process creates a certain number of worker processes (input at compilation) and assigns a portion of the array to each process. The master compiles the findings and sets up the calculations of the 3 stages.

VI-B. Population Processing

The worker processes calculate the totals of each stage of the model in their portion of the array and add to the totals.

These are used for the initial values at the start of the observation period.

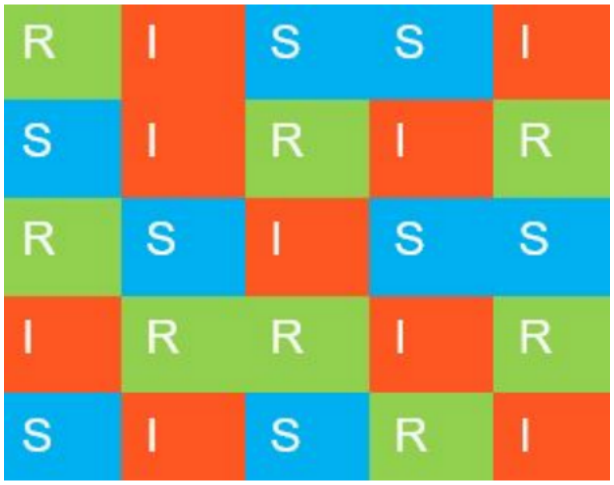
S: $\frac{8}{25}$ Susceptible – 32% of population.

I: $\frac{9}{25}$ Susceptible – 36% of population.

R: $\frac{8}{25}$ Susceptible – 32% of population.

Population totals of people in each stage S, I, and R are calculated every portion and updated.

Sample Assignment Randomizer



R	I	S	S	I
S	I	R	I	R
R	S	I	S	S
I	R	R	I	R
S	I	S	R	I

VI-B1. Population Processing Code

```python
import math
import random
```

Initialize Parameters The code first sets up some initial counts for the number of Susceptible (Scount), Infected (Icount), and Recovered (Rcount) individuals in the population.

It also creates a 2D array, Population, to represent a grid of individuals in the population."

```python
Scount = 480
Icount = 326
Rcount = 194
Population = [[0 for _ in range(100)] for _ in
    range(100)]
```

initPopGrid(pop): This function initializes the Population grid. For each individual in the population, it randomly assigns them a state of being either Susceptible (1), Infected (2), or Recovered (3).

```python
def initPopGrid(pop):
    global x, y
    popRoot = math.sqrt(pop)
    x = math.ceil(popRoot)
    y = math.ceil(popRoot)
    #initialize people
    for i in range(x):
        for j in range(y):
            randNum = random.randint(1, 3)
            Population[i][j] = randNum
```

doWork(x, y): This function goes through the Population grid and updates the counts for the number of Susceptible, Infected, and Recovered individuals based on the current states in the grid.

```python
def doWork(x, y):
    global Scount, Icount, Rcount
    Scount = 0
    Icount = 0
    Rcount = 0
    for i in range(x):
        for j in range(y):
            if Population[i][j] == 1:  #1
    stands for Susceptible
                Scount += 1
            elif Population[i][j] == 2:  #2
    stands for Infected
                Icount += 1
            else:
                Rcount += 1  #else Recovered
```

calculate(days, population): This function simulates the spread of the disease over a given number of days, based on the initial counts of Susceptible, Infected, and Recovered individuals, and the defined infection rate (beta) and recovery rate (gamma). It prints the fraction of the population that is Susceptible, Infected, and Recovered at each day of the simulation.

```python
def calculate(days, population):
    dt = 1.0  #time step in days
    beta = 0.2  #infection rate
    gamma = 0.07  #recovery rate

    S = [0]*480
    I = [0]*326
    R = [0]*194

    I[0] = Icount / population  #initial
infective population
    S[0] = Scount / population - I[0]  #initial
 susceptible population
    R[0] = Rcount / population  #initial
recovered population

    #print initials
    print("Fraction of population susceptible
to infection at beginning of observation:"
, S[0])
    print("Fraction of population infected at
beginning of observation:", I[0])
    print("Fraction of population recovered at
beginning of observation:", R[0])

    for i in range(days):
        S[i + 1] = S[i] - beta * (S[i] * I[i])
* dt
        I[i + 1] = I[i] + (beta * S[i] * I[i] -
 gamma * I[i]) * dt
        R[i + 1] = R[i] + gamma * I[i] * dt

        #print values
        print("Fraction of population
susceptible to infection at day", i, ":",
S[i])
        print("Fraction of population infected
at day", i, ":", I[i])
        print("Fraction of population recovered
 at day", i, ":", R[i])
```

