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

Andrew Becker

Middle Tennessee State University, Murfreesboro, Tennessee, USA

***Abstract*— Employing 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 based on cellular automata. Cellular automata have 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 popular models of infectious diseases. 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.**

Keywords*— *Epidemic Chain Models, Markov Chain (as alternative to), Message Passing Interface (MPI), OpenMP, Parallel Processing, Plague Inc., Pthreads, Kermack-McKendrick Model, SIR Model, Polyhedra, diameter, Hirsch conjecture, simplex method, simplicial complex, deep learning; vaccination strategy; Corrupted Blood, World of Warcraft; Infectious disease epidemiology, influenza, mathematical modelling, Susceptible-Infected-Removed (SIR) model

Note that the Background and Literature Review are partly GPT-4 assisted.

I. Introduction

THE Spanish Flu, also known as the H1N1 virus, "killed more people than in a shorter period of time than any other disease in human history." [1] This prompted the first global health crisis in 1918, with a higher death toll than the entirety of the First World War. This horrific event also marked the beginning of modern intervention programs, where technology allowed countries to coordinate efforts to share data in ways that were previously impossible. It was the dawn of epidemiological modeling, with quantifiable infection rates and casualty numbers that are still being used in research to this day. [2,3] We look to the past to predict and prevent the next pandemic, using deep-learning and virtual worlds to simulate new mitigation strategies for the future.

Mathematical Modeling of Infectious Diseases

Varying systems of differential equations are used to study the dynamics of an epidemic, with the base mathematical model used in this work dividing the population into three groups, Susceptible (S), Infected (I), and Recovered (R).

Extensions of this SIR model have been applied to the Spanish Flu in the past [4], while recent studies on intervention efficacy require more effective deep-learning architectures. One of the main challenges during a pandemic event is ascertaining an epidemiological model's parameters, so we will be using the least square method to determine reasonable values for constant and time-dependent parameters. [5]

Kermack-McKendrick Model

The Kermack-McKendrick theory was presented in a series of papers from 1927 to 1933, [6,7,8] a theory meant to predict the path of an infectious disease moving through a populace, calculating the distribution of infections over time. It was the origin of the SIR model, compartmentalizing individuals into groups based on a closed population. It was designed to explain the rapid fluctuations in the number of infections observed in deadly epidemics like the bubonic plague and cholera, but was later modified to support research on endemic diseases. The model presumes a homogeneous population (no age, spatial or social structures), a fixed population size (no births, no deaths due to disease or natural causes), and a disease with an instantaneous incubation period that stays infectious for its entirety. The Kermack-McKendrick model is made up of three coupled non-linear ordinary differential equations,

The model consists of a system of three coupled non-linear ordinary differential equations,

$$\frac{dS}{dt} = -\beta SI \tag{1}$$

$$\frac{dI}{dt} = \beta SI - \gamma I \tag{2}$$

$$\frac{dR}{dt} = \gamma I, \tag{3}$$

where t is time,
S(t) is the number of susceptible individuals,
I(t) is the number of infected individuals,
R(t) is the number of recovered individuals, who have developed immunity to the infection,
beta is the infection rate,
and gamma is the recovery rate.

The key value governing the time evolution of these equations is the epidemiological threshold,

$$R_0 = \frac{\beta S}{\gamma} \tag{4}$$

The value R_0 is identified as the number of secondary infections caused by a primary infection, or the number of individuals infected by contact with a singular individual before death or recovery.

If R_0 is less than 1, each new infected individual will infect less than one other individual, and the outbreak will eventually disappear:

When $R_0 < 1$, infected individuals will spread the disease to more than one other individual, and the outbreak will continue to grow:

$$\frac{dI}{dt} < 0. \tag{5}$$

When $R_0 > 1$, each person who gets the disease will infect more than one person, so the epidemic will spread:

$$\frac{dI}{dt} > 0. \tag{6}$$

R_0 is used in nearly all models of epidemiology, although the equivalence of $R_0 = \frac{\beta S}{\gamma}$ only applies to the basic Kermack-McKendrick model. These early models used a concept known as 'threshold density', where a critical number of susceptible people determines whether or not an epidemic will form. Kermack and McKendrick eventually updated their basic model to include endemic diseases, adding formulas for births and deaths to open their originally closed system.

