University of Amsterdam

Stochastic and Spatial Models of Epidemics

Stochastic and Spatial Models of Epidemics

October 28, 2024

Students:
Victoria J Peterson
15476758

Wessel T L Beumer
12640662

mentor:
Dr. M.H. (Mike) Lees

Course:
Introduction to Computational Science

1 Abstract

This report explores epidemic modeling by extending the traditional Susceptible, Infected, Recovered model to incorporate stochastic processes and spatial modeling. In order to address the limitations of deterministic models in accounting for random fluctuations, particularly in smaller populations where stochastic events can significantly alter outcomes, this study applies Gillespie's Direct Algorithm to add stochasticity to the SIR framework. The resulting stochastic SIR model captures key characteristics of random processes in disease dynamics, including variability in simulation outcomes, transient fluctuations that deviate and/or return to deterministic attractors, harmonic resonance effects over deterministic oscillations, and potential for disease extinction events even when an epidemic might otherwise persist in a deterministic context.

Furthermore, this study employs spatial models using network-based structures to simulate social connectivity and its influence on disease transmission, considering a regular network and four structured network types: scale-free Barabási-Albert, small-world Watts-Strogatz, random Erdős-Rényi, and a real-world Sociopatterns dataset. By analyzing network attributes such as node degree distribution, clustering coefficient, and network diameter, the report illustrates how variations in network topology can distinctly influence the spread and duration of disease transmission within a network – specifically, the structural differences between networks affect the pathways available for infection to propagate, with highly connected nodes (hubs) in scale-free networks, short path lengths in small-world networks, and uniform connection probabilities in random networks each contributing to different outcomes.

The report further investigates several vaccination strategies to determine their impact on reducing epidemic spread within real-world networks. By simulating random vaccination, high-degree node vaccination (targeting highly connected individuals), and dynamic vaccination strategies that adapt based on testing and connectivity, the study evaluates which approaches offer the most effective containment under various vaccination and testing parameters. Findings reveal that targeted vaccination, particularly in nodes with higher connectivity, can substantially limit epidemic spread, emphasizing the significance of tailoring vaccination strategies to the network structure in real-world scenarios, supporting the need for stochastic and spatial models to better inform intervention planning.

0 (errors:1) words

Contents

2 Introduction

Since its introduction, the SIR model has been pivotal in the area of epidemic modelling. [1] The model is able to show the progression of an epidemic wave, despite the simplicity of the model. Over the last few decades, however, there has been a growth of both a demand for more complex models, which agree more with reality, and available computational power. This has led to the development of more advanced variants of the SIR model, which are able to model certain effects not present in the standard model. In this paper, we will evaluate two of such models: a stochastic model, based on Gillespie's Direct Algorithm and a stochastic, network based model.

First, we will explain how the normal SIR model works, the limitations of this model and how it can be adapted to account for some of these limitations, by introducing stochastic effects and networks. Second, we will show how we have implemented both stochastic and network based models. After this, we will analyze how a SIR model build on Gillespie's algorithm behaves differently from the deterministic model. We will also explore how diseases spread through both randomly generated networks and networks derived from real world data. Finally, we will design a vaccination strategy against a disease in the real world model and analyze how this strategy performs.

3 Theory

In the field of epidemic modelling, many methods originate from the SIR model, originally introduced by Kermack and McKendrick in 1927. [2] In this model, a population of size N is divided into three groups; group X, who are Susceptible to the disease; group Y, who are Infected and can transmit the disease to the susceptible population; and group Z, consisting of individuals who have Recovered from the disease. The change in population of these groups can then be expressed as

$$\frac{dX}{dt} = \mu N - \frac{\beta XY}{N} - \mu X \tag{3.1}$$

$$\frac{dI}{dt} = \frac{\beta XY}{N} - \gamma Y - \mu Y \tag{3.2}$$

$$\frac{dR}{dt} = \gamma Y - \mu Z, \tag{3.3}$$

where β is the rate of transmission, γ is the rate of recovery, and μ is both the birth and death rate, which add demography into the model. [1] This model can show how a new disease can transition from an initial epidemic to an endemic state.

While this simple model can be useful in certain applications, it contains several simplifying assumptions, of which we will focus on two in this paper. The first simplification is that of determinism: the basic SIR model completely disregards any possible stochastic effects in the evolution of the disease. This has an impact on modeling diseases in smaller populations, as random effects can lead to drastically different outcomes in these cases. The second simplification is the ignorance of social structures. In the basic SIR model, an infected individual has an equal chance to infect any other susceptible individual. This ignores how diseases usually tend to spread between friends or colleagues, as disease transmission requires physical proximity. To work around these simplifications, this paper will explore two alternative SIR models, which attempt to rectify these shortcomings.

3.1 Gillespie's Direct Algorithm

Gillespie's Direct Algorithm can be used to convert the deterministic SIR model into a model driven by stochastic effects. Gillespie's Direct Algorithm is a discrete event models, meaning that changes in variables such as the amount of susceptible or infected are considered as discrete events, rather than gradual changes over time. [1] These events can be described with the resulting change in population of the X, Y and Z classes, and their rate of occurrence, which can be taken from Equation 3.

A Transmission event, for example, would state that the X class would loose one individual, the Y class would gain one, and the rate of occurrence of the event would be

$$\frac{\beta XY}{N}. \tag{3.4}$$

When the rates for all possible events have been calculated, the total of all rates R_{total} is calculated, which is used to determine the size of the next time step

$$\delta t = \frac{-1}{R_{\text{total}}} \log(RAND_1). \tag{3.5}$$

An event is then chosen with a weighted random choice, using the rates of the events as weights. This process can then be repeated, until a given time t.

3.2 Network models

In network models, the structural features of a network plays an integral part in how nodes interact with one another by influencing connectivity between nodes such that distance, routing, and robustness become important aspects of the system.

The first of these features is the Degree of individual nodes, which is simply the number of edges connected to a node from other nodes such that a higher degree means the node has many neighbors connected to it. The overall network can be described with a degree distribution which provides insight into how many nodes in the network have how many degrees.

Related closely with the Degree of a node is the Clustering Coefficient, which is a measure of the proportion of neighboring nodes which are also connected to one-another, a value that more visually represents the small groups of highly connected nodes in a network.

Diameter is a feature that generally depends on if the network is fully connected without any isolated nodes or groups of nodes such that the longest shortest path is the diameter of a network.

Network based SIR models allow for the stochastic modeling of the spread of diseases through social networks. In these networks, individuals are represented as nodes and connected with edges when they are in contact, allowing the transmission of diseases. [1] Each node in the graph has a status, representing if the node is susceptible, infected or recovered. The spread of the disease is then modeled in discrete time steps. In each step, a susceptible node with an infected neighbour has chance β of being infected, and an infected node has chance γ of recovering. [3] Demography is commonly ignored in network based models.

For this paper, we compared a real world network [4] to four randomly generated graph types with similar properties. The first network used is the Erdős-Rényi network, which is also known as the binomial network. [5] For this simple network, a node has a chance to generate an edge with another node with probability p. The second network used is the Random Regular Graph. [6] In this type of network, all nodes have identical degrees, as opposed to the Erdős-Rényi network. The third network used is the Watts-Strogatz network. [7] This is a network known as a Small-Worlds network. In this type of network, nodes are normally connected to other nodes in their close neighbourhood. Some nodes are then also given edges to random nodes outside of their neighbourhood. The last network used is the Barabási-Albert network. [8] This is a network from the Scale-Free category. When this type of network is generated randomly, nodes are more likely to generate connections to nodes with a higher degree, which promotes the creation of hubs, or nodes with many connections.

4 Methods

4.1 Gillespie's Event-Driven Algorithm for SIR Infection

To represent the events present in our SIR model, we created classes for each possible event which could occur, which all inherited from a base Event class. These classes all contained the change in the X, Y and Z parts of our population and the rate of their occurrence. A different

class representing the model could then generate these classes for each iteration and choose one based on their rates.

To show how changes to the model parameters change the behavior of the stochastic models, we created two series of data. For the first series, the value of β was varied from 0.05 to 0.95, with a fixed value of $\gamma = 0.05$. For the second series, the value of $\beta = 0.25$ was kept constant, and γ was varied between 0.045 and 0.255. For each combination of parameters, 50 simulations were run. The mean and standard deviation of the data points of these 50 simulations was then run, which could then be plotted and compared to an equivalent deterministic SIR model. The covariance between X and Y was also calculated for each simulation, with the values for each model being averaged. A similar process was used to create three dimensional plots of the mean, standard deviation and covariance. Here, β was varied from 0.05 to 1.0 and γ was varied from 0.1 to 0.6. Data about extinction times was also collected for these simulations. The proportion of simulations which went extinct and the average life time of the extinct simulations was recorded and plotted in a three dimensional graph.

To show and analyze the effect of stochastic resonance in the stochastic model, we first used a Fourier Transformation to analyze the frequencies of the Y class in our model. This was done on two models, which both had a β of 0.5 and γ values of 0.10 and 0.15. To show how the parameters of the model can affect the strength of the stochastic resonance, we then plotted how the normalized standard deviation of the models varied in relation to the population size and the value of R_0. The data for these points was obtained by running five simulations for each combination of the parameters N and γ, calculating the standard deviation of the points, dividing this by N and taking the average of the five simulations. The error on the points is the standard deviation of the values obtained from the five simulations. The data for the points was taken from $t = 300$ to $t = 1000$ to ensure the data range only contains the endemic state of the model. Additionally, the data from the models was only taken from simulations which did not experience stochastic extinction.

4.2 Network Models

The network creation process begins with Networkx [9], a Python library that generates networks such as Barabasi-Albert, Erdos-Reyni, and Watts Strogatz using library methods to define network features based on node and edge parameters. This library may also be used to create networks from edge matrices such that the Sociopatterns transmission network data may be used to create a network of nodes once read. Generated network features may then be analyzed using Networkx [9] methods to determine clustering, diameter, degree, node number, edge number, degree distribution, and so on.

4.2.1 SIR Infections

NDLib [3, 10] is the Python library that is used to construct an infection simulation using a range of model types such as SIR, SIS, SEIR, and so on. These constructed network models may then be configured using a model configuration sub library such that SIR parameters may be defined and initial states may be established to simulate infection by iterating over time. These iterations contain the node states that have changed such that the infection model can be updated based on the delta change of each iteration, although in this case, the code stores all states, including unchanged states, in each iteration to allow for ease of access for network analysis and vaccinations at each time step.

4.2.2 Vaccination Strategies

Networks that are infected may have vaccine interventions applied using a number of methods each relying on removing individual nodes from the Susceptible population at each time step given a limited number of vaccinations per time step. An initial null vaccination strategy sees a random subset of nodes vaccinated at each time step without testing the node state such that infected or recovered nodes are unaffected by vaccination.

A "Blind" Vaccination strategy sees the application of vaccinations to the nodes of the highest degree within the network, but retains the lack of testing from the null strategy. As such,

this vaccination strategy will also run the risk of vaccinating infected or recovered nodes without effect, but will be able to limit extreme infection spread from high degree nodes if they are not already infected.

A Dynamic Vaccination strategy introduced a complex set of vaccination steps that tests and vaccinates depending on priority, vaccination limits per time step, testing limits per time step, and overall test maximum. This strategy will default to the "Blind" vaccination strategy if it has reached the overall maximum number of tests so that vaccinations will continue even after testing stops. If the maximum number of vaccinations is reached, the time step will iterate to avoid wasting tests without vaccinations. There is also an additional parameter for the limit of tests per-time step, which will be equal to the maximum number of tests for these vaccination simulations

Both testing and vaccination will always prioritize the highest degree neighbors of infected nodes that were discovered in previous tests such that any neighbor to an infected node will be vaccinated if it is susceptible and infected nodes will have their own neighbors added to the priority stack.

The secondary priority stack are the highest degree nodes of the full network wherein tests are done if no neighbors of infected nodes have been found yet. Any infected nodes found will yet again have their neighbors added to the priority stack, while susceptible nodes will be added a new stack that will vaccinated if-and-only-if there are vaccines available once testing is completed for a single time step. This new stack ensures that all vaccines are used even if there are no neighbors of infected nodes and will retested again in the following time step if un-vaccinated.

Testing also includes a testing accuracy parameter which determines the probability of a False Negative occurring such that an infected node is mistakenly determined to be susceptible. In this case, high priority neighbors of infected nodes may be vaccinated with no effect, while both low and high priority infected nodes will not have their neighbors added to the high priority testing stack.

Refer to A for Python code explanation for the three vaccination strategies tested.

5 Results

5.1 Gillespie's Direct Algorithm SIR

The implementation of Gillespie's direct algorithm creates a wide range of noise patterns across many different simulations of the stochastic model, and by selecting a range of parameters much like one would do with a deterministic SIR model, it is possible to observe how stochasticity influences simulation results.

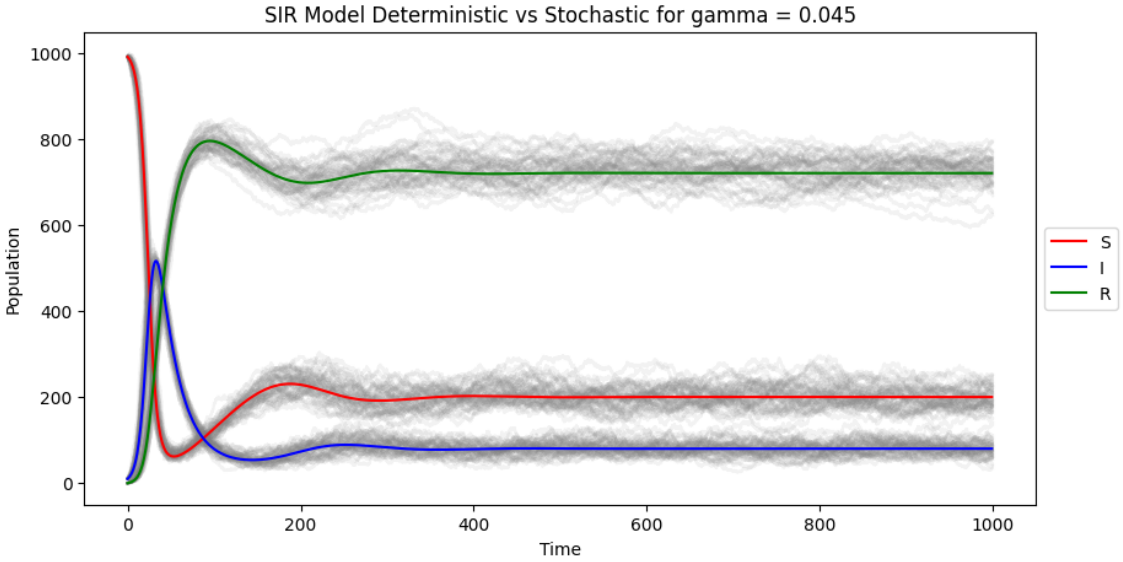


Figure 1: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.045$, and $\beta = 0.25$

Assuming a $\mu = 0.005$ for all SIR models going forwards, it is possible to observe in Figure 1 a stochastic set of simulations with a similar behavior to that of its deterministic parameter counterpart with a $\beta = 0.25$ and a $\gamma = 0.045$ ($R_0 = 5$). The dynamics of the Gillespie's algorithm follow closely with that of the deterministic model, although no stochastic simulation is ever the same, expressing the constant variability between simulations that is a feature of stochasticity. Additionally, this variability gives rise to the presence of increased transients away from the deterministic model in the form of outlier results for SIR data, a feature that is most observable in the Figure 1 susceptible population over the time step 200-400 – this increased transient behavior tends to return towards the deterministic model as it acts as a deterministic attractor of the stochastic simulations [1].

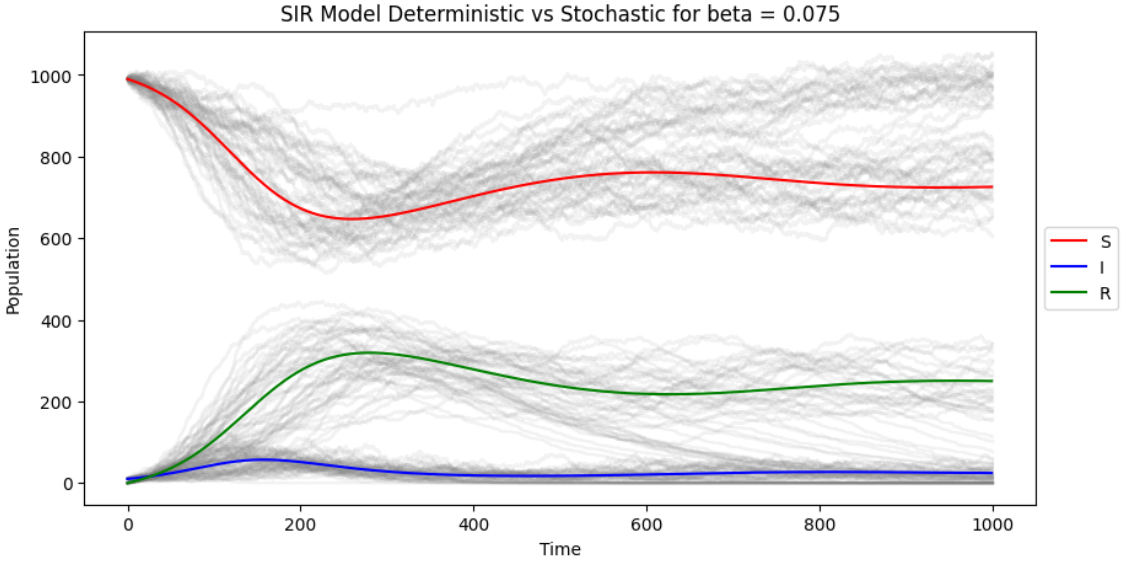


Figure 2: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.045$, and $\beta = 0.075$

By adjusting the parameters such that $\beta = 0.075$ and $\gamma = 0.05$, the Reproductive Rate of the simulations becomes $R_0 = 1.5$, which causes the overall infections to be reduced and the noise level of the stochastic simulations to increase in Figure 2. The variability of these simulations increases substantially while the tendency towards the deterministic attractor is reduced, although still present. The low R_0 also expresses another feature of stochastic models in the form of extinctions which occur due to the random variability of the stochastic simulations that increase the possibility of infected populations to reduce to zero from downward noise or transients [1]. These extinctions are observable in the Susceptible populations that expand away from the deterministic attractor as the force of infection no longer retains the pull of the model.