main(): This is the main function that runs the simulation. It defines the total population size and the number of days for the simulation. Then it initializes the Population grid, calculates the initial counts of Susceptible, Infected, and Recovered individuals, and then runs the simulation for the given number of days.

```python
def main():
    population = 1000
    days = 30
    print("Using population size of 1000")
    print("Over 30 days")

    random.seed()
    initPopGrid(population)
    doWork(x, y)
    calculate(days, population)

if __name__ == '__main__':
    main()
```

Note: In the SEIR model, the 'Exposed' category often refers to individuals who have been infected but are not yet infectious themselves. However, in this particular simulation, there doesn't appear to be an 'Exposed' category; instead, individuals are modeled as transitioning directly from 'Susceptible' to 'Infected'.

Equations

S: the fraction of the population that is susceptible

I: the fraction of the population that is infected

R: the fraction of the population that is recovered

•β is the number of contacts per infected persons per day

•γ is the recoveries per person per day

•dt is time, measured in days.

$$\frac{ds}{dt}(\frac{S}{time}) : -\beta SI$$

$$\frac{dI}{dt}(\frac{I}{time}) : \beta SI - \gamma I$$

$$\frac{dR}{dt}(\frac{R}{time}) : I\gamma$$

VI-C. Setting our Constant Values

Our constant values are: dt, β, γ

Constant values in the equation are set.

```
int dt = 1;
// time step in days
int beta = 1/5;
// infection rate
int gamma 1/14;
// recovery rate

double S[], I[], R[];

I[0] = Icount/population;
//initial infective population
S[0] = Scount/population - I[0];
//initial susceptible population
R[0] Rcount/population;
// initial recovered population
```

Calculations

Values for the initial day of the observation period (Stage[0]) set using totals calculated before.

Equations to calculate each stage. Calculations made in a loop to cover the number of days specified by the user.

Additions

For the classic SIR model:

•$\frac{1}{y}$ Days to Recover

•$\frac{\beta}{y}$ Contacts per infection

• People in the model that have recovered may become susceptiple once more.

•μn Additions to the population

•$\mu S, \mu I, \mu R$ People exiting the model

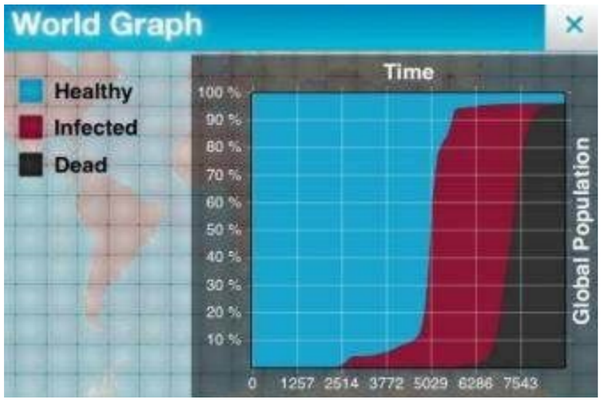
VII. Analysis

Analyzing the results you have achieved through your research

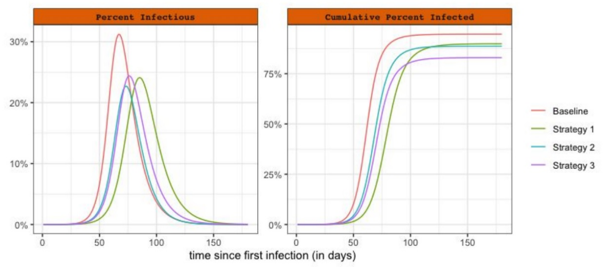
- What do these results mean?
- What are the possible consequences of these results
- Has my thesis statement been satisfied by my research? Why (not)?
- How could these results be different if the methodology was adjusted

Video Games and Modeling

This is a screenshot of a game of Plague, Inc. Notice how this game is a take on the classic SIR model; essentially a race between endemic and epidemic!