Later models, especially those created by Anderson and May, would improve on the functionality of Kermack-McKendrick by including cases of secondary infections, vaccination, population transfer, and life expectancies. [9] Modern epidemics such as HIV or COVID require greater emphasis on societal structure and behavior analysis [10,11], so we further modify the models to more closely depict the specific biology of any given disease.

Threshold Theorem of epidemiology

Let $S_0 = p + v$ and assume that v/p is vert small compared to one. Assume moreover, that the number of initial infectives I_O is very small. Then, the number of individuals who ultimately contract the disease is $2v$. In other words, the level of susceptibles is reduced to a point as far the below the threshold as it was above it.

The Corrupted Blood Virus Incident

Questions have arisen over the continued usefulness of the basic SIR model, as the real world isn't a closed system, and deadly pandemics don't simply move people from Infected status to Recovered. But what if there was such a world, where infections and even death can be fixed with a click of a button? [12] This brings us to the world of Azeroth, the fictional location of the massively multiplayer online roleplaying game (MMORPG) known as World of Warcraft (WoW).

Players of WoW log in to individual servers where their player-states are maintained, creating hundreds of closed-world settings to study threshold density. There are millions of players who do not reproduce, and 'resurrect' after a 6 minute delay. It is almost the perfect example for an SIR model, but how could such a world be used for epidemiological research?

Corrupted Blood is a fictional disease status introduced on September 13, 2005 in World of Warcraft. [12] Originally designed to exist in only one contained level and to expire quickly, programming errors caused the status to affect the pets of the player community, in the second example of a virtual zoonotic (has jumped from a non-human animal to humans) pandemic. (In 2000, The Sims was infiltrated by a virus spread by an in-game Guinea Pig, killing entire neighborhoods as an unintentional side effect of a programming bug [13,14].)

One unforeseen disease vector would've been enough, but further problems were caused by human behavior. High-level players would attempt to escape reinfection by teleporting away to heavily populated town centers. Due to the high difficulty rating of the level that originated Corrupted Blood, once it breached containment, thousands of unaware players were killed in an instant. [15] Not only was the disease moving from pet to player, but also to the merchants and guards that carry invincibility status. This status made them perpetually immune carriers, forcing harsher mitigation strategies.

As towns filled up with corpses, slowing the game to an unplayable rate, the player-base began quarantining the infected, creating immunization clinics, and then finally evacuating population centers entirely. None of their mitigation strategies worked, and even when the developers stepped in to patch the Corrupted Blood code itself, it still failed to end the virtual pandemic. [16] As a last resort, the designers of WoW chose to restart their hosting servers with a version of the game that pre-dated Corrupted Blood. In the real world, stopping a pandemic isn't as simple as rolling the code back, but there are many parallels to disease vectors and infection rates that make MMORPGs and other virtual worlds an important tool to design better mathematical models and observe human responses to outbreaks. [17]

II. Background

THE aim of our project is to leverage 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 statistical techniques like Markov Chains, to develop more practical tools for simulating disease spread. The project will explore the use of machine learning for model refinement and seeks potential applications of these models in virtual systems like city planners, online roleplaying worlds, and even epidemic simulators.

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. 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. [4,5,6]

Project Description & Scientific Problem

The scientific problem being addressed in this project is that current simulation techniques for epidemic models can be computationally intensive and might not scale efficiently. Our challenge is to refine these models to better fit real-world observations without over-fitting, specifically the SIR and SEIR models, using parallel processing techniques and cellular automata. Ultimately, the project seeks to develop more practical tools for simulating disease spread. [10, 11]

Scientific Conclusions

Through this project, we hope to achieve several scientific conclusions:

1. Explore the practicality and applicability of these simulation tools beyond academic research, such as their use in public health planning or game development.

2. Provide insights into the potential integration of machine learning techniques for automatic model refinement and adaptive simulation techniques.

3. 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. [18]

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.

Algorithms & Numerical Methods

In this project, 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 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 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.

We then present the results and discussion of data-driven simulations along with error metrics.

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.

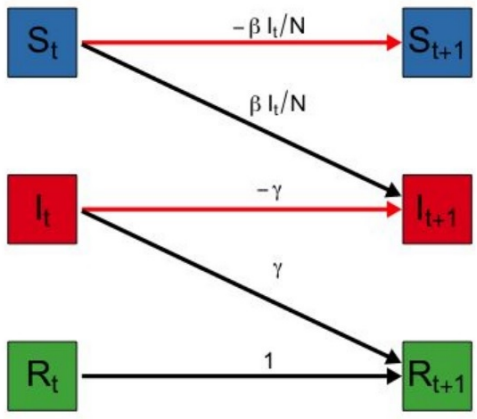


Figure 1. A block diagram of the baseline SIR system

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 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 Models: We will be writing functions or classes to represent the SIR 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 [5]

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.