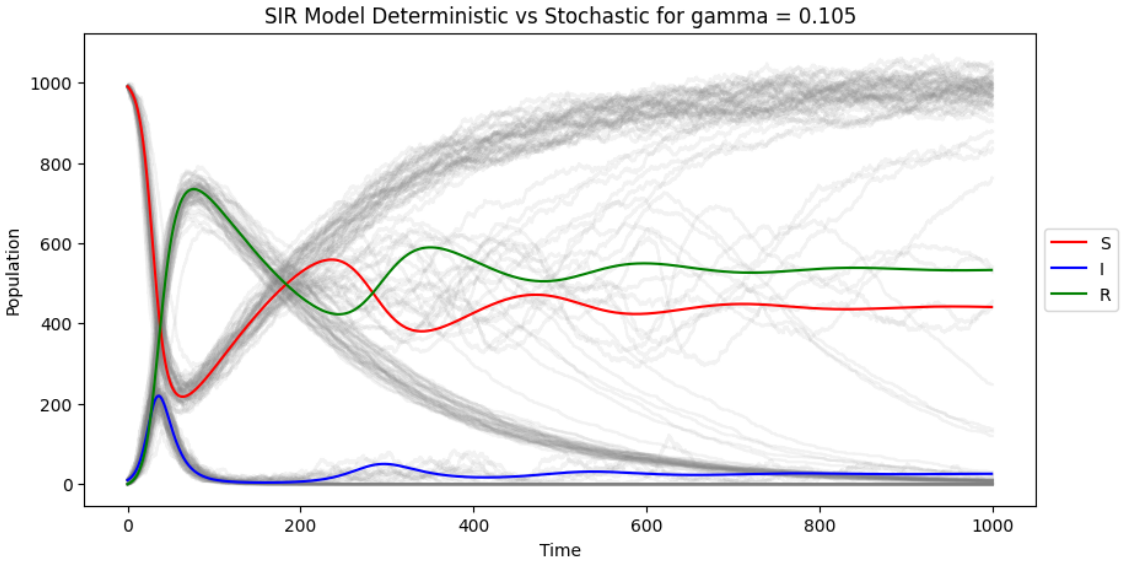


Figure 3: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.105$, and $\beta = 0.25$

By adjusting parameters to increase oscillations in the deterministic SIR model, it becomes possible to further exacerbate the variability and extinction behavior of the stochastic simulations such that Figure 3 sees a complete extinction of most infections in the population. With $\beta = 0.25$ and a $\gamma = 0.105$ such that $R_0 \simeq 2.4$, it is possible to see how oscillations in the de-

terministic model and its attractor pressure causes the "momentum" of attraction to push the infected population to extinction such that the susceptible and recovered population are free to continue a co-variable system based on the birth/lifespan parameter pressures. This behavior around the oscillations of the deterministic model are features of stochasticity that develop from the existence of transients – spikes in simulation states that occur randomly due to stochastic processes – but which are also exacerbated by the presence of oscillations in the deterministic state [1].

Additional SIR figures for ranges of β and γ are available in the Appendix B.

5.2 Stochastic Resonance Analysis

To explore the effects of stochastic resonance, we first applied a Fourier transform on the Y fraction of two stochastic SIR models. The first model had the parameters $\beta = 0.5$ and $\gamma = 0.1$. The Fourier transformation and the found peaks are shown in Figure 4 Four peaks have been detected and marked in this plot. The first peak has a frequency of $f = 0.000333$, or a period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.014333$, or a period of $T = 46$. The forth peak has a frequency of $f = 0.02133$, or a period of $T = 46$.

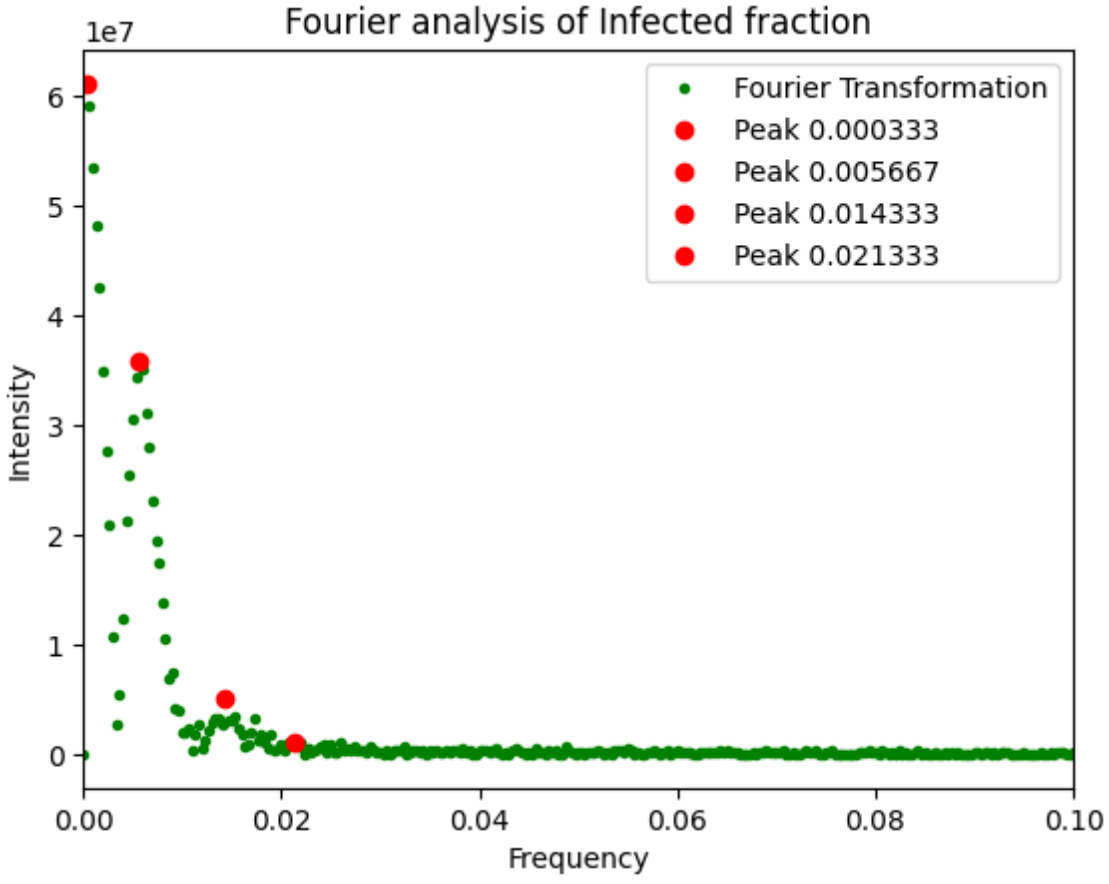


Figure 4: A Fourier transform of a stochastic SIR simulation with $\beta = 0.5$ and $\gamma = 0.1$. Four peaks have been detected and marked in this plot. The first peak has a frequency of $f = 0.000333$, or a period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.014333$, or a period of $T = 46$. The forth peak has a frequency of $f = 0.02133$, or a period of $T = 46$.

The second model had the parameters $\beta = 0.5$ and $\gamma = 0.1$ and is shown in Figure 5. Three peaks have been marked for this model. The first peak has a frequency of $f = 0.000333$, or a

period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.012667$, or a period of $T = 69$.

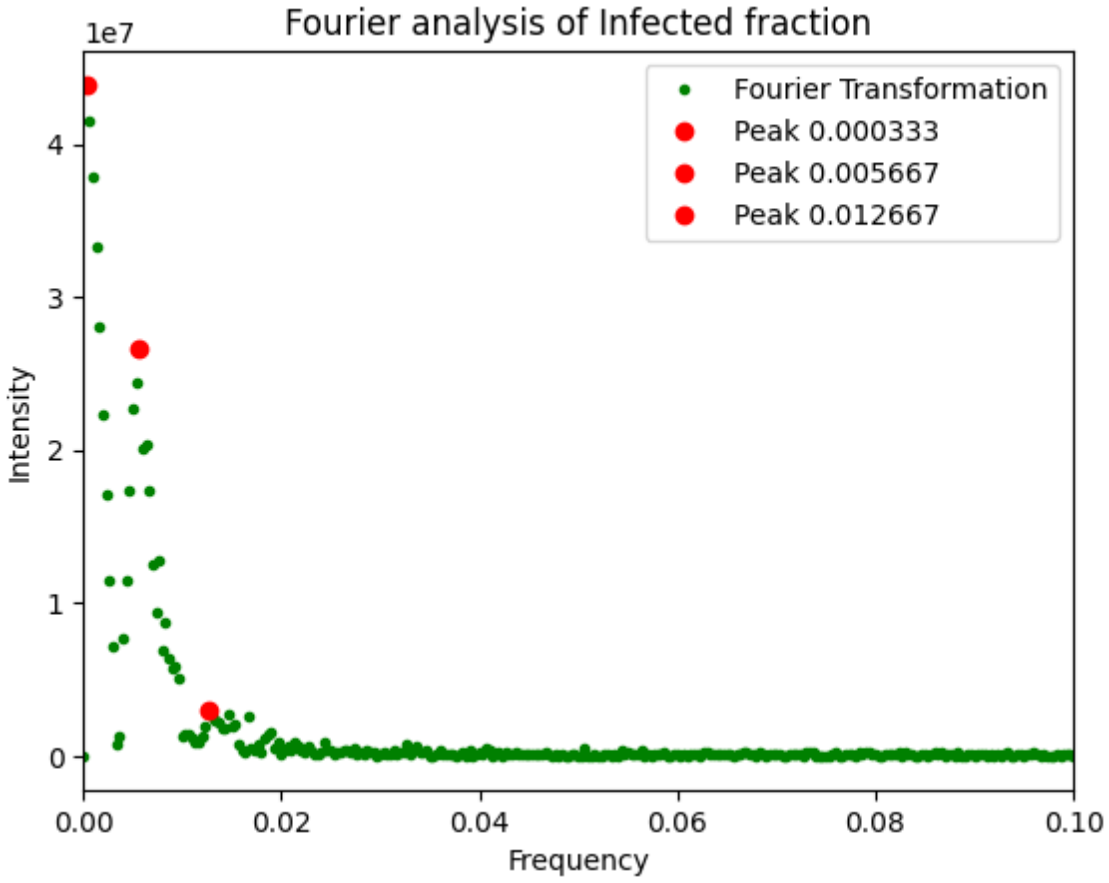


Figure 5: A Fourier transform of a stochastic SIR simulation with $\beta = 0.5$ and $\gamma = 0.15$. Three peaks have been detected and marked in this plot. The first peak has a frequency of $f = 0.000333$, or a period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.012667$, or a period of $T = 69$.

For both of these models, the highest peak corresponds to a period which is longer than the total runtime of the simulations, which makes it unlikely that this is the frequency of the stochastic resonance. The second peaks have a period of 176, which optically does appear to align with the length of the oscillations in the models. The other found peaks are low in amplitude and can be disregarded.

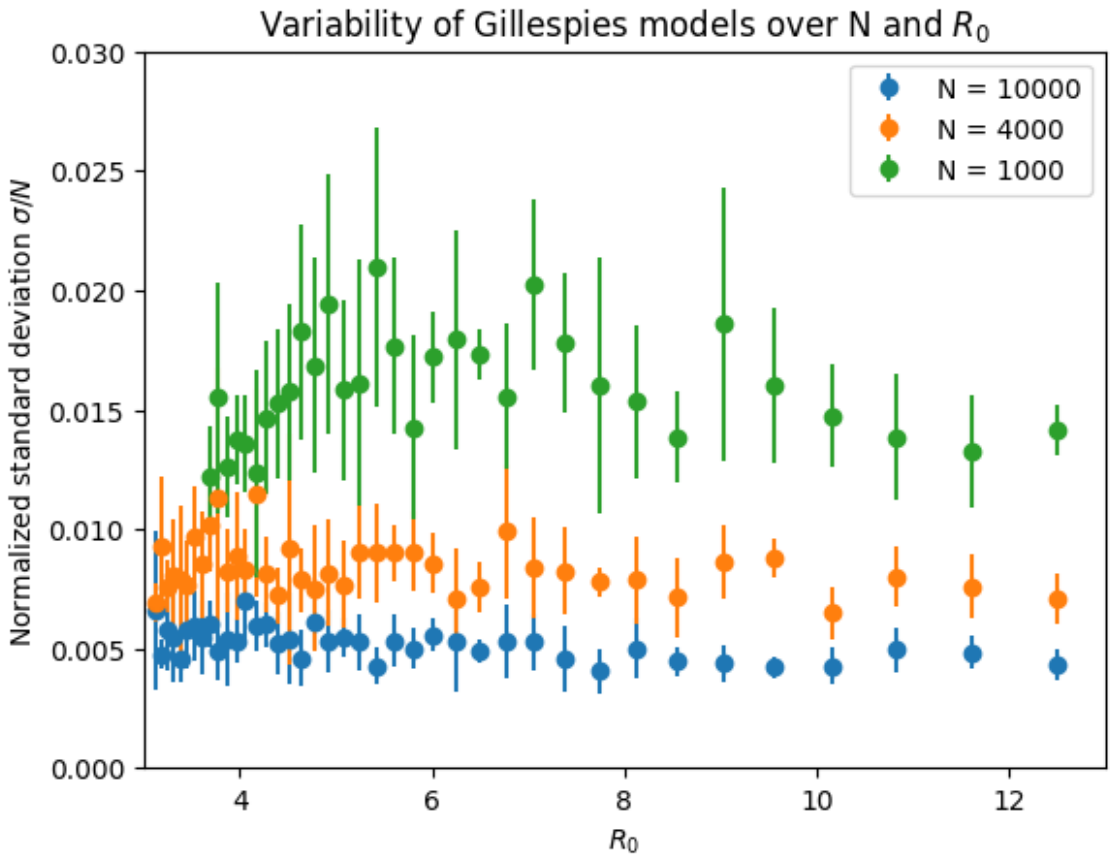


Figure 6: The normalized standard deviation of stochastic SIR models. The models were run with populations of 1000, 4000 and 10.000. The $\beta = 0.3$ and γ was varied between 0.04 and 0.16. For the $N = 1000$ models, the γ was only increased to a value of approximately 0.13, as simulations running at higher values went extinct to frequently, which caused a significant slowdown in the code. The plot shows that smaller populations have a higher normalized standard deviation, and are therefore more affected by stochastic effects. Additionally, the $N = 1000$ dataset also appears to show a peak around $R_0 = 5.5$. This might indicate that this value exhibits stochastic resonance.

Figure 26 shows how the standard deviation of stochastic SIR models varies for different population sizes and values for R_0. The models were run with populations of 1000, 4000 and 10.000. The $\beta = 0.3$ and γ was varied between 0.04 and 0.16. For the $N = 1000$ models, the γ was only increased to a value of approximately 0.13, as simulations running at higher values went extinct to frequently, which caused a significant slowdown in the code. Figure 104 shows that smaller populations have a higher normalized standard deviation, and are therefore more affected by stochastic effects. Additionally, the $N = 1000$ dataset also appears to show a peak around $R_0 = 5.5$. This might indicate that this value exhibits stochastic resonance.

5.3 Statistical Analysis of Gillespie's

Now given some $\gamma = 0.05$ and a range of β values, it's possible to determine the behaviors of these stochastic SIR models compared to the deterministic model at each β between 0.0 and 1.0 using statistical analysis of each Susceptible, Infected, and Recovered population. Statistics such as the Mean, Variance, Standard Deviation, and Covariance can express generally how these stochastic models act in comparison to the deterministic model over a set time-span given what is known about how certain parameter values cause increased oscillation and noise while others do not.

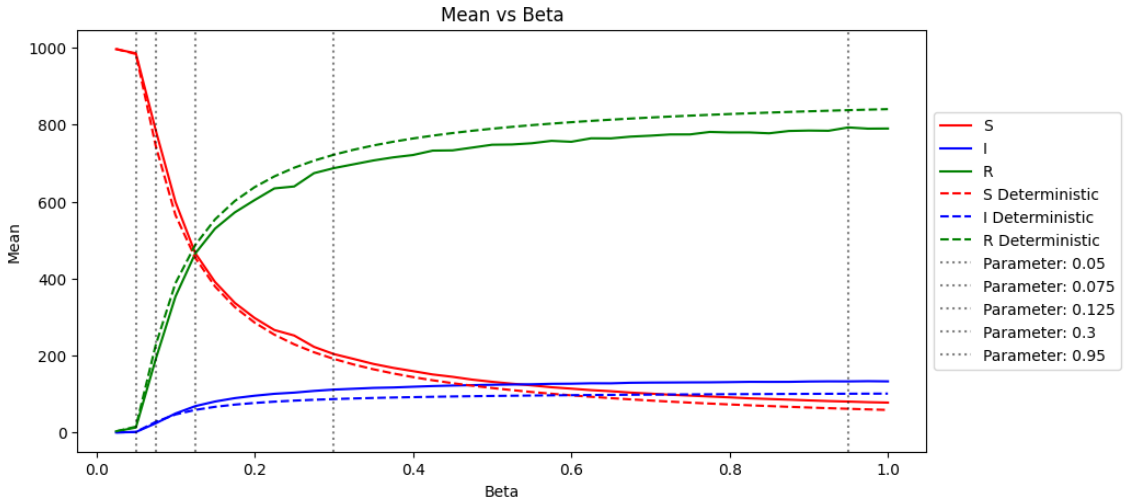


Figure 7: Mean of SIR for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = 0.045$, and $\beta = [0.0, 1.0]$

By calculating the overall mean of the each SIR population over 1000 time steps of 50 stochastic simulations and comparing the result to the mean of the deterministic SIR populations over the same time steps, it is possible to see how closely each population follows one another for the range of $\beta = [0.0, 1.0]$ in Figure 7. Given the specific $\gamma = 0.05$, it is apparent in the figure that the overall behavior of the stochastic and deterministic models don't vary from one-another substantially over the 1000 time steps, showing how the deterministic attractor effectively acts as an average for stochastic model dynamics.