Models Derived through Different Strategies



The Discussion section should revisit your main insights gained from your project and how these fit into the ideas presented in the Introduction and Background sections. Be sure to explain how your project has provided some additional insight into the specific idea mentioned at the end of this section. Also, feel free to provide some examples of new directions and ideas where your would like to see this work unfold in the future

APPENDIX A
CORRUPTED BLOOD VIRUS

First let us look at the data split from the COVID-19 virus from Hubei, China.

```
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.model_selection import
    GridSearchCV


def data_spilt(data, orders, start):
    x_train = np.empty((len(data) - start -
    orders, orders))
    y_train = data[start + orders:]

    for i in range(len(data) - start - orders):
        x_train[i] = data[i + start:start +
    orders + i]
```

We exclude the day (Feb. 12, 2020) of the change of the definition of confirmed cases in Hubei China. x$_t rain = np.delete(x_t rain, np.s_[28-(orders+1)-start : 28-start], 0) y_t rain = np.delete(y_t rain, np.s_[28-(orders+1)-start : 28-start])$

return x$_t rain, y_t rain$

```

def ridge(x, y):
    print('\nStart searching good parameters
    for the task...')
    parameters = {'alpha': np.arange(0,
    0.100005, 0.000005).tolist(),
                  "tol": [1e-8],
                  'fit_intercept': [True, False],
                  'normalize': [True, False]}

    clf = GridSearchCV(Ridge(), parameters,
    n_jobs=-1, cv=5)
    clf.fit(x, y)

    print('\nResults for the parameters grid
    search:')
    print('Model:', clf.best_estimator_)
    print('Score:', clf.best_score_)

    return clf
```

REFERENCES

[1] Abbey, Helen (1952). "An examination of the Reed-Frost theory of epidemics". Hum. Biol. 3:201

[2] Blaise Barney, Lawrence Livermore National Laboratory, UCRL-MI-133316

[3] Generalized Kinetic Models in Applied Sciences: Lecture Notes on Mathematical Problems Arlotti, Luisa et al.

[4] An Introduction to Mathematical Modeling of Infectious Diseases; Li, Michael.

[5] www.gapminder.org

[6] Plague Inc. video game, Apple Store

[7] J.D. Lawson and J. Ll. Morris, The Extrapolation of First Order Methods for Parabolic Partial Differential Equations SIAM Vol. 15, No. 6, December 1978

[8] Francisco Santos, Recent progress on the combinatorial diameter of polytopes and simplicial complexes July 24, 2013

[9] Comparative estimation of the reproduction number for pandemic influenza from daily case notification data; Gerardo Chowell, Hiorshi Nishiura and Luis M. A. Bettencourt 12 October 2006

[10] Deep-Data-Driven Neruar Networks for COVID-19 Vaccine Efficacy; Epidemiologia 2021, 2, 564-586

[11] Empirical model for short-time prediction of COVID-19 spreading; PLOS Computational Biology Dec 9, 2020 https://doi.org/10.1371.pobi.1006431

[12] Eric T Lofgren, Nina H Fefferman, The untapped potential of virtual game worlds to shed light on real world epidemics, The Lancet Infectious Diseases, Volume 7, Issue 9, 2007, Pages 625-629, ISSN 1473-3099,
https://doi.org/10.1016/S1473-3099(07)70212-8.
(https://www.sciencedirect.com/science/article/pii/S1473309907702128)

[13] COVID-19 data with SIR model
https://www.kaggle.com/code/lisphilar/covid-19-data-with-sir-model/notebook

[14] 24C3: Modelling Infectious Diseases in Virtual Realities
https://www.youtube.com/watch?v=u3eLSkUfw-M

[15] Center for Disease Control and Prevention SARS Basics Fact Sheet
https://www.cdc.gov/sars/about/fs-sars.html

[16] https://www.wired.com/2008/03/wow-terror/ Wired: World of Warcraft Shines Light on Terror Tactics

[17] The Emergence of Disease in Early World-Systems: A Theoretical Model of World-System and Pathogen Evolution; Anthony Roberts Department of Sociology and Institute for Research on World-Systems (IROWS)
University of California, Riverside
https://irows.ucr.edu/papers/irows62/irows62.htm#_ftnref2

[18] Padé approximation
Padé approximation (Brown)