III. Literature Review

COMPARTMENTAL models are useful for providing broad views of potential disease spread and intervention methods, while agent-based models simulate state changes and can use specific behaviors and environmental factors for analyzing more discrete scenarios.

Virtual worlds offer a special opportunity to study infections in a controlled environment, combining both types of model as a human-influenced agent system, with more unpredictability based on realistic motivations like emotion and social dynamics. [19]

This unpredictability is always a part of real-life pandemics, but is difficult to intentionally model. The Corrupted Blood incident was a unique opportunity to observe human behaviors like fleeing infected areas, asymptomatic carriers, and the lack of preparedness of authority figures. When we integrate the insights we gain from unexpected events into comprehensive epidemiological models, we can more properly prepare strategies for real-world outbreaks. [20, 21]

The SIR model is more characterized by a disease like measles, where once infected or vaccinated, immunity is lifelong. When the 'Exposed' status is added to create the SEIR model, this extends a group of individuals that are incubating a virus without being infectious, modeling viruses like COVID-19. When immunity is short-term, allowing for recovery back to the susceptible state, this creates the SIRS model, which is more representative of something like Norovirus.

The Corrupted Blood virus is best modeled by the SIS model that allows for reintroduction of individuals to the susceptible state. Whether or not WoW characters survive Corrupted Blood, there is no immunity conveyed, so the only way to take individuals out of the system is to quit the game. [12]

The effective contact rate for disease is measured as the total number of contacts multiplied by the transmission risk, or the probability of contracting the disease from an infected. For Corrupted Blood, the transmission risk is 1, meaning that contact rate is equivalent to the number of total number of individuals in proximity. Since Corrupted Blood only lasts for 10 seconds and infects all within 30 feet, we have strict measurements to determine contact rate, which makes Corrupted Blood 3 times more contagious than measles, one of the most infectious diseases ever known. [22]

WoW servers have on average 3,000 - 5,000 players, with a concurrent 8.5 million monthly players over the whole world. An instance of Corrupted Blood would take roughly 20 percent of a high-level character's health, intended to be taxing, but not deadly. For low-to-medium level characters, Corrupted Blood would be instantly fatal. [16]

With such a high fatality rate and very short infection time, basic models that use homogeneous mixing(where each individual has an equal chance of contact), would predict a quick rise to pandemic status, and equally quick fade away However, since there's no permanent death in WoW, players act more recklessly and in some cases, intentionally spread the disease. Even in a virtual world, populations have strata with different numbers of contacts per individual. This led to an unpredictable super-spreader event, with individual merchants and guards infecting many others without ever changing their infection status. [22]

Padé Approximation Units

The Taylor series is a representation of a function as a sum of powers in a variable or function, specifically an expansion of a function around a given point. When attempting the best approximation of said function in a rational function where the Taylor series does not converge, a Padé approximation is preferred, using a ratio of polynomials rather than truncating the Taylor series. This is especially preferred when capturing non-linear behavior, which is why despite being over a century old, they are currently strongly desirable in neural networks. These rational functions, known as Padé Activation Units (PAU), can be used to design new robust algorithms with similar performance to the most powerful deep networks when learning new activation functions. Considering each novel virus requires a different training net, evaluating new variants and mutations make it necessary to replace and retrain on new datasets. [23, 24]

The Hirsch conjecture

Despite the current mathematical consensus that the conjecture is generally false, the Hirsch conjecture, put forth in 1957, claimed that for an edge-vertex graph of a d-dimensional, n-facet polytope may not have a diameter larger than n – d. There is still value in the study, as later relaxations of the theorem stated that the conjecture holds true only if it holds for all simple polytopes.

However, as Francisco Santos Leal proved in his 2011 counter-example, this could be disproven with a polytope in which every facet of said polytope contains exactly one of a pair of distinct vertices. The known lower bounds are linear for very large values of n and d, while the known upper bounds are very mildly sub-exponential, and there is still no known upper bound for either polynomial or linear diameters, so by generalizing study to only bounded complexes of simplicial polytopes, we may consider richer data in our neural networks that may one day break the linear barrier. [25]

Cellular Automata and Markov Chains

Markov chains are used extensively in diverse fields ranging from statistics to physics, and notably gained prominence in machine learning, [26] so we will give a basic example here:

Markov Chain Example

Suppose there is a movie theater that only plays three types of movies: comedies, dramas, and cartoons. However, on any given day, they can show one type of movie, depending on what they showed the day before.