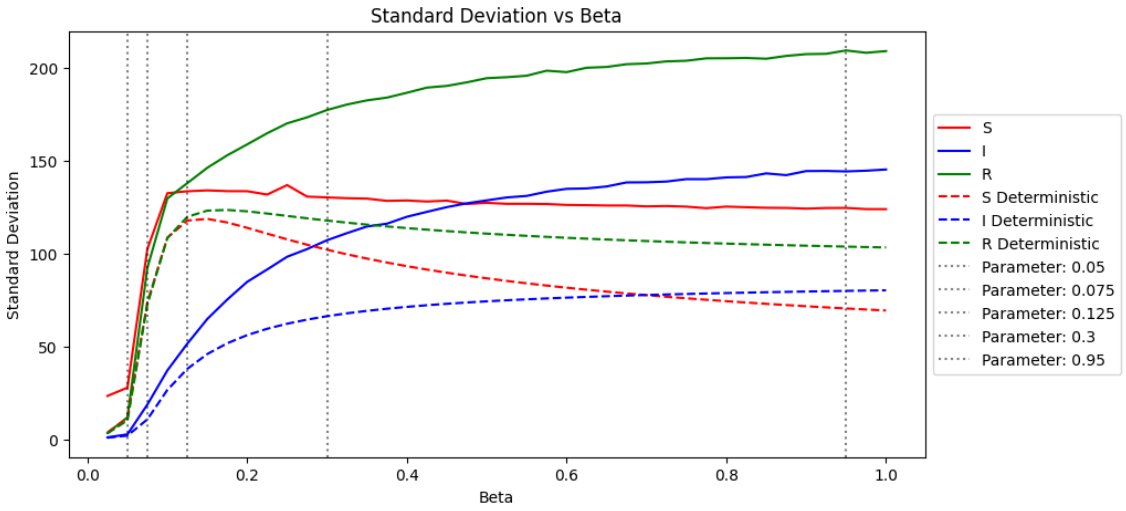


Figure 8: Standard Deviation of SIR for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = 0.045$, and $\beta = [0.0, 1.0]$

By calculating Standard Deviation and likewise Variance for the same γ and range of β, it is possible to determine the general variability of the stochastic simulations and deterministic model away from their Figure 7 Means, as depicted in Figure 8. This visualization represents the general oscillations of the models away from their means such that a highly variable or noisy model will express a higher Standard Deviation than one that is flat. As such, it is possible to see in Figure 8 a spike in the variance of both stochastic and deterministic models which match each other closely until some point between $\beta = 0.075$ and $\beta = 0.125$ where they diverge substantially.

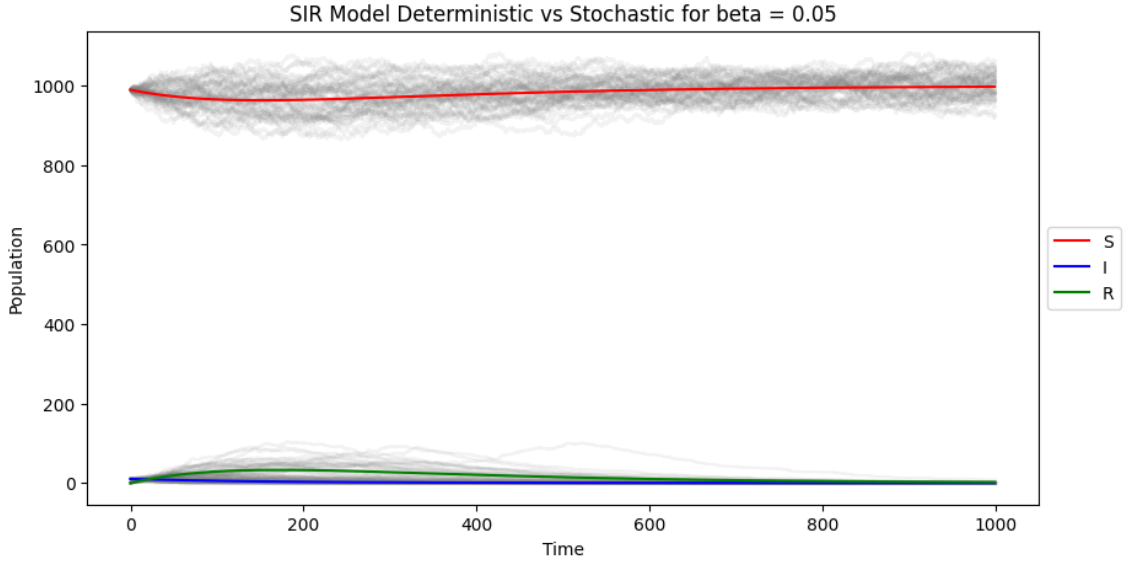


Figure 9: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.05$, and $\beta = 0.05$

The spike in the Standard Deviation of the model coincides with the rise in infections when $R_0 > 1.0$ as depicted in the Figure 9 where $\beta = 0.05$ such that $R_0 \simeq 0.9$ and the SIR populations remain nearly flat. The sudden divergence of the variance between $\beta = 0.075$ and $\beta = 0.125$ occurs for related reasons between the stochastic and deterministic models from underlying behaviors of the deterministic oscillations.

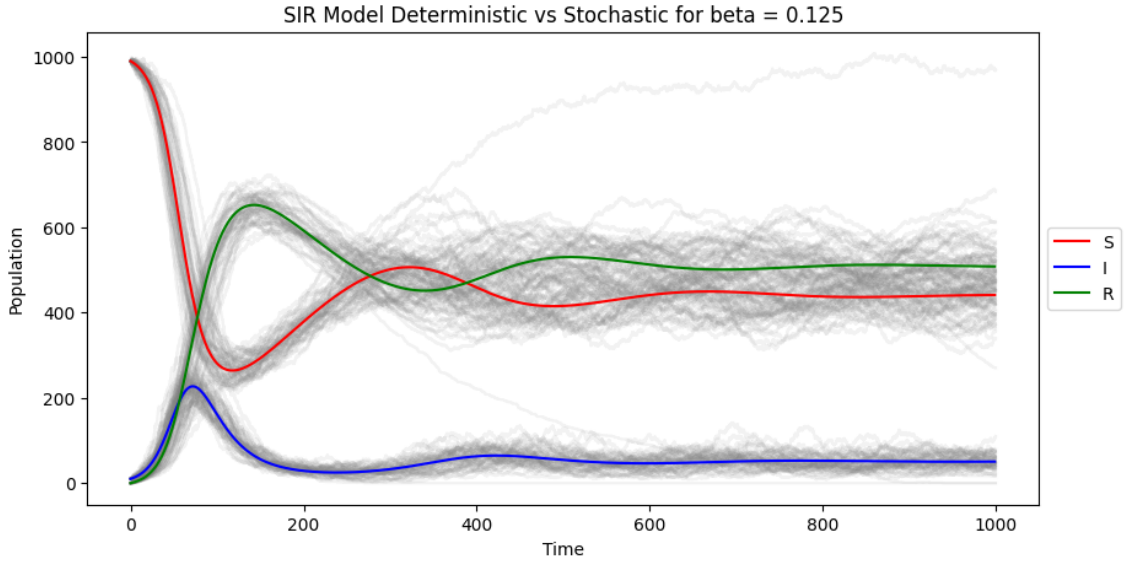


Figure 10: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.05$, and $\beta = 0.125$

The variance of the deterministic model peaks as oscillations become most pronounced in the populations at around $\beta = 0.125$ in Figure 10, but it's also apparent to see how these oscillations effect the stochastic simulations by creating harmonic resonance around the peaks that encourage noise, and thus variability.

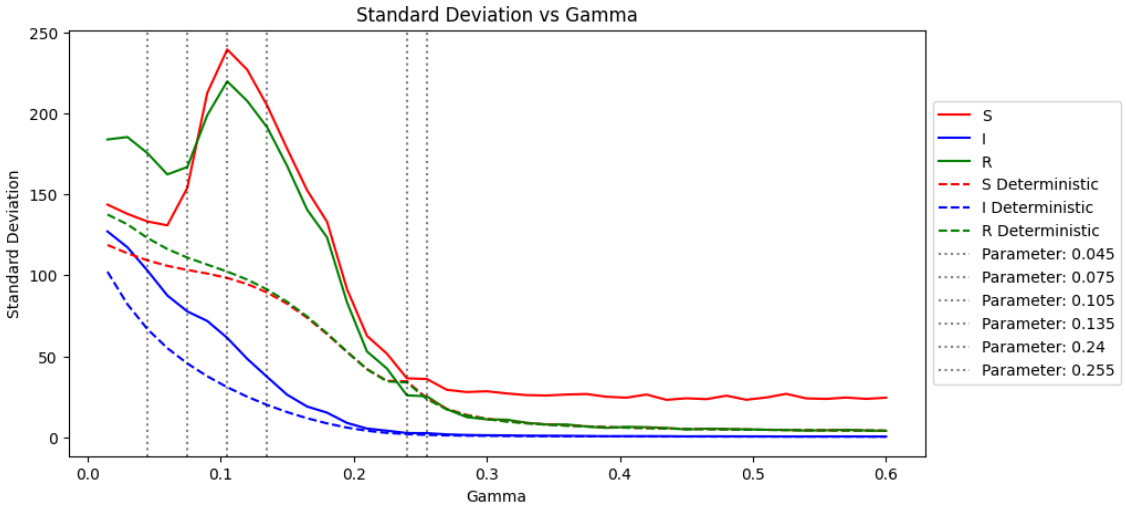


Figure 11: Standard Deviation of SIR for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = 0.25$

Interestingly, this harmonic resonance behavior is *more* pronounced when γ is changed with $\beta = 0.25$ and a $\gamma = [0.05, 0.6]$ as seen in Figure 11. This set of parameters expresses a larger Standard Deviation for stochastic simulations over many values of γ with equivalent R_0 within the range of β such that both Figures 10 and 3 have the same $R_0 \simeq 2.27$ with $\mu = 0.005$, but have completely different stochastic behaviors.

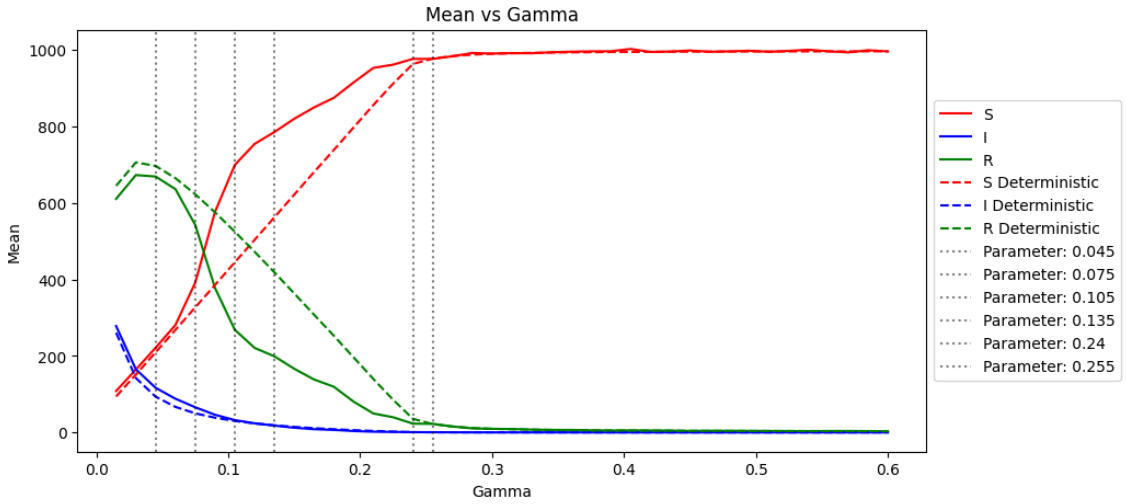


Figure 12: Mean of SIR for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = 0.25$

The force of harmonic resonance is apparently much higher for Figure 3 than Figure 10, causing increased noise which becomes apparent in the peak of the Standard Deviation values of Figure 11 between $\gamma \simeq 0.5$ and $\gamma \simeq 0.2$. This change in behavior effects the dynamics of the stochastic simulations enough to shift the overall Means of the SIR populations over the range of γ for $\beta = 0.25$ in Figure 12 such that the spike in standard deviation is present in the susceptible and recovered populations of the stochastic simulations.

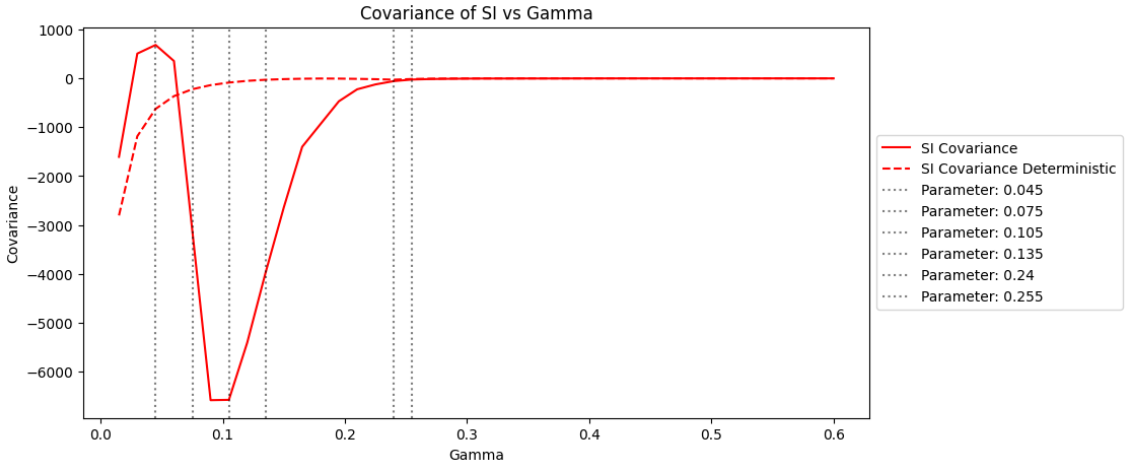


Figure 13: Covariance of SI for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = 0.25$

Likewise, it effects the Covariance of the simulations in Figure 13 where the natural negative covariance that exists in the deterministic model is not reflected in the stochastic simulations where the standard deviation spikes between $\gamma \simeq 0.5$ and $\gamma \simeq 0.2$. This is understandable when observing the Figure 3 as the susceptible populations for a number of stochastic simulations rises to maximum over the time span while the infected remain at or close to 0.

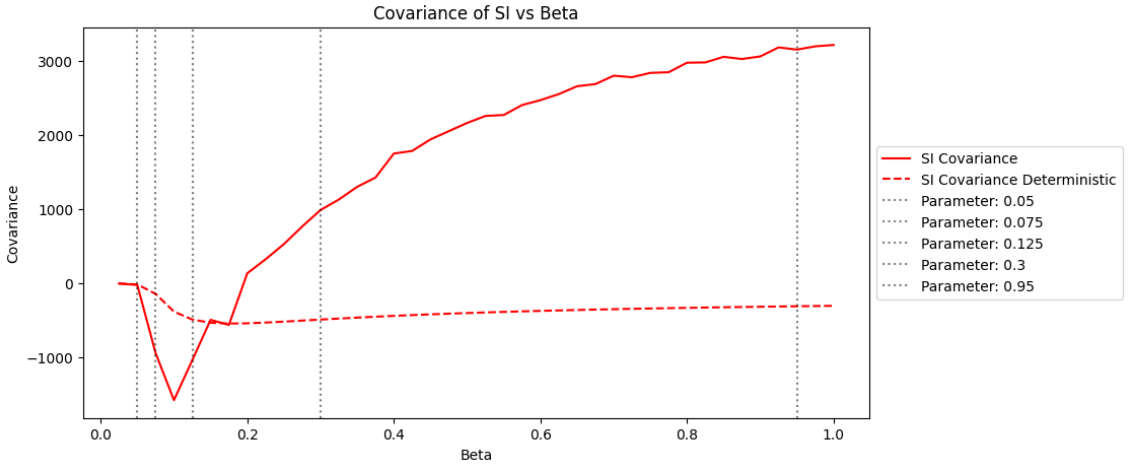


Figure 14: Covariance of SI for 1000 time steps and 50 stochastic runs vs 1 deterministic run $\mu = 0.005$, $\gamma = 0.05$, and $\beta = [0.0, 1.0]$

Given what is known about deterministic SIR models and the fact that changes to β with a low $\gamma = 0.05$ see a tighter coupling of the stochastic simulations to the deterministic attractor, it might be easy to assume that the negative covariance behavior of these simulations would continue to be present in both. However, Figure 14 shows how this assumption fails for the given parameters as the covariance instead becomes substantially positive as β becomes larger.