If we know today's movie, we can predict tomorrow's with a certain probability. For instance, if it's a drama day today, there's a 60% chance they'll show a cartoon tomorrow. This is depicted using weighted arrows indicating probabilistic transitions from one state to another, with the sum of all probabilities (weights on the arrows) exiting any state adding to 100%.

So if today is drama day, there's also a 20% chance a drama will be shown again tomorrow, being represented by a self-looping arrow on the drama. Each arrow represents a transition between states, together forming what is known as a Markov chain. The most important property is that the future state relies only on the present state and not on any previous ones.

If the theater shows a comedy on Day 1, drama on Day 2, and a cartoon on Day 3, to predict Day 4's movie, we only need to consider Day 3. If it was a cartoon day, there's a 70% probability they'll show a comedy next. This is the essence of Markov chains, referred to as the Markov property.

Markov Chain Walk

We will simulate a random walk along our Markov chain, starting with a comedy day. After observing the restaurant for ten consecutive days, we can calculate the probabilities associated with each type of movie by counting occurrences. To determine if these probabilities stabilize or keep fluctuating in the long run, we created a Python script that simulated 100,000 steps and found that the probabilities eventually stabilize, reaching what we term the 'stationary distribution' or equilibrium state. A more efficient way to decipher the stationary state involves linear algebra, representing the Markov chain as an adjacency matrix, known as the transition matrix.

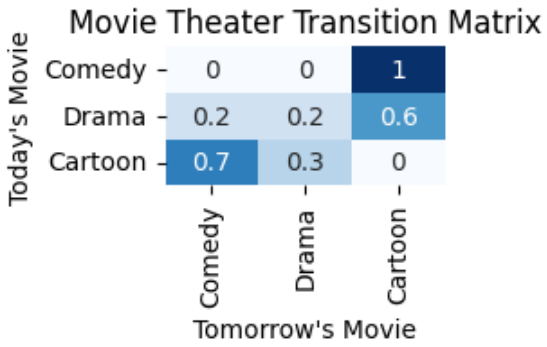
To find the probabilities associated with each state, we can use a row vector, denoted as pi. This vector multiplied by our transition matrix gives us future probabilities. If a stationary state exists, the input and output vectors will eventually become identical. Solving these equations, we can predict that the theater shows comedies 35% of the time, dramas 21%, and cartoons 46%.

Markov Chain Code

```
import numpy as np

# Define the states
states = ["Comedy", "Drama", "Cartoon"]

# Transition matrix:
# Row indices correspond to today's movie and column indices to tomorrow's movie.
# E.g., transition_matrix[0][1] gives the probability of transitioning from Comedy (row 0) to Drama (column 1).
```

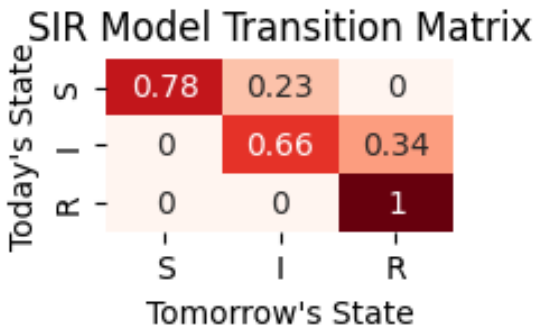
Movie Theater Example Transition Matrix

```
transition_matrix = [
    # Comedy, Drama, Cartoon
    [0.0, 0.0, 1.0],   # Comedy (from the info, if today is a Comedy, tomorrow is guaranteed to be a Cartoon)
    [0.2, 0.2, 0.6],   # Drama
    [0.7, 0.3, 0.0]    # Cartoon
]
```




```
def predict_next_movie(current_movie):
    return np.random.choice(states, p=transition_matrix[states.index(current_movie)])

# Simulate for a given number of days
def simulate_days(start_movie, days=10):
    current_movie = start_movie
    movie_sequence = [current_movie]
    for i in range(days-1):
        next_movie = predict_next_movie(current_movie)
        movie_sequence.append(next_movie)
        current_movie = next_movie
    return movie_sequence

# Test
days_to_simulate = 10
starting_movie = "Drama"
print(simulate_days(starting_movie, days_to_simulate))
```

SIR Transition Matrix



Note that the sums of each column add up to 100%, and once someone has reached the Recovered state, they don't have anywhere to go in this simplified closed model.

Cellular Automata

Cellular automata are algorithmic models that use computation to iterate on very simple rules, in so doing these very simple rules can create complex emergent phenomena through the interaction between agents as they evolve over time. [26] To illustrate the functioning of a cellular automaton we will take an example from probably the most famous algorithm called the Game Of Life devised by the mathematician John Conway.

Conway's Game of Life

The Game Of Life is played on a grid of square cells. A cell can be live or dead, a live cell is shown by putting a mark on its square, a dead cell is shown by leaving the square empty, each cell in the grid has a neighborhood consisting of all adjacent cells to it and there are just three rules governing the behavior of an agent. [27]

1. Any live cell with fewer than two live neighbors dies, as if caused by under- population.

2. Any live cell with two or three live neighbors lives on to the next generation.

3. Any live cell with more than three live neighbors dies, as if by overcrowding.

4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

As we input a starting condition and run the program to see what we get, this pattern is called still life for obvious reasons, its product is probably the most simple class of pattern, called class one, where nearly all of these patterns evolve quickly into a stable, homogeneous state and any randomness in the initial pattern disappears.

The second class of pattern we may get is where the system evolves into an oscillating structure. The simplest of these being a blinker that has a period two oscillation. We can also have oscillating structures that cycle over prolonged periods of time for example a pulsar has a period three oscillation, but oscillators of many more periods are known to exist.

Class three patterns are random where nearly all initial patterns evolve in a semi-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise. Local changes to the initial pattern tend to spread indefinitely. Here we can get what are called gliders where a group of cells appear to glide across the screen and this is a good example of emergence as we no longer see the simple rules that are producing them but instead this emergent structure of an object gliding.

Lastly automata can also produce patterns that become complex and endure over a prolonged period of time, with stable local structures. With these more complex patterns cellular automata can simulate a variety of real-world systems, including biological and chemical ones.

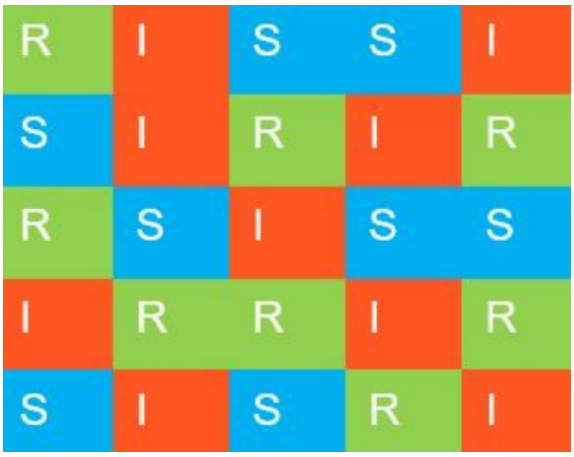


Diagram of Cellular Automata [28]

Using Cellular Automata in the Place of Markov Chains

Most of the literature available in the field of CA and/or Markov chain analysis in the geospatial domain can be broadly classified into two groups – one being those which discuss various methodologies and their adaptations and variations, and another being the application of these methodologies in the real or sometimes, hypothetical world. First of all, each and every category is classified according their domains of applications in a tree-like diagram and the content of their works is

IV. Methods • Games and Graphs

MATHEMATICAL modeling is 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.

Worker Process Function

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.

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 popular game, 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. [1, 6]

Indeed it is a race against competing model curves that was actually used by the World Health Organization in a public awareness campaign to fight COVID-19. Bringing together the minds behind the Coalition for Epidemic Preparedness Innovations (CEPI) and the Global Outbreak Alert and Response Network (GOARN), they developed a variant called Plague Inc.: The Cure, where players are tasked with mitigating and fighting a virus, while educating the gaming community on preventing misinformation. [7]

Disease	R_0
Measles	15 to 20
Foot and Mouth Disease	Initially 8.4 (Reduced to 1.3 with Animal Movement Restrictions)
Influenza	< 3
Smallpox	3 or 4
HIV	Between 2 and 12 (with condoms, << 1)
Corrupted Blood	D x c x p = 10 sec x c x 1 50 (city: 5 contacts per second countryside: one every 20s)

Table 1. R_0 of some diseases [22]

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. [29]

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. 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 [30] (since everyone is in the model when they aren't transitioning.)