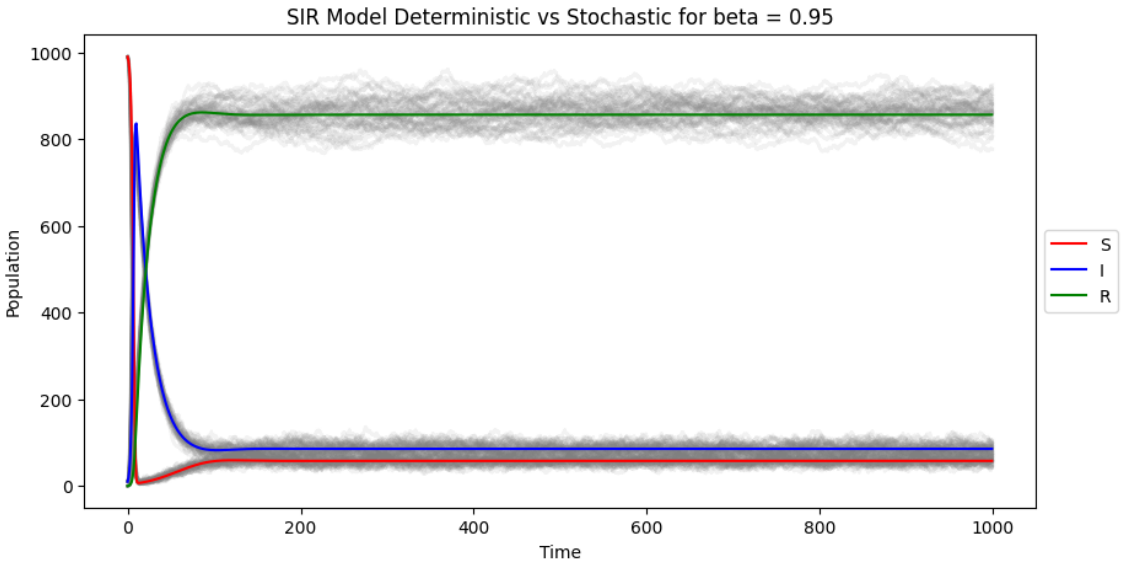


Figure 15: SIR Model w/ Demography $\mu = 0.005$, $\gamma = 0.05$, and $\beta = 0.95$

The reason behind this spike in positive covariance may be explained through Figure 15 where-in both the infected and the susceptible populations are close to 0 and noise of the stochastic simulations seems minimal. The deterministic attractor has reached an equilibrium state, but the stochastic simulations remain noisy, so it's possible that extinction risk has become a pressure on the simulations such that the rate of stochastic noise that would cause either the susceptible or infected population to drop to zero is less likely to occur with such low values.

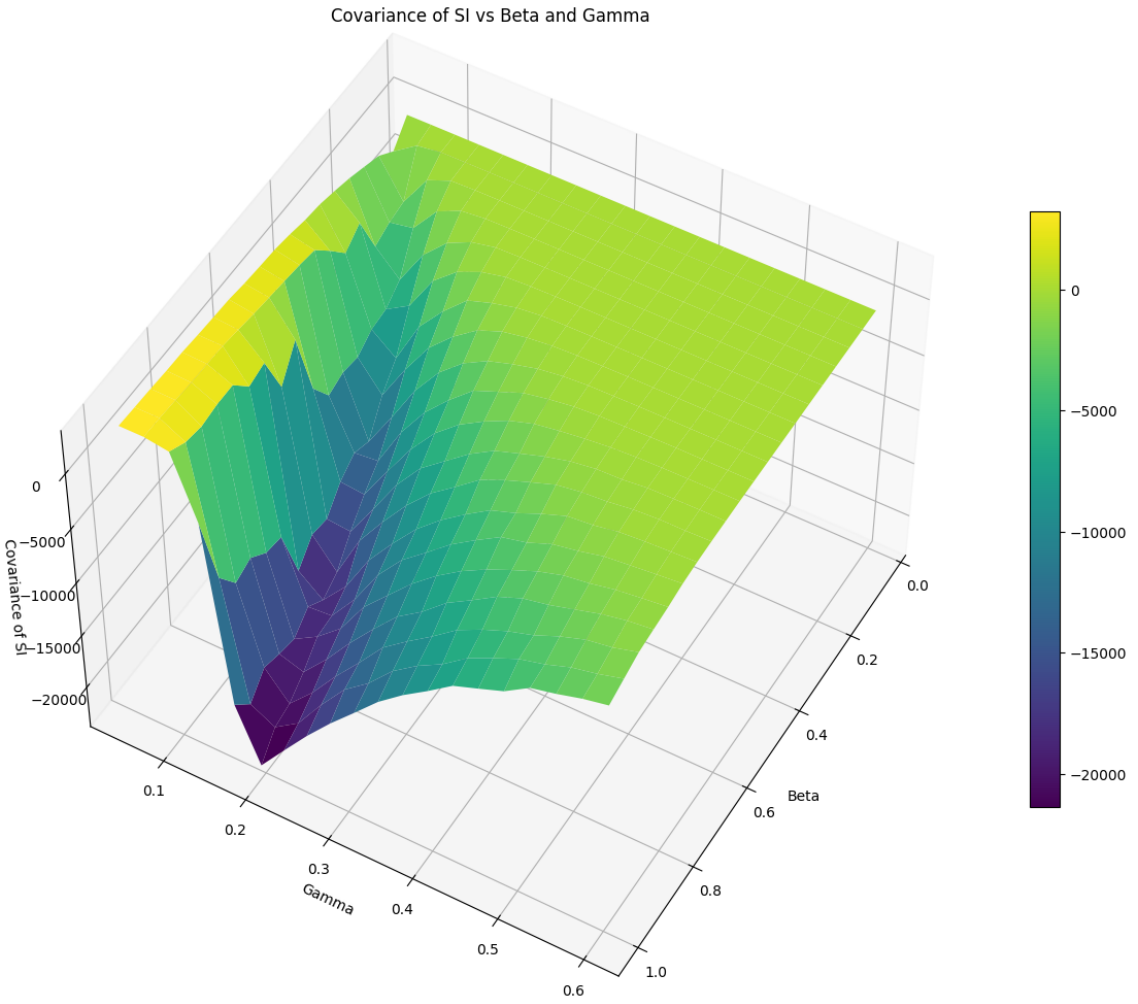


Figure 16: Covariance of SI for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = [0.0, 1.0]$

With selection of a single γ or β over a range of either, it's difficult to determine if these statistical dynamics are unique to these parameters or are an overall trend between both parameters, so by plotting the ranges of both in Figure 16, it becomes more apparent what is occurring with the covariance between S and I. What becomes immediately apparent is that the range of γ where the spike of negative covariance occurs in Figure 13 is actually a rift along a range of β representing the R_0 values that cause extreme harmonic resonance like that seen in Figure 3.

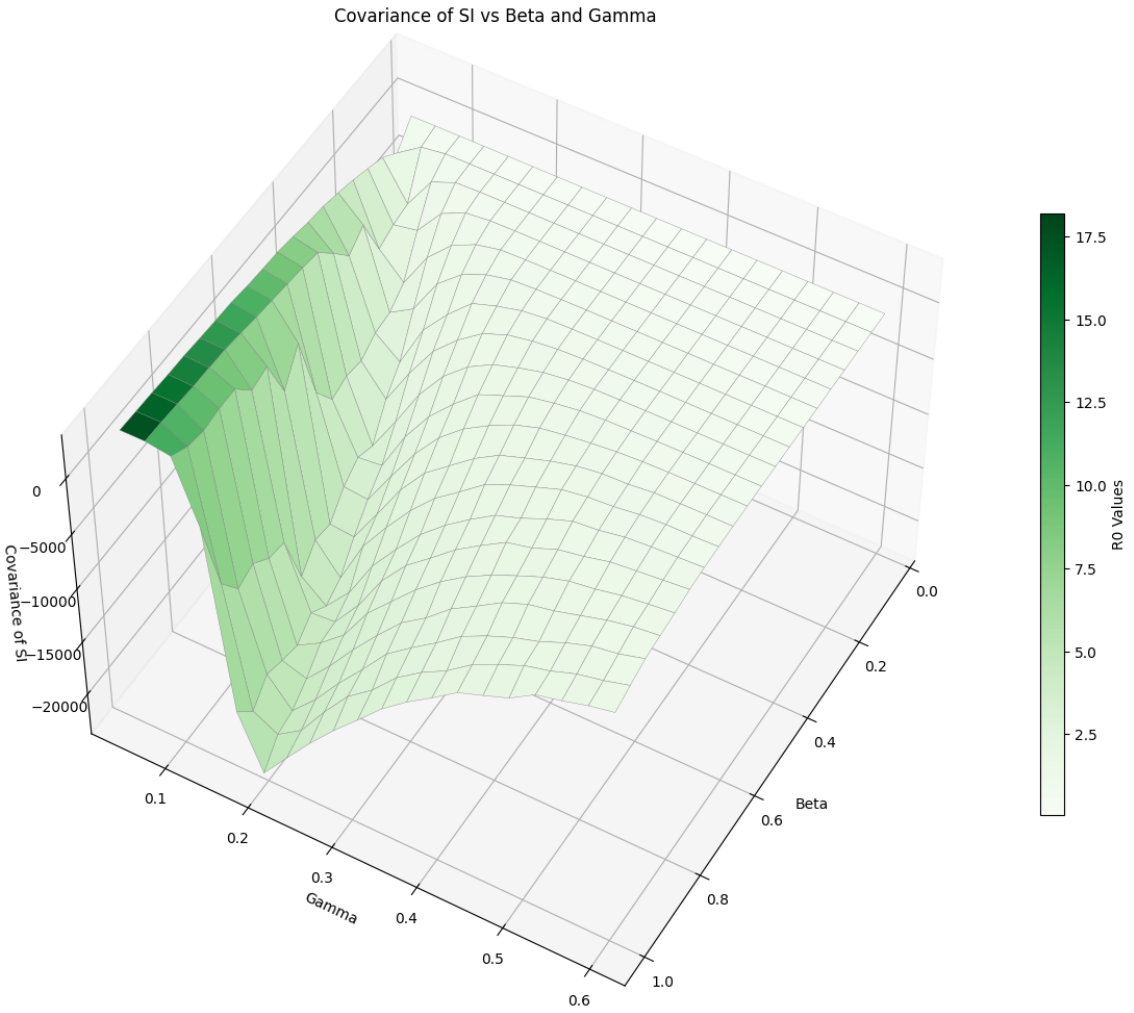


Figure 17: Covariance of SI for 1000 time steps and 50 stochastic runs with R_0 $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = [0.0, 1.0]$

Figure 17 shows how this rift relates to the hyperbolic increase of R_0 as γ approaches 0, which also coincides with the sudden increase of the positive covariance across values of β, reflecting the findings in Figure 14 for $\gamma = 0.05$. Overall, the covariance rift and subsequent spike of positive covariance is apparently related to the R_0 value as γ approaches zero in the stochastic simulations, a feature that is also apparent when observing the relationship between β and γ in regards to standard deviation and variance in Figure 18.

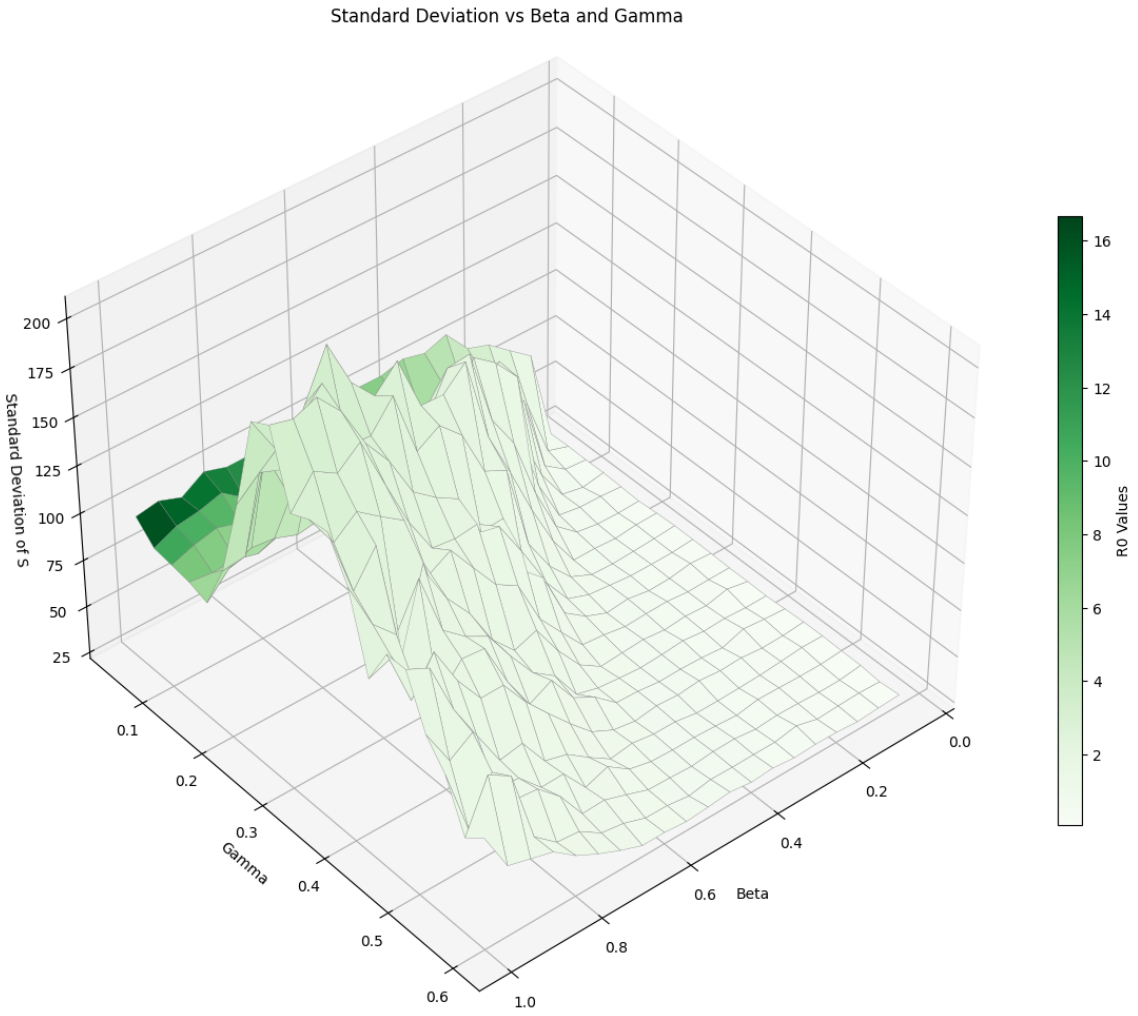


Figure 18: Standard Deviation of Susceptible Population for 1000 time steps and 50 stochastic runs with R_0 $\mu = 0.005$, $\gamma = [0.05, 0.6]$, and $\beta = [0.0, 1.0]$

The Standard Deviation across ranges of β and γ for the infected population express the peak observed in Figure 11 caused by the spike of negative covariance in Figure 17 and directly correlating with the values of β and γ along the most negative covariance values.

Additional Statistical Figures for the β and γ range are available in the Appendix B.

5.4 Extinction

With the possibility that the spike in positive covariance for low values of γ and high values of β seen in Figure 14 ($\gamma = 0.05, \beta = [0.0, 1.0]$) may be the relationship between R_0 and extinction, it is useful to determine when and how often these extinctions are occurring for different parameter values. One general feature of stochastic models is their ability and tendency to go extinct, a dynamic caused by the noise of stochastic simulation and one that effectively means that all stochastic infections will go extinct given enough time [1].

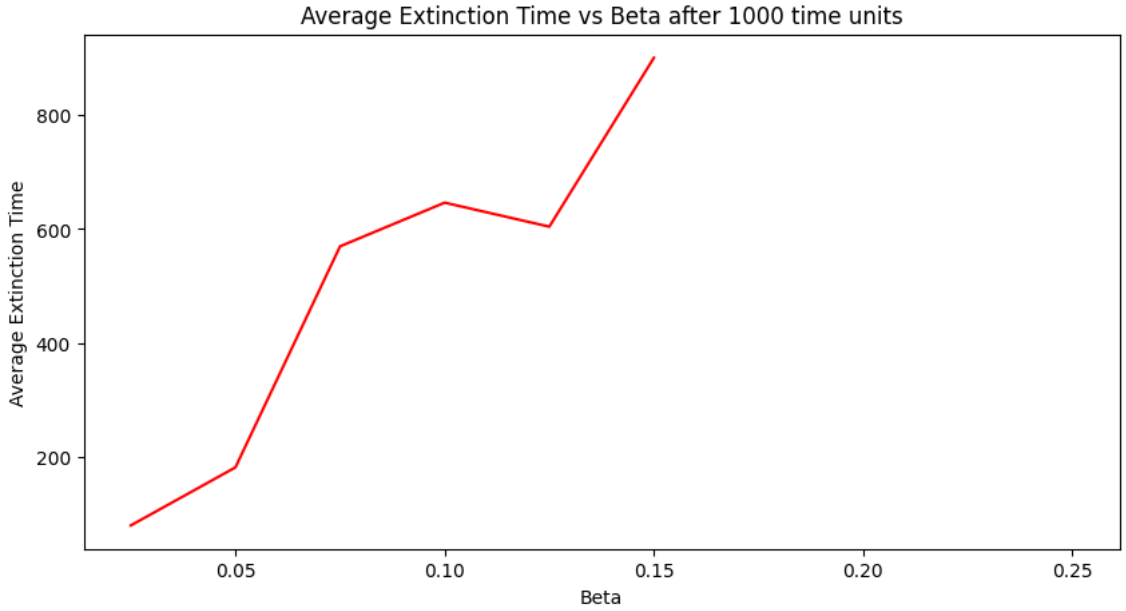


Figure 19: Infection Extinction Time for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = 0.05$, and $\beta = [0.0, 1.0]$

The issue with this becomes apparent in Figure 19 where-in no further data exists for the average extinction time of the model for $\gamma = 0.05$ and $beta > 0.15$ due to likelihood of extinction being far less frequent as R_0 increases. This means that even with a time span of 1000 iterations, data on extinction times of higher R_0 values requires considerably more time to simulate – it is also possibly why the covariance of the stochastic simulations becomes positive as γ becomes extremely low.