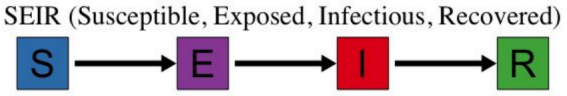
Overview of Relevant Literature

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.

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. [31]

One of the products of the SARS epidemic of 2002-2003 was a variety of epidemic models including general contact rates, quarantine, and isolation. [32] These models can be viewed as age of infection epidemic models and analyzed using the approach of the full Kermack-McKendrick Model.

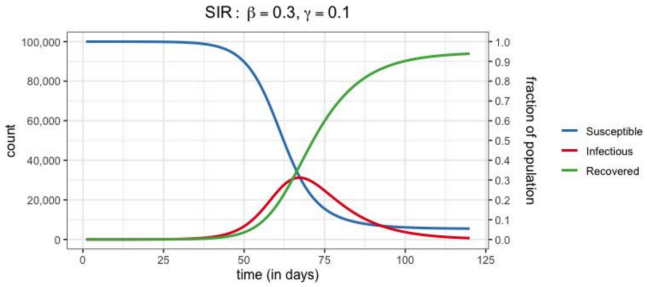
All of these models share the basic properties that there is a threshold between disappearance of the disease and an epidemic outbreak, and that an epidemic will die out without infecting the entire population. [6, 7]

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.

	Measles	**Syphilis**	**Norovirus**	**Corrupted Blood**
Incubation period	10 days	1-3 weeks	10 hours	0 sec
Duration of infectiousness	8 days	1 year	2-4 days	10 sec
Transmission probability p	95%	30%	high	100%
Immunity	yes, lifelong	weak	only against subtype	no
Mode of Transmission	Droplet (airborne)	Sexually	Fecal-Oral, droplet, fomite	magic (droplet like)

Table 2. Disease Parameters [22]

V. 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, [18] 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.

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. [33]

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"))
```

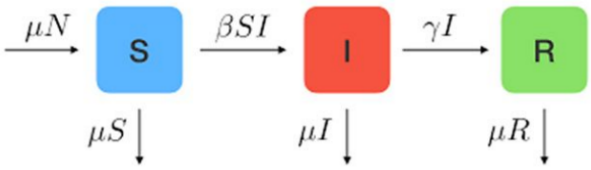
Fixed Recovery Time

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.



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.

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

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, the number of contacts per infected persons per day
γ , the recoveries per person per day
dt, the time measured in days

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

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

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

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 susceptible once more.

• μn Additions to the population

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

Population Processing Code

```
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."

```
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).

```
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.

```
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.

```
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])
```

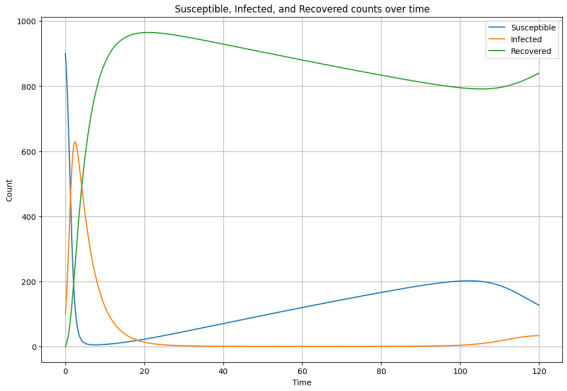
main(): This is the main function that runs the simulation. It defines the total population size and the number of days for the simulation. Then it initializes the Population grid, calculates the initial counts of Susceptible, Infected, and Recovered individuals, and then runs the simulation for the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

```
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'.

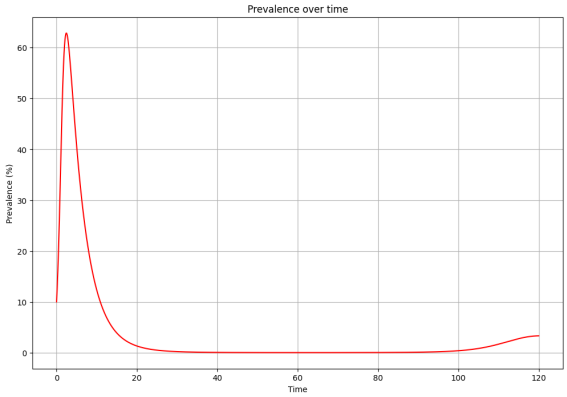
Analysis

Corrupted Blood has a very high infection rate, and this has the effect of the model spiking, or having the intersection of the S, I, and R curves very early in its spread.



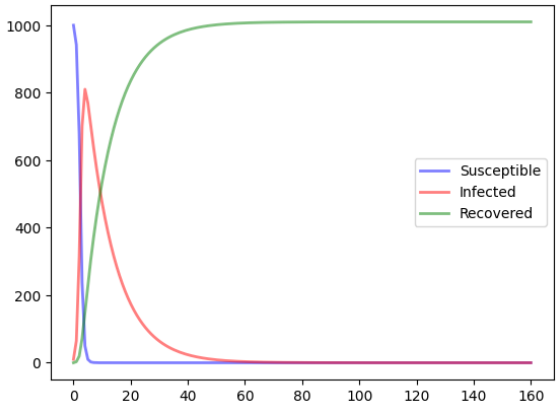
Results of RK4

Further, the prevalence of the virus appears to burn through its population quickly. However this model does not allow for the respawning agents. In the game, players will be added again to a new Susceptible curve, indicating we will need to consider this is a future model. This is similar in real-life to people entering an area from one that was previously unaffected.



Prevalence Over Time

Adjusting our methodology to account for these parts of the models that may not normally appear in real life can lead us to discoveries in a nonstandard manner of thinking about these functional relations, and that we may find ways to model diseases in the real-world, like the Spanish Flu, under this new functional vocabulary.



Results of Cellular Automata

Conclusion

This exploration into using Cellular Automation in the place of Markov Chains can lead us to begin to draw conclusions about diseases like the Spanish Flu whose data may be permanently corrupted by time and the manner in which it was collected, as well as modern disease modeling with SARS, COVID-19, and HIV.

By studying simulation models, we are exploring avenues not yet considered, and developing and language, both in technology but also in popular understanding.

Future Work

Next steps are certainly refining these models and comparing to the real-life data both of COVID-19 as we have, and in the approximations in the spotty data in the Spanish H1N1 Flu that created the need for research of this type.

A study into functional homeomorphic behavior would give research a greater facility to work with in generating much more sophisticated models.

Using nonstandard analysis, for example Surreal Number computation, can give us deeper understanding into the deep structure of our world. As they have been used in other realms like quantum analysis.

REFERENCES

[1] The Great War, "The Spanish Flu I THE GREAT WAR Epilogue 3," YouTube, Dec 3, 2018. [Video]. Available: https://www.youtube.com/watch?v=XRtXckhuquo.

[2] L. Arlotti et al., Generalized Kinetic Models in Applied Sciences: Lecture Notes on Mathematical Problems. World Scientific, 2003

[3] M. Li, An Introduction to Mathematical Modeling of Infectious Diseases. Springer, 2018

[4] G. Chowell, H. Nishiura, and L. M. A. Bettencourt, "Comparative estimation of the reproduction number for pandemic influenza from daily case notification data," Journal of the Royal Society Interface, pp. 155-166, Oct. 12, 2006.

[5] T.K. Torku, A.Q.M. Khaliq, K.M. Furati, "Deep-Data-Driven Neural Networks for COVID-19 Vaccine Efficacy," Epidemiologia, vol. 2, pp. 564-586, 2021

[6] W. Kermack and A. McKendrick, "Contributions to the mathematical theory of epidemics – I," Bulletin of Mathematical Biology, vol. 53, no. 1–2, pp. 33–55, 1991.

[7] W. Kermack and A. McKendrick, "Contributions to the mathematical theory of epidemics – II. The problem of endemicity," Bulletin of Mathematical Biology, vol. 53, no. 1–2, pp. 57–87, 1991.

[8] W. Kermack and A. McKendrick, "Contributions to the mathematical theory of epidemics – III. Further studies of the problem of endemicity," Bulletin of Mathematical Biology, vol. 53, no. 1–2, pp. 89–118, 1991.

[9] R.M. Anderson, Bulletin of Mathematical Biology, vol. 53, no. 1/2, pp. 3-32, 1991.