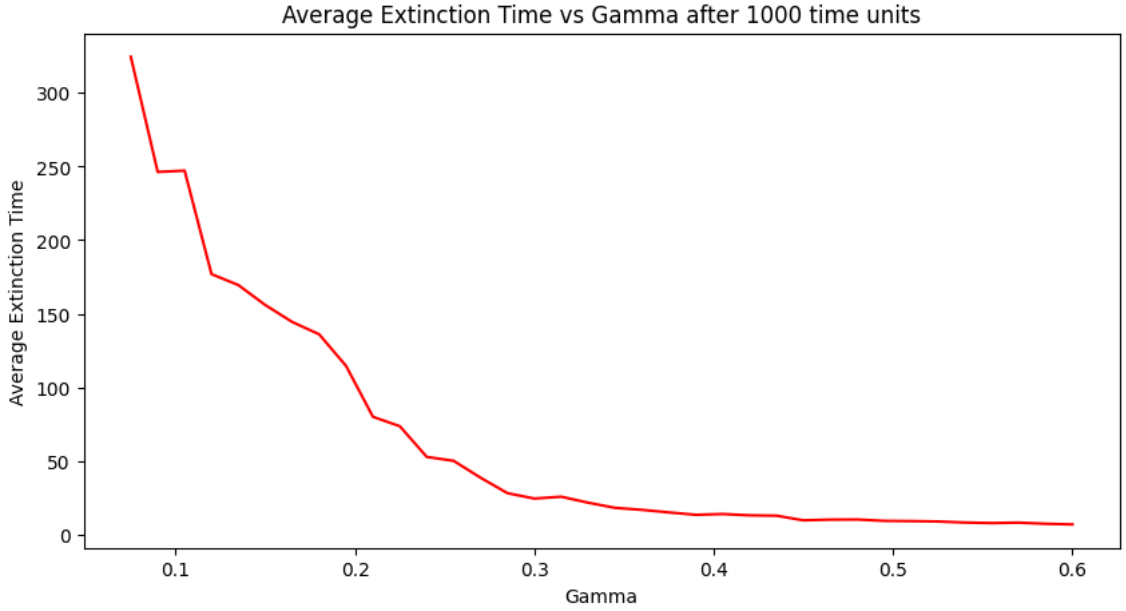


Figure 20: Infection Extinction Time for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = [0.1, 0.6]$, and $\beta = 0.25$

One possible solution to this is to select higher γ values or test average extinction times over a range of γ that keeps R_0 within a reasonable range such as in Figure 20 where $\beta = 0.25$ and

$\gamma = [0.1, 0.6]$. This shows how the average extinction time decreases as γ increases and R_0 subsequently decreases in a way that is more effective than Figure 19, but lacks values of γ below 0.1.

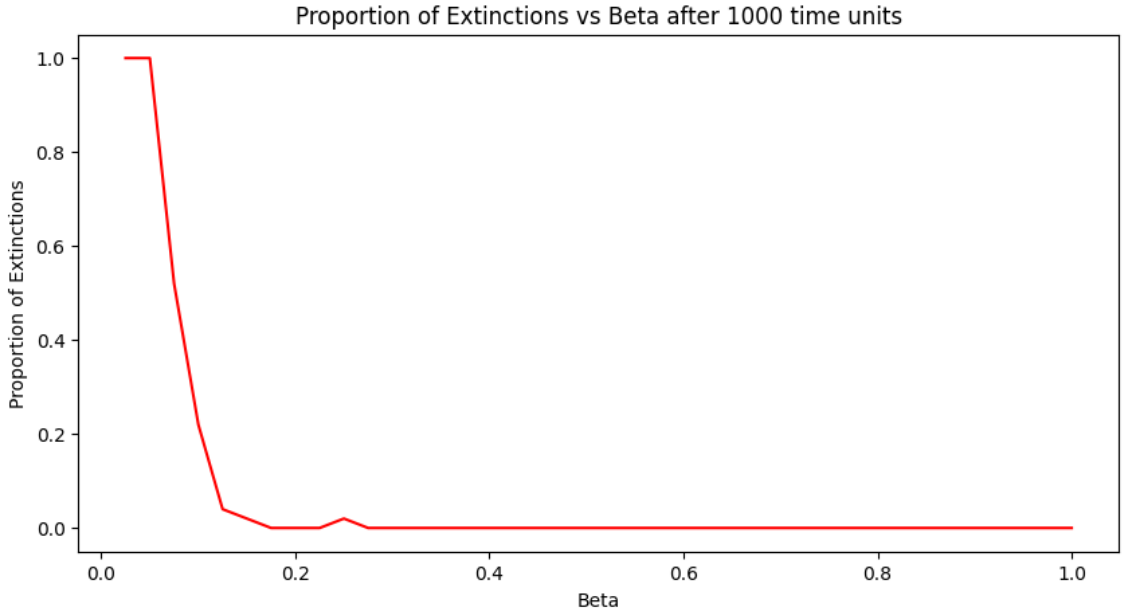


Figure 21: Infection Extinction Proportion for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = 0.05$, and $\beta = [0.0, 1.0]$

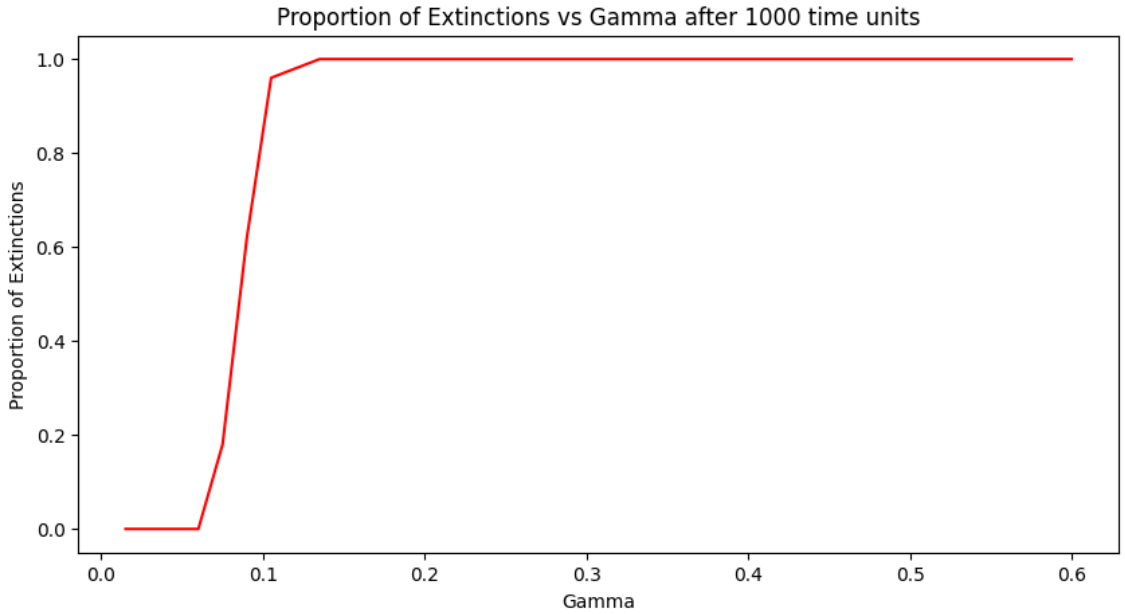


Figure 22: Infection Extinction Proportion for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = [0.1, 0.6]$, and $\beta = 0.25$

Extinctions may also be determined by finding the proportion of simulations that go extinct over a given time period in Figures 21 and 22. This gives data for the entire range of values, but is again reliant on the time span being simulated, with the proportion of extinction of lower R_0 values increasing with a larger time span, but also allowing for more data on higher R_0 values.

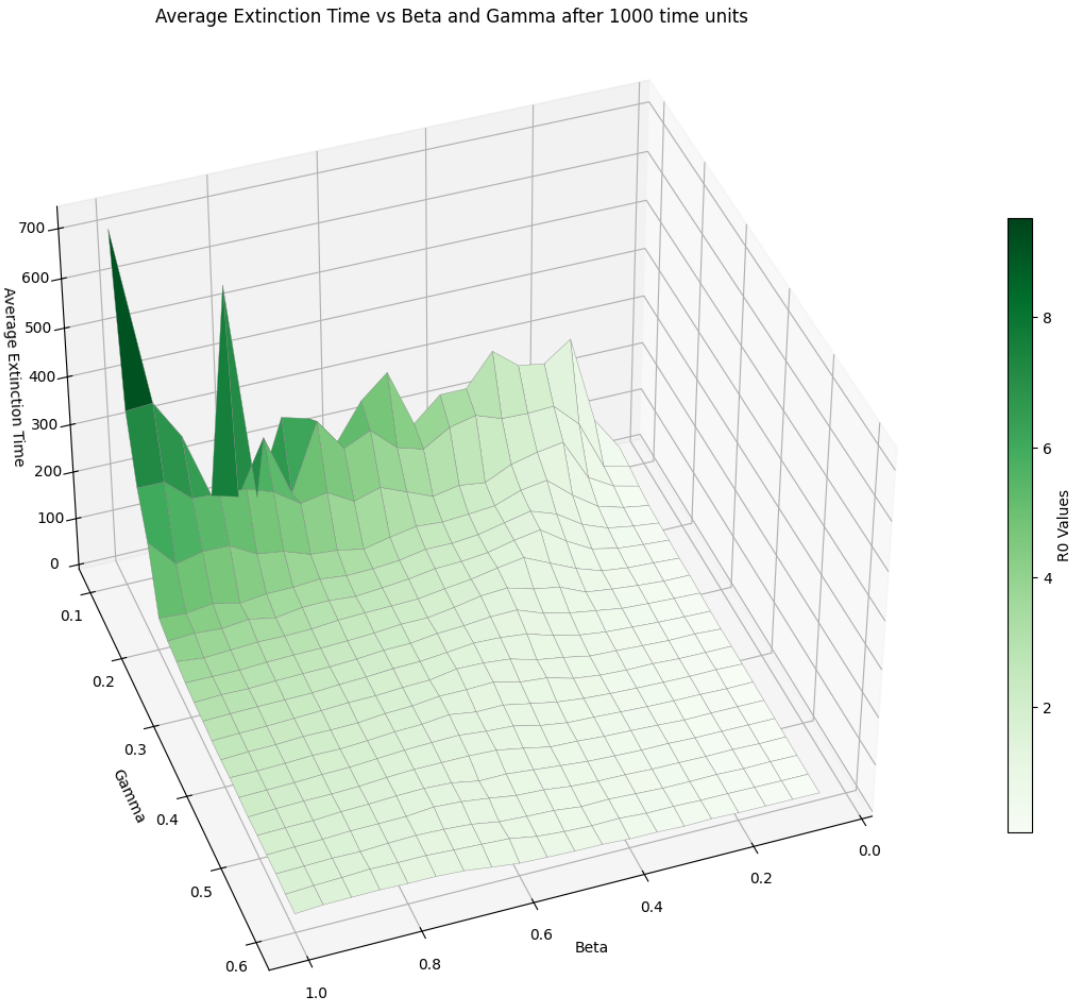


Figure 23: Infection Extinction Time for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = [0.1, 0.6]$, and $\beta = [0.0, 1.0]$

Plotting Extinction time for both a range of β and a range of γ in Figure 23 again gives slightly misleading understanding of average extinction times as the extinctions seem to peak at low γ, but in reality are cutting-off as there are no further simulations that go extinct for extremely low R_0 values again. It's also likely that the extinction times are low outliers when R_0 is high given the fact the the proportion of extinction is much lower at equivalent values of R_0 as per Figure 24, causing the values that *are* recorded at these values to be rare transient extinctions

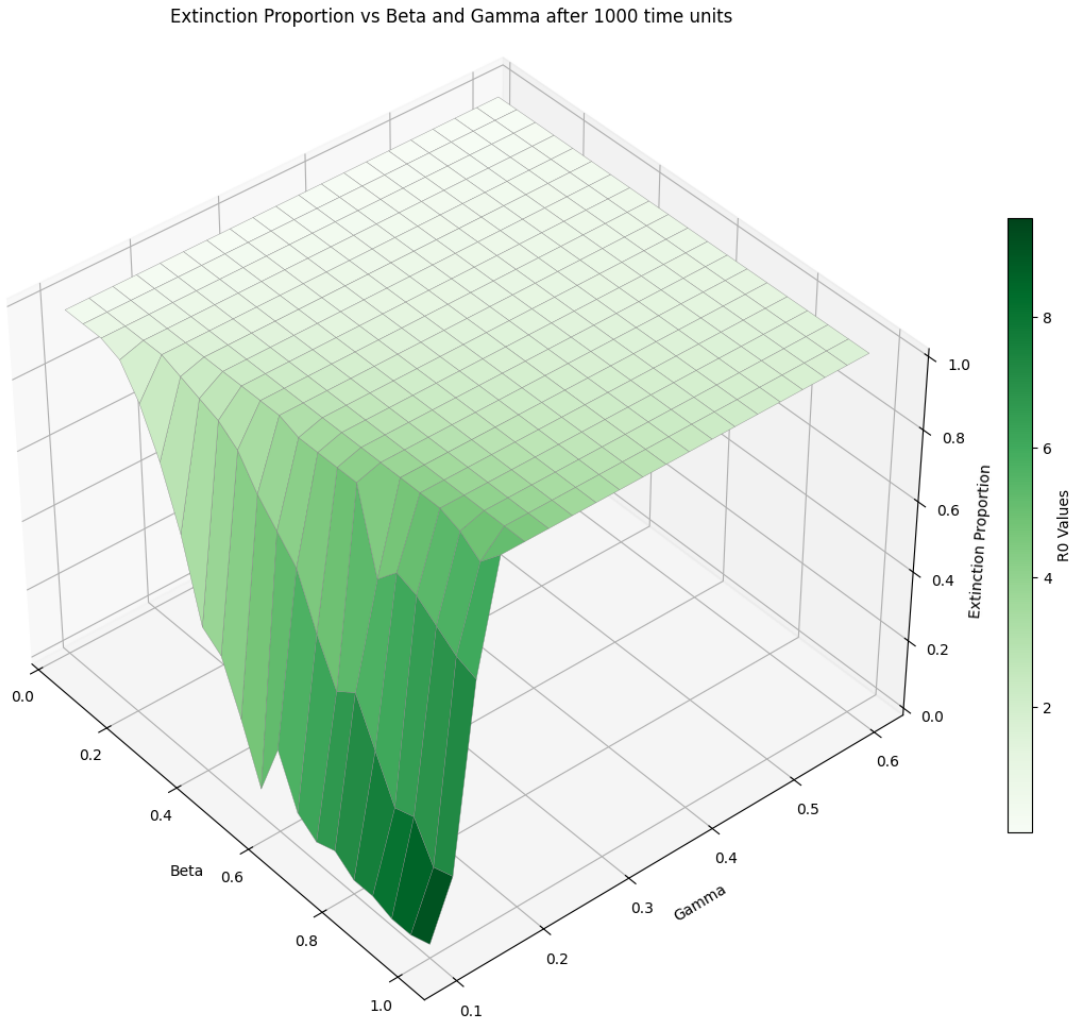


Figure 24: Infection Extinction Proportion for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = [0.1, 0.6]$, and $\beta = [0.0, 1.0]$

What these figures *do* show is that there is some structural equivalence such that equivalent R_0 values for different γ and β values have reasonably similar behavior. This means that instead of simulating for ranges of β and γ, simulations for R_0 itself may be done using the the initial $\mu = 0.005$ without worrying about unique dynamics such as those seen in the Covariance Figures 14, 13, 17.

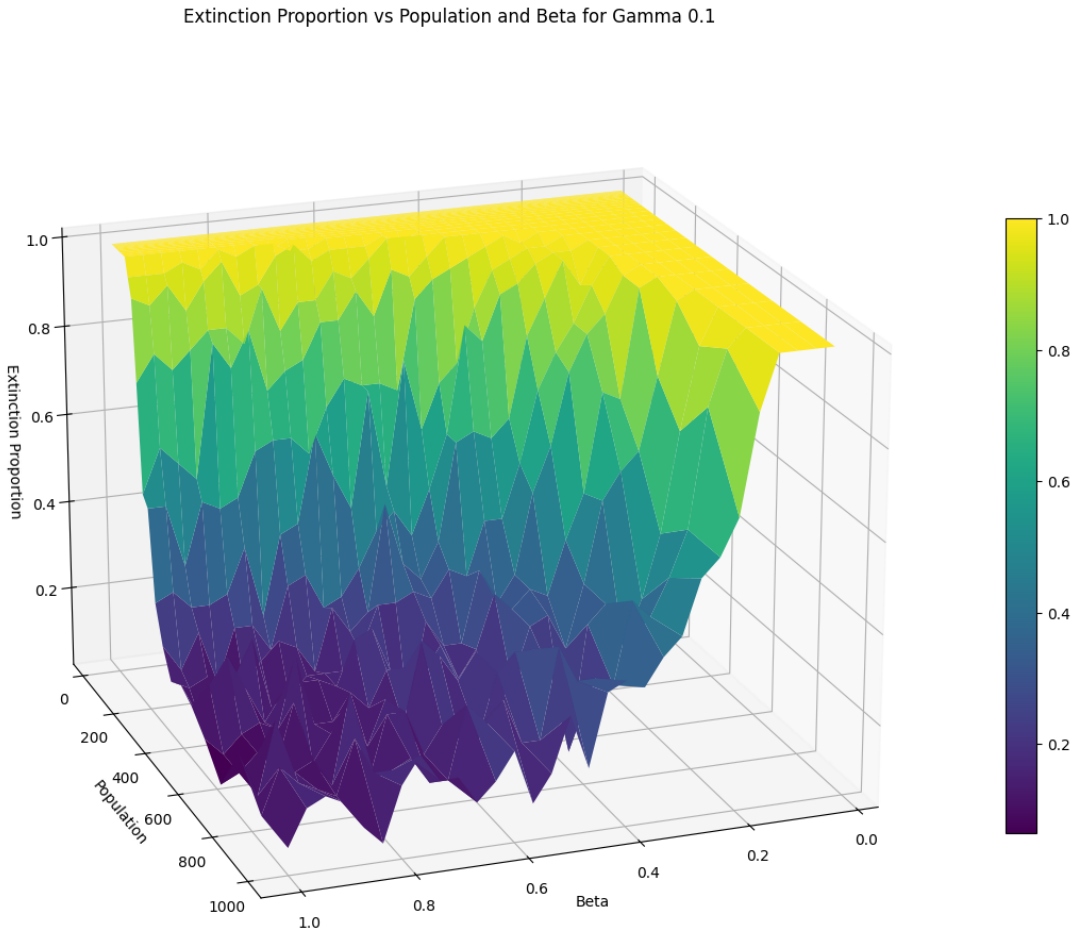


Figure 25: Infection Extinction Proportion for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = 0.1$, $\beta = [0.0, 1.0]$, and Population $= [0, 1000]$

By adjusting β as stand-in for a variable R_0 given a $\gamma = 0.1$ and a range of populations between 0 and 1000, Figure 25 shows how both R_0 and population size influence the tendency towards extinctions over 1000 iterations. This extinction proportion will rise as the time span increases, but it is apparent in the figure that both population size and R_0 effect the rate of extinction similarly such that some population size may determine whether an infection goes extinct in a reasonable amount of time within a closed population.

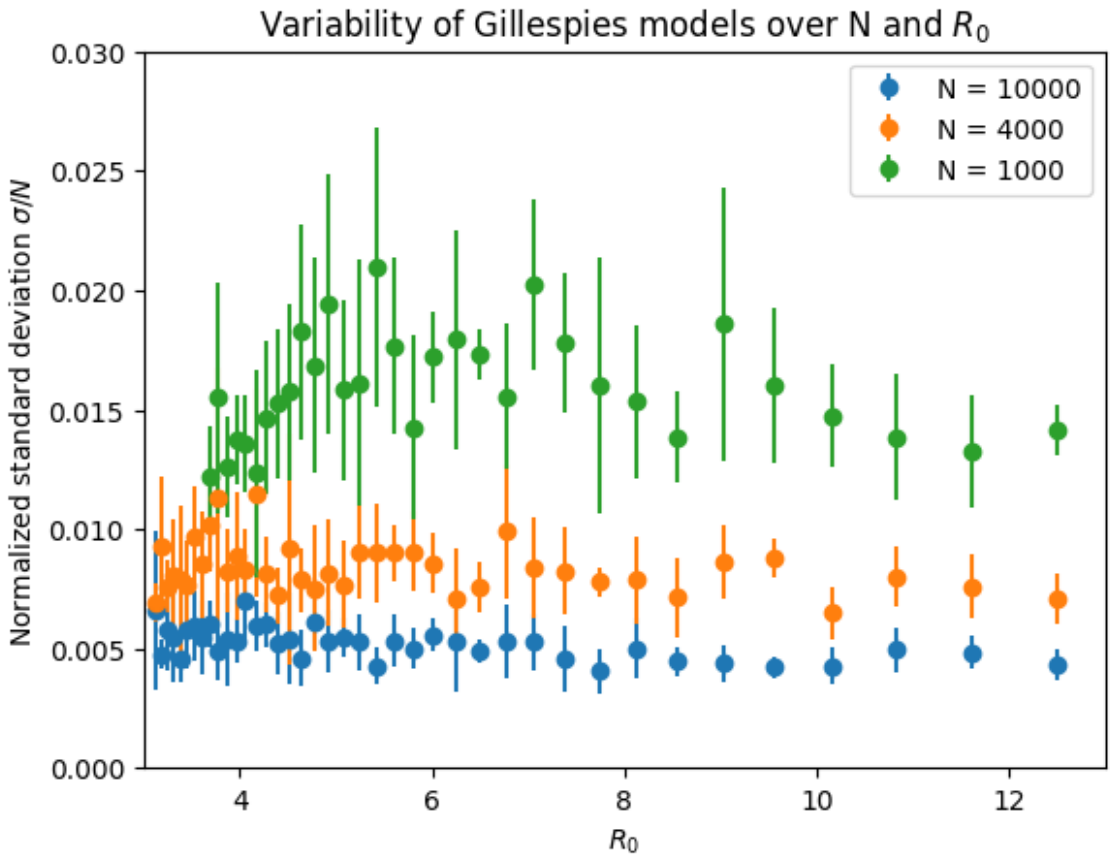


Figure 26: The normalized standard deviation of stochastic SIR models. The models were run with populations of 1000, 4000 and 10.000. The $\beta = 0.3$ and γ was varied between 0.04 and 0.16. For the $N = 1000$ models, the γ was only increased to a value of approximately 0.13, as simulations running at higher values went extinct to frequently, which caused a significant slow-down in the code. The plot shows that smaller populations have a higher normalized standard deviation, and are therefore more affected by stochastic effects. Additionally, the $N = 1000$ dataset also appears to show a peak around $R_0 = 5.5$. This might indicate that this value exhibits stochastic resonance.

This behavior is likely directly related to the variability of the stochastic simulations such that a higher variability creates larger transients which lead to more extinctions which is apparent in Figure 26 where-in large populations express less variability, especially as R_0 increases. Ultimately, this information seems to dictate some Critical Community Size where extinctions become rare and variability of the noise lacks the transient amplitude to go extinct. For 1000 iterations in Figure 27, this critical population size seems to be around $300 - 400$ individuals, although it might be expected to increase with a greater time span.

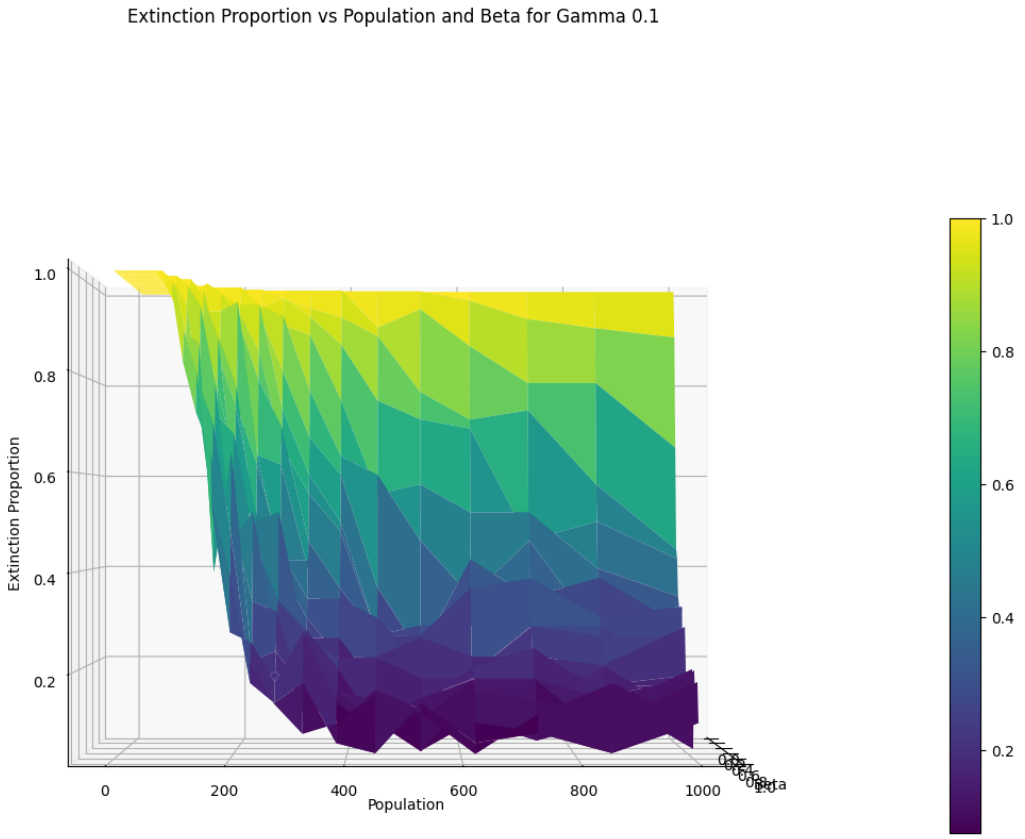


Figure 27: Infection Extinction Proportion for 1000 time steps and 50 stochastic runs $\mu = 0.005$, $\gamma = 0.1$, $\beta = [0.0, 1.0]$, and Population $= [0, 1000]$

Additional Extinction Figures are available in the Appendix B.

5.5 Network Features

A further expansion of the stochastic infection simulation is the related area of spatial modeling where-in systems are further discretized into models that take into account spatial scale. These types of models seek not only to model the parameters by which systems interact, but also the distance between individuals in these systems such that local interactions may be captured spatial spread may be simulated [1].

Networks are one such model that are able to abstractize spatial interactions and explore how the connections between individuals or groups of individuals might influence the spread of disease. Features of the network are highly influential in how an infection might spread from node-to-node as overall rate of infection for the network depends on how connected infected nodes are to susceptible nodes.

5.5.1 Barabási-Albert Network

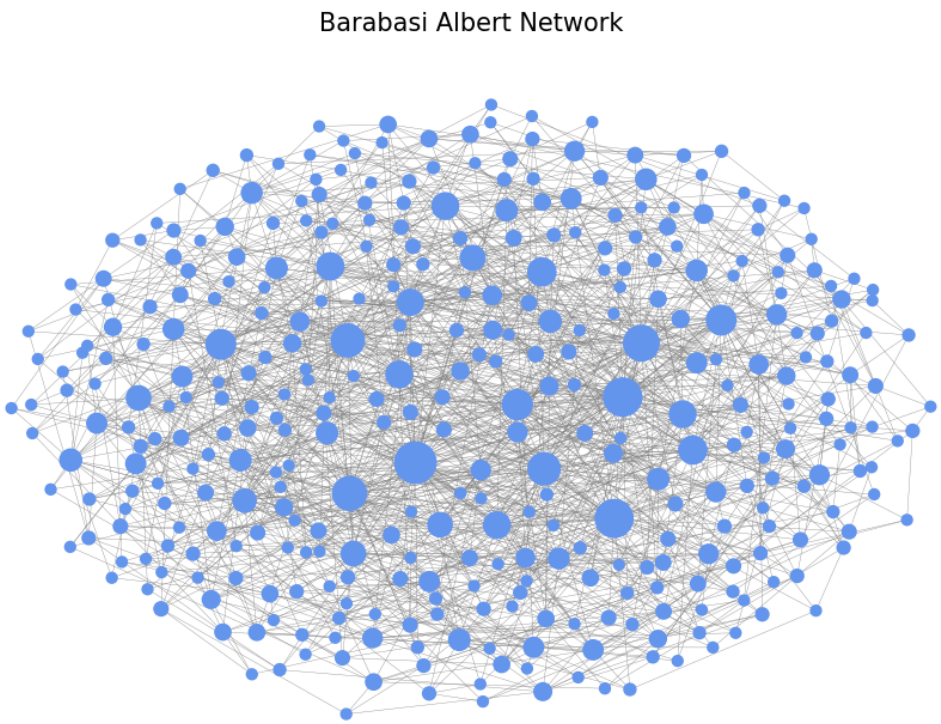


Figure 28: Barabási-Albert Network with 374 Nodes and 7 Average Edges per Node with degree-dependent node size visualization

A Barabási-Albert Network with 374 nodes and 7 Edges per new node in Figure 28 presents a Scale-Free Network in which a few nodes known as "hubs" are highly-connected while most other nodes have far fewer connections. By plotting the network Degree Distribution in Figure 29, it's possible to see how a vast majority of nodes have 10 or fewer edges while a few singular nodes have degree counts over 40.

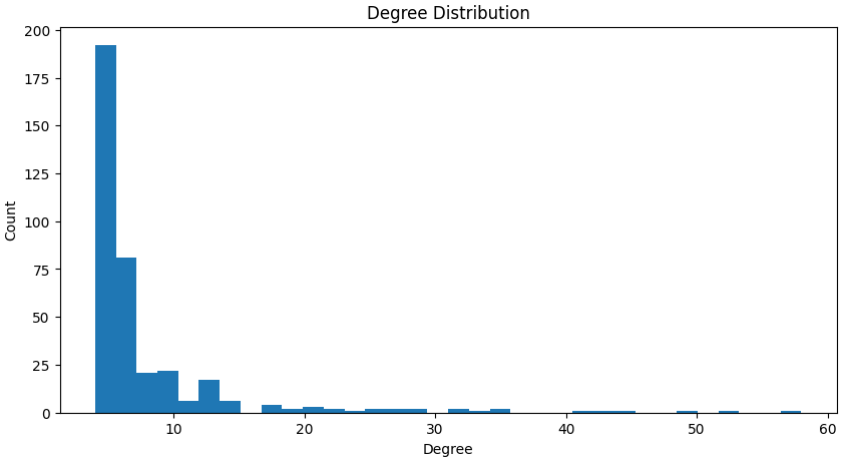
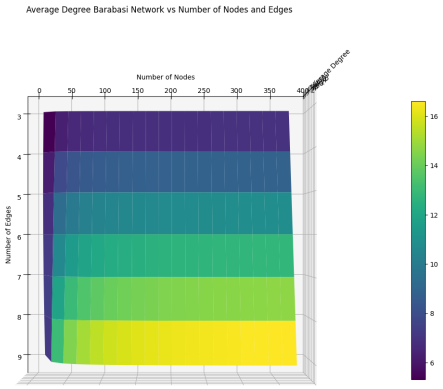


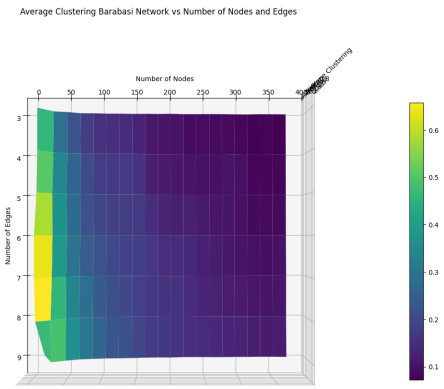
Figure 29

When plotting the Barabási-Albert network over a range of edge connections $[3, 9]$ and number of nodes $[0, 374]$ in Figure 30a, the trend exhibits an apparent linear growth over the range

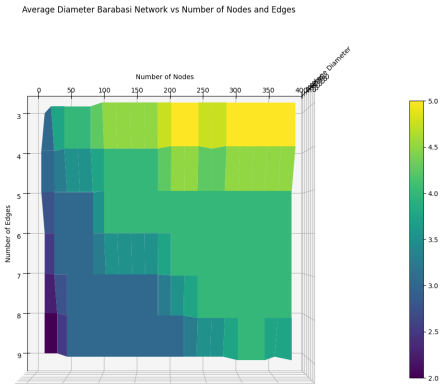
of edge connections with a drop-off as the number of nodes decreases due to limited connection options and an incomplete scale-free preferential attachment algorithm.



(a) Average Degree



(b) Average Clustering



(c) Diameter

Figure 30: Barabási-Albert Network features for edge connections = $[3, 9]$ and number of nodes = $[0, 374]$. Larger Figures are available in the Appendix B.

The inability for the Barabási-Albert network to complete the preferential node attachment process with fewer nodes also causes initial nodes in the process to cluster such that a smaller number of nodes will have a higher clustering coefficient as seen in Figure 30b. This figure also shows how the scale-free method of network creation has very little clustering with a sufficient number nodes due to the tendency towards "hubs" of connectivity that make nodes and their neighbors unlikely to be connected beyond their direct partners.

Additionally, as the number of nodes increases, the diameter trends upwards as per Figure 30c while a decrease in the number edges per node connection sees a similar increase in diameter. Due to the scale-free nature of the network, this traversal of of the longest shortest path is generally minimal as the "hub" nodes act as transfer nodes for movement across the network, connecting many nodes together quickly through centralized connections.

5.5.2 Watts-Strogatz Network

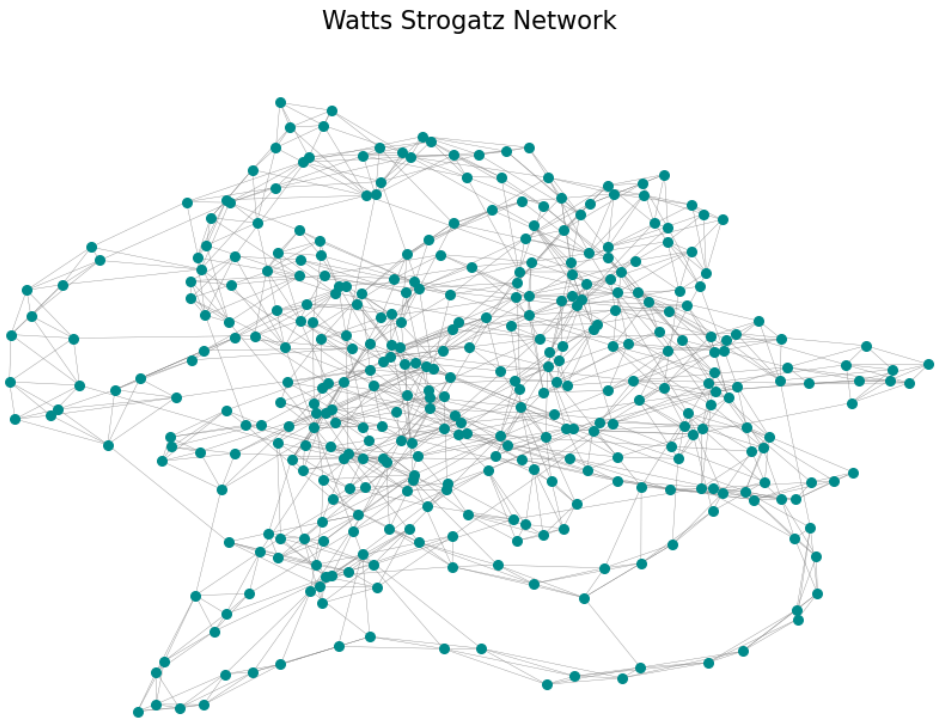


Figure 31: Watts-Strogatz Network with 374 Nodes, 0.1 Edge Rewire Probability, and 7 Edges per Node with Spring Layout

A Watts-Strogatz Network with 374 Nodes, 7 edges per node, and a 0.1 probability for an edge to rewire creates an effective "small-world" network in which clustering is common and average path lengths between nodes is short, creating connected "communities" of nodes like those in Figure 31.