[10] M. Català, S. Alonso , E. Alvarez-Lacalle, D. López, P. Cardona, C. Prats "Empirical model for short-time prediction of COVID-19 spreading," PLOS Computational Biology, Dec. 9, 2020. [Online]. Available: doi.org/10.1371/journal.pcbi.1008431

[11] Lisphilar, "COVID-19 data with SIR model," Kaggle, December 28, 2020. [Online]. Available: www.kaggle.com/code/lisphilar/covid-19-data-with-sir-model/notebook

[12] "Corrupted Blood (debuff)," Wowpedia, May 11, 2023. [Online]. Available: https://wowpedia.fandom.com/wiki/Corrupted_Blood_(debuff)

[13] World Health Organization, "Zoonoses," WHO, July 29, 2020. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/zoonoses

[14] J. Markoff, "Something Is Killing the Sims, and It's No Accident," New York Times, Apr. 27, 2000.

[15] "Corrupted Blood," Wikipedia, Nov. 16, 2007. [Online]. Available: http://en.wikipedia.org/wiki/Corrupted_Blood

[16] S. Messner, "How Blizzard coped with World of Warcraft's blood plague and other early disasters," PC Gamer, Aug. 26, 2019.

[17] D. Thier, "World of Warcraft Shines Light on Terror Tactics," Wired, March 20, 2008. [Online]. Available: www.wired.com/2008/03/wow-terror/

[18] B. Barney, Lawrence Livermore National Laboratory, UCRL-MI-133316.

[19] E. T. Lofgren and N. H. Fefferman, "The untapped potential of virtual game worlds to shed light on real world epidemics," The Lancet Infectious Diseases, vol. 7, no. 9, pp. 625-629, 2007.

[20] "Plague Inc.," Ndemic Creations, PC/Mac/Google Play/App Store, May 2016. Available: www.ndemiccreations.com/en/42-store

[21] World Health Organization, "Experts and gamers join forces to fight COVID-19 via Plague Inc: The Cure game," YouTube, March 31, 2021. [Online]. Available: www.youtube.com/watch?v=MTuGH7CNm-k

[22] Chaos Communication Camp 2011, "24C3: Modelling Infectious Diseases in Virtual Realities," YouTube, February 26, 2011 [Online]. Available: www.youtube.com/watch?v=u3eLSkUfw-M

[23] Alejandro Molina, Patrick Schramowski, Kristian Kersting, "Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks," ICLR 2020, February 4, 2020

[24] Applied Differential Equations. The Primary Course by Vladimir Dobrushkin, CRC Press, 2022 "Pade Approximations," Brown University, [Date of Access]. [Online]. Available: https://www.cfm.brown.edu/people/dobrush/am33/Mathematica/pade.html

[25] F. Santos, "Recent progress on the combinatorial diameter of polytopes and simplicial complexes," TOP: An Official Journal of the Spanish Society of Statistics and Operations Research, Springer; vol. 21(3), pp. 426-460, Jul. 24, 2013.

[26] P. Ghosh, A. Mukhopadhyay, A. Chanda, P. Mondal, A. Akhand, S. Mukherjee, S. K. Nayak, S. Ghosh, D. Mitra, T. Ghosh, and S. Hazra, "Application of Cellular automata and Markov-chain model in geospatial environmental modeling- A review," Remote Sensing Applications: Society and Environment, vol. 5, pp. 64-77, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352938516300258.doi: https://doi.org/10.1016/j.rsase.2017.01.005.

[27] J. Conway, R. Guy, The Book of Numbers, New York: Copernicus, 1995

[28] "Simple but Stunning: Animated Cellular Automata in Python" iSquared Digital, May 2, 2021. [Online]. Available: https://isquared.digital/blog/2021-05-02-cellular-automata/

[29] H. Abbey, "An examination of the Reed-Frost theory of epidemics," Hum. Biol., vol. 3, pp. 201, 1952.

[30] D. Smith and L. Moore, "The SIR Model for Spread of Disease - Introduction," Convergence, Dec. 2004.

[31] D. Nykamp, "Introduction to an infectious disease model, part I," YouTube, Jan 10, 2013. [Video]. Available: https://www.youtube.com/watch?v=XWXqXzAYe4E

[32] Centers for Disease Control and Prevention, "SARS Basics Fact Sheet," December 6, 2017 [Online]. Available: www.cdc.gov/sars/about/fs-sars.html

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

[34] H. Hu, K. Nigmatulina, and P. Eckhoff, "The scaling of contact rates with population density for the infectious disease models," Mathematical Biosciences, vol. 244, no. 2, 2013.

[35] J. Burke, Connections. London: McMillan London Limited, 1978.

[36] D. Knuth. Surreal Numbers, New York: Addison-Wesley Publishing Company, 1974

[37] Gapminder Foundation, World Health Chart, 2021 Available: www.gapminder.org/fw/world-health-chart/