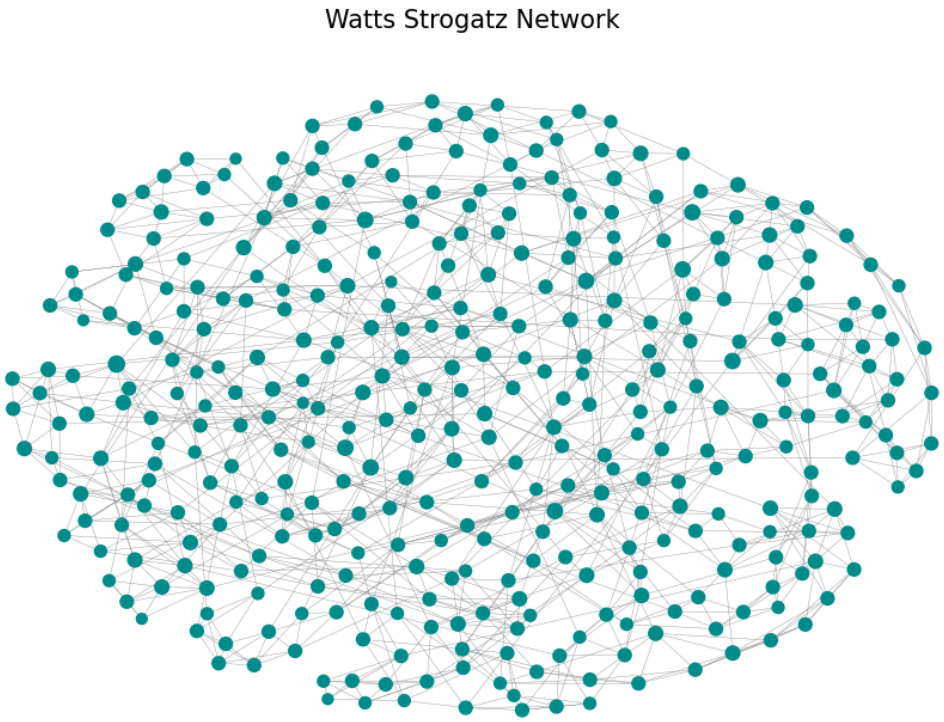


Figure 32: Watts-Strogatz Network with 374 Nodes, 0.1 Edge Rewire Probability, and 7 Edges per Node with degree-dependent node size visualization

This network sees an equally distributed number of edge across all nodes which are then rewired randomly according to the rewire chance (In this case, 0.1) such that shortcuts between nodes are created and communal clusters develop at random. The randomness of the rewiring distributes the node degrees in a narrow peak distribution around the original number of edges as per Figure 33, maintaining the same average degree as that of a random regular network without rewiring.

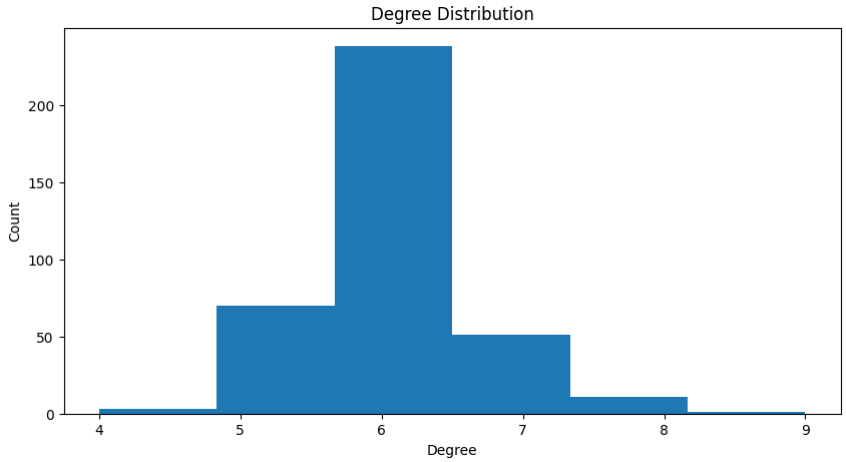
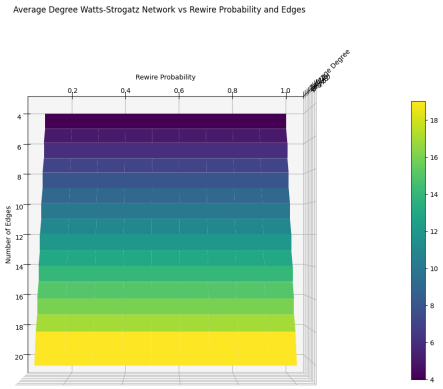
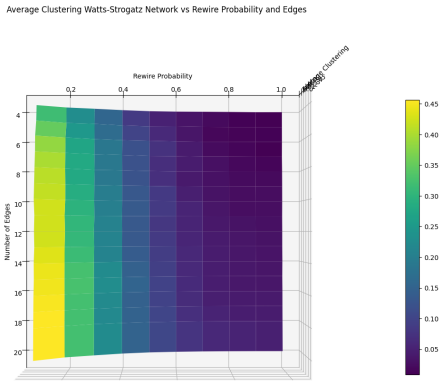


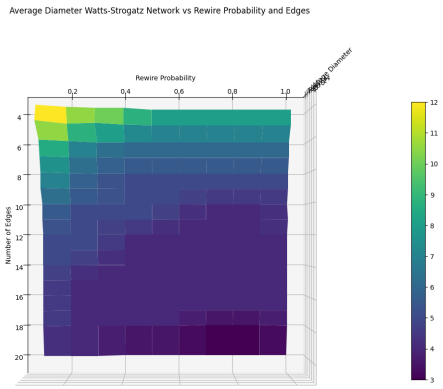
Figure 33: Degree Distribution of Watts-Strogatz Network with 374 Nodes, 0.1 Edge Rewire Probability, and 7 Edges per Node



(a) Average Degree



(b) Average Clustering



(c) Diameter

Figure 34: Watts-Strogatz Network features for Rewire Probability = $[0.1, 1.0]$ and number of edges = $[4, 20]$ and number of nodes = 374. Larger Figures are available in the Appendix B.

Likewise, this average degree number is observable in Figure 34a as the degree value increases linearly across a range of edge rewiring probabilities from 0.1 to 1.0 and number of edges 4 to 20 for 374 nodes.

The clustering behavior appears inverse to the small-worlds clustering behavior described previously, but in reality, the average clustering coefficient increases as random rewiring decreases in Figure 34b. However, while one might expect a higher clustering coefficient with a small-world network, the starting network before rewiring is already a fully connected, highly clustered network, so average clustering for the entire network decreases with rewiring increases.

The tendency for network diameter to increase as the number of edge per node decreases is also readily apparent in Figure 34c for this network type, but there is also apparent behavior that sees a large increase to diameter with a low rewire probability and a low number of edges. This is likely caused by the limited number of rewiring in the network, causing node connectivity paths to be severed easily and not enough rewiring having occurred to redistribute the adjusted edges equally.

5.5.3 Erdős-Rényi Network

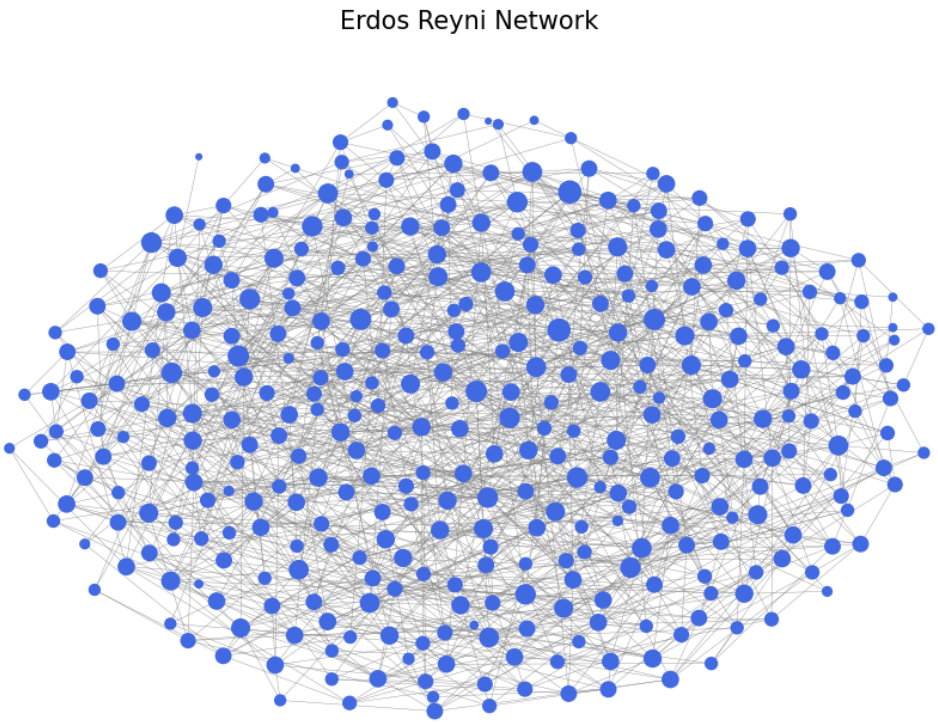


Figure 35: Erdős-Rényi Network with 374 Nodes and 0.018 Edge Probability with degree-dependent node size visualization

The Erdős-Rényi Network in Figure 35 is a random network that is formed by connecting 374 nodes at random given that each pair of nodes has a probability of being connected equal to $0.018 = 7 edges/374 nodes$. The random probability of the edge connections creates a degree distribution that is approximately binomial as per Figure 36 with a peak around $7 - 7.5$, reflecting the edge/node calculation made for edge probability.

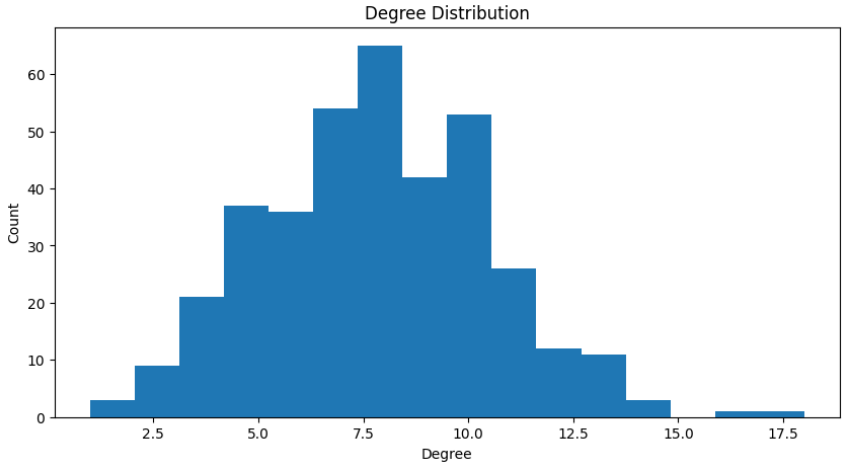
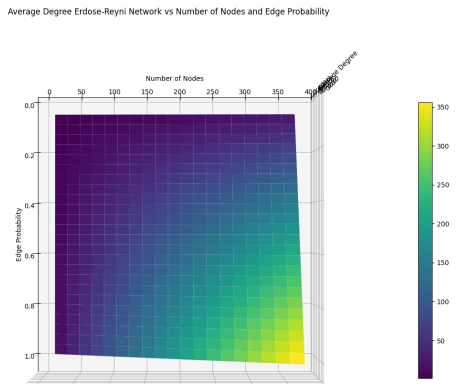
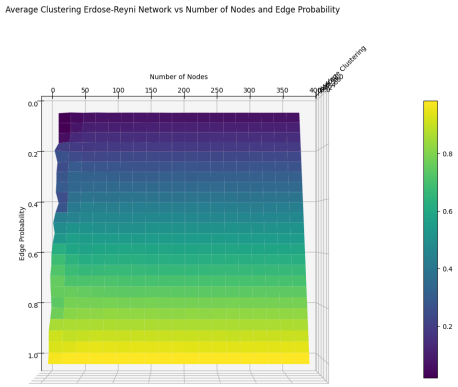


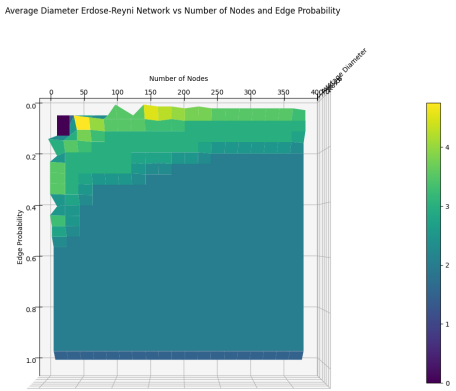
Figure 36: Degree Distribution of Erdős-Rényi Network with 374 Nodes and 0.018 Edge Probability



(a) Average Degree



(b) Average Clustering



(c) Diameter

Figure 37: Erdős-Rényi Network features for Edge Probability = $[0.05, 1.0]$ and number of nodes = $[10, 374]$. Larger Figures are available in the Appendix B.

This randomness of the network creates a completely linear relationship between the range of edge probabilities $[0.05, 1.0]$ and the number of nodes $[10, 374]$ such that a random network with an edge probability of 1.0 will see a will see maximum connectivity between all nodes and linearly decreasing connectivity otherwise in Figure 37a.

Similarly, the average clustering of this network increases linearly across the range of edge probabilities where all neighbors will be connected to all other neighbors as the the probability increases as per Figure 37b. Uniquely, when the number of nodes is low, there is an increased noise in the clustering do to the low number of nodes causing a lack of degree distributions amongst the small number of nodes.

This noise for small numbers of nodes is also apparent in the Diameter of the network in Figure 37c alongside the continued observation that a lower number of edges between nodes sees an increased diameter, although in this case, there is also the added creation of possible unconnected graphs at the absolute lowest edge probabilities and node numbers.

5.5.4 Random Regular Network

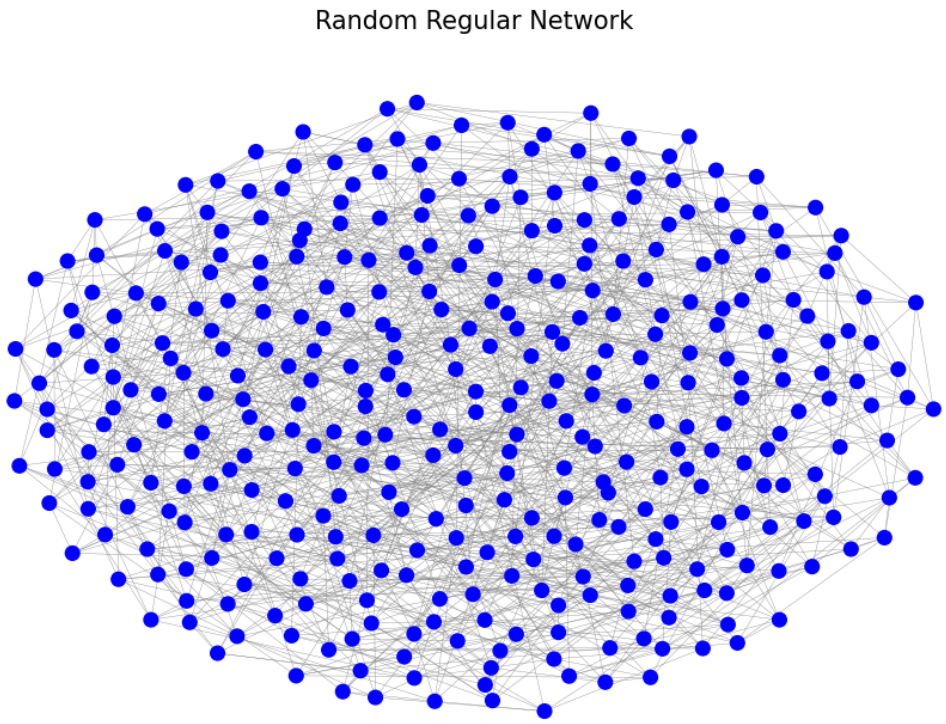


Figure 38: Random regular Network with 374 Nodes and 7 Edges per Node with degree-dependent node size visualization

The basic Random Regular network in Figure 38 is a type of regular network that has a fixed degree = 7 for each of the number of nodes = 374 such that all edges are randomly distributed amongst nodes. This network creates a homogeneous structure and a uniform baseline such that degree distribution in Figure 39 is singular at degree 7.

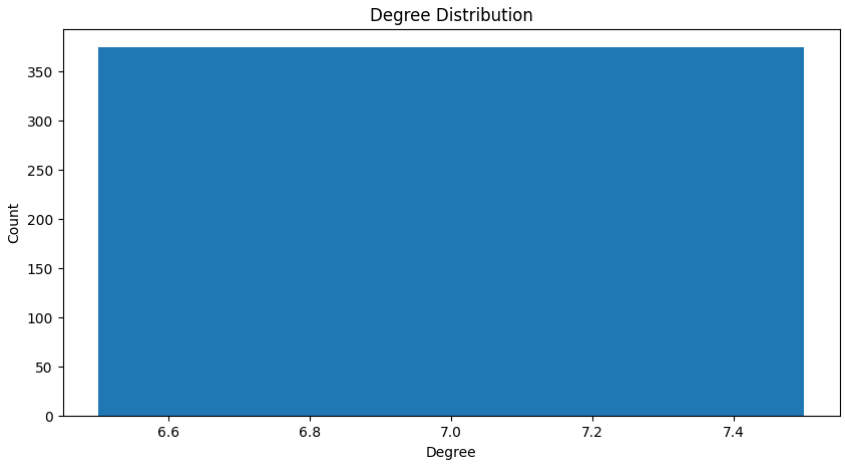


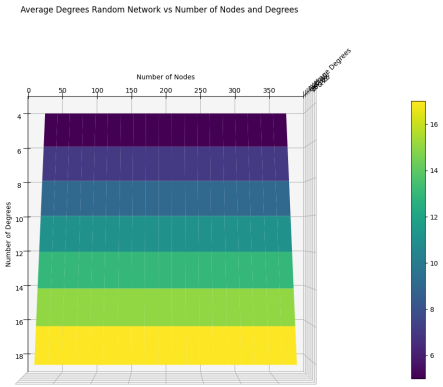
Figure 39: Degree Distribution of Random Regular Network with 374 Nodes and a Degree of 7

This homogeneous structure makes the average degree over a range of nodes [25, 374] and

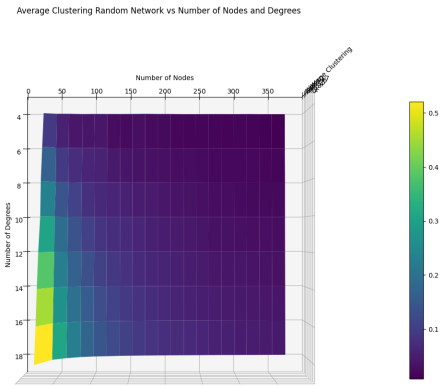
degrees $[4, 18]$ completely linear in Figure 40a as the number of degrees per node increases, so does the average degree.

Average clustering also continues the trend that sees highly locally connected networks, caused by factors such as high degree number and low node number, have larger clustering coefficients in Figure 40b.

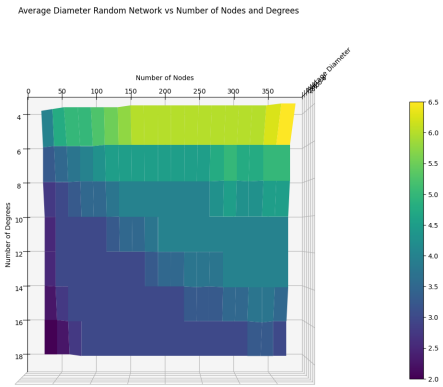
Figure 40c for the Random Regular network takes on a similar shape to that of the Barabasi Network in Figure 30c even as the structures of the networks highly differ. The key feature of both is the high number of nodes and low number of edges which causes the structure of both to become more similar and homogeneous, creating a longer path of connected nodes rather than a full network with many connections.



(a) Average Degree



(b) Average Clustering



(c) Diameter

Figure 40: Random Regular Network features for Number of Degrees = $[4, 18]$ and number of nodes $[25, 374]$. Larger Figures are available in the Appendix B.

5.5.5 Sociopatterns Network

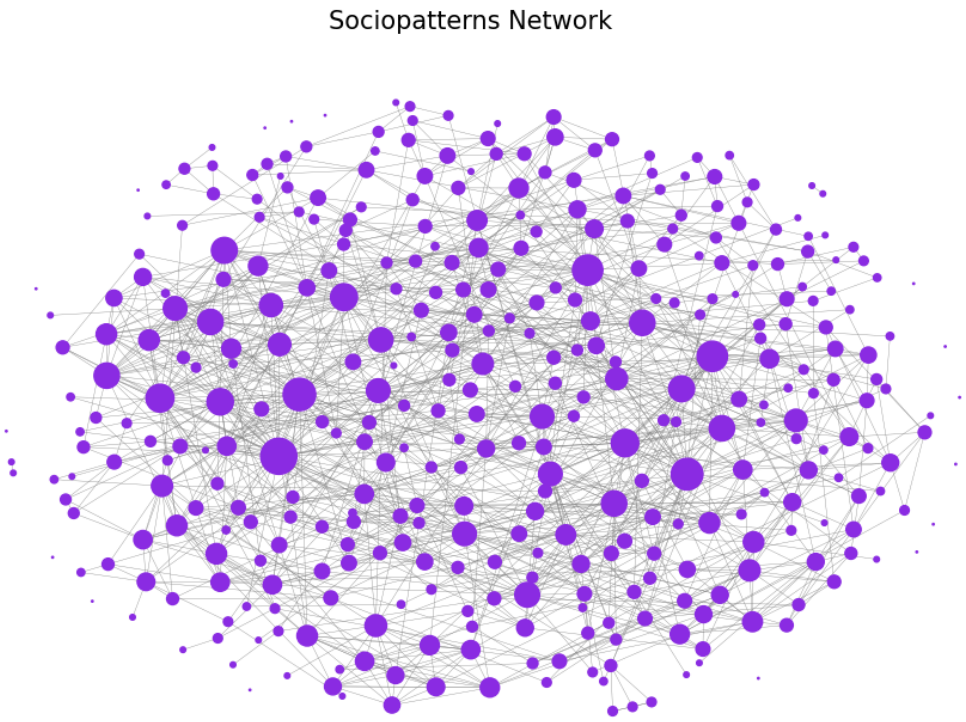


Figure 41: Sociopatterns Network with 374 Nodes and 7 Edges per Node with degree-dependent node size visualization

Using a real-world network dataset, it becomes possible to compare different network structures to the real-world system in Figure 41 such that each preceding network was based on the direct network features of this data, including the number of nodes = 374 and the average number of edges $\simeq$ 7. The degree distribution of the sociopatterns network in Figure 42 is noisier than the mathematically created networks and expresses a range of degree values including a number of unconnected nodes some high outliers.

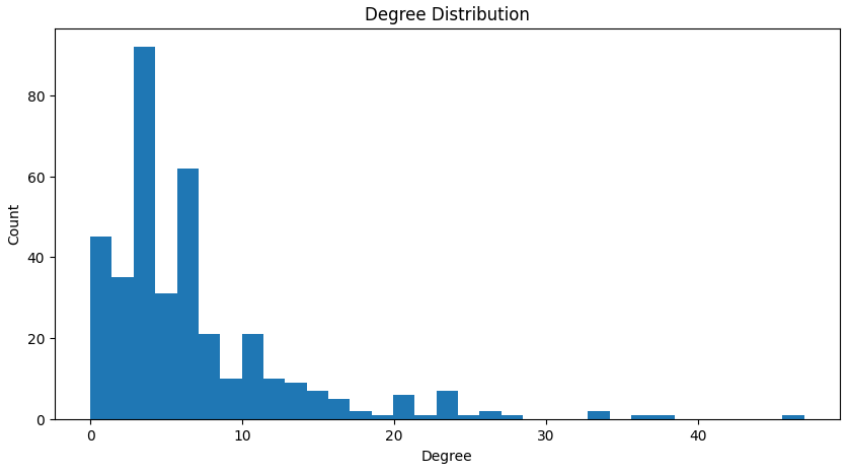


Figure 42: Degree Distribution of Random Regular Network with 374 Nodes and a Degree of 7

5.6 Network Comparisons

Using the sociopatterns network features in Figure 41 and the node and edge equivalent estimates for the Barabási-Albert, Erdős-Rényi, Watts-Strogatz, and Random Regular Networks, it is possible to roughly compare each of the features of the networks on a similar scale. The baseline for each network is 374 nodes, observable in Figure 43, as well as roughly 7 edges per node or degrees per node on average in Figure 44 given the different parameter types of each network.

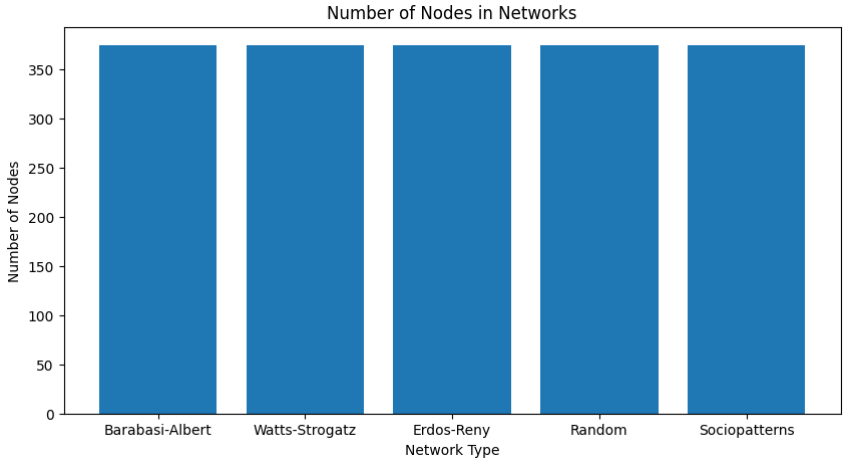


Figure 43: Comparison of Network Nodes with 374 Nodes and a Degree of 7

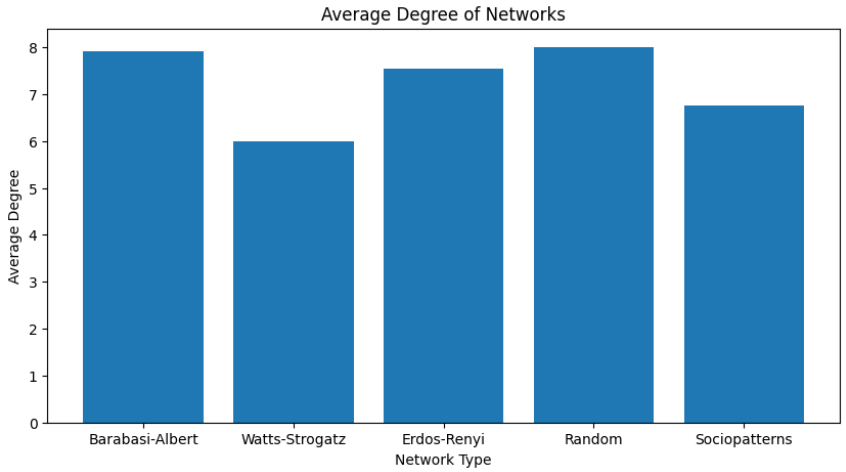


Figure 44: Comparison of Network Average Degree with 374 Nodes and a Degree of 7

In Figure 45 it becomes possible to see earlier observed clustering behaviors in the Watts-Strogatz network from Figure 34b such that the clustering of the network leads all other compared networks. The sociopatterns network also exhibits considerable clustering overall, which puts it in contrast with the Barabási-Albert network even though both networks shared slightly similar degree distributions in Figures 29 and 42.

Both the Random regular network and the Erdős-Rényi network place themselves at the bottom of the clustering coefficient values given baseline specifications similar that of the sociopatterns network. For the Erdős-Rényi network, these means the edge probability is a low 0.18 to limit edge creation and thus clustering, but overall, both of these networks are unlikely to exhibit natural clustering behaviors as their random distribution of edges minimizes this behavior

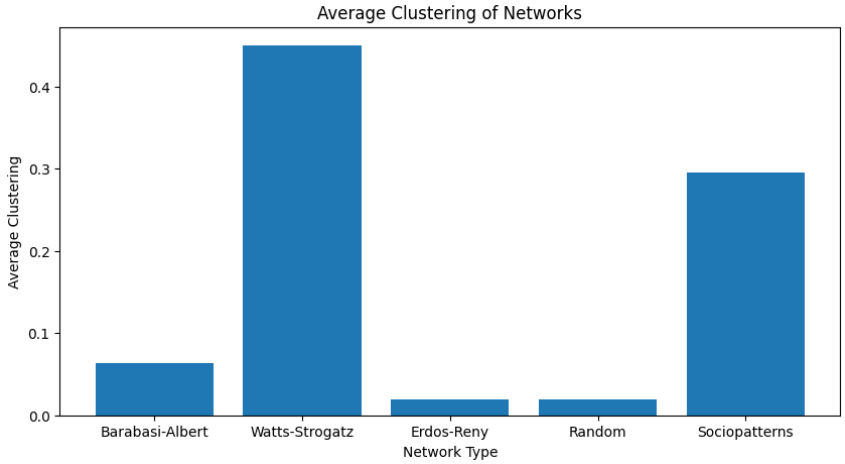


Figure 45: Comparison of Network Average Clustering with 374 Nodes and a Degree of 7

Given that the sociopatterns network is unconnected, it technically doesn't have a valid diameter and as such is not comparable to the connected graphs, but the similar baseline features show how the longest shortest path of each network compare to one-another in Figure 45. The Random Regular and Barabási-Albert networks end up nearly identical in diameter which matches the previous observation of similarity between diameters in Figures 30c and 40c across a range of parameters, while the Erdős-Rényi network is only slightly larger in comparison but expresses a completely different diameter behavior in Figure 37c.

By removing unconnected nodes and components, the sociopatterns network diameter may be found to be about equivalent to the Watts-Strogatz network without the small-world structure, calling to attention the fact that clustering coefficient in Figure 45 may correlate somewhat with diameter.

The Watts-Strogatz networks has the other largest diameter of the networks likely due to the combination of the 0.1 rewire probability and the baseline 7 number of edges in the network, causing this specific set of parameters to be a possible outlier. Likewise, given a different set of parameters, it would be expected that the Random Regular Network to have the largest diameter given that each node would have equal number of edges without shortcuts to any other nodes.

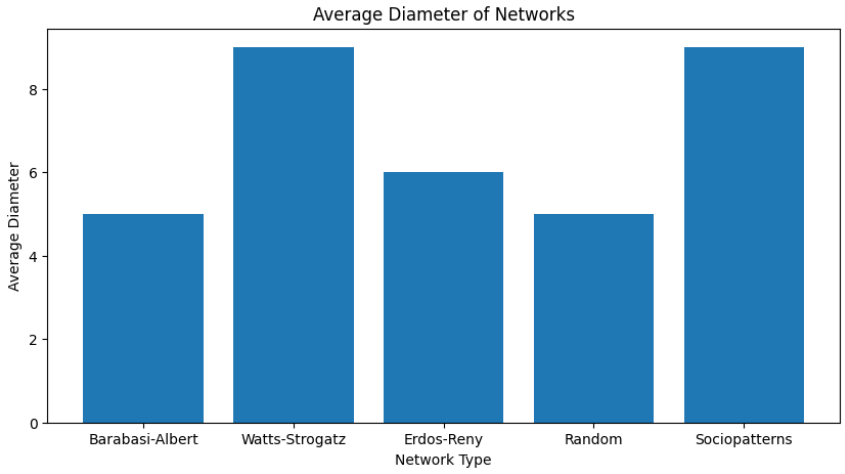


Figure 46: Comparison of Network Diameter with 374 Nodes and a Degree of 7

5.7 Infected Network

By adding implementing an infection SIR model without demography on each of these networks, it becomes possible to see how network structure and dynamics might influence the spread of disease between connected nodes. By selecting a random set of 5 nodes to infect per network and setting parameters such that $\gamma = 0.05$ and β is some range of values, comparisons may be made to the normal behavior of a deterministic model with a population equal to the number of nodes in each network, 374.

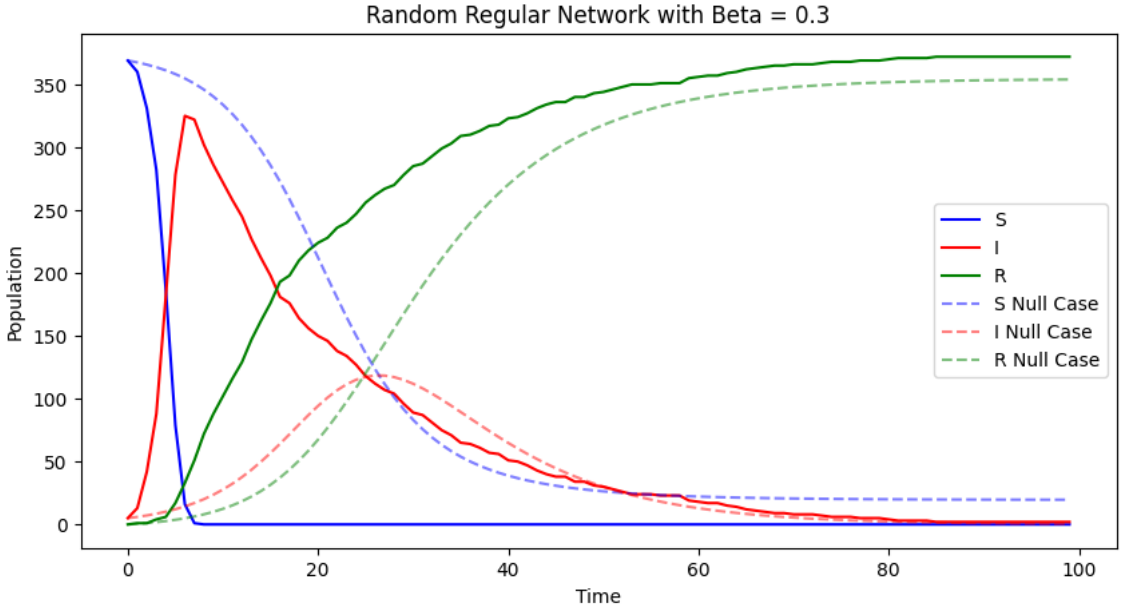
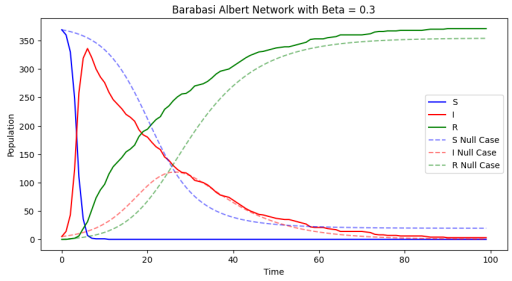


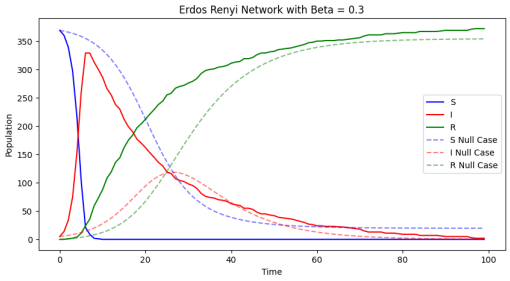
Figure 47: SIR Infection without Demography on the Random Regular Network for $\gamma = 0.05$, $\beta = 0.3$, population/nodes = 374, and 5 initial infected individuals/nodes.

In the Figure 47 simulation of infection on the baseline Random Regular Network from Figure 38 with $\beta = 0.3$, the contrast between the deterministic and network infections express a network that sees a faster infection and higher peak. This dynamic of the network infection adjusts what might be the R_0 of system to be higher than that of the deterministic system, making the network more sensitive to parameter changes of the SIR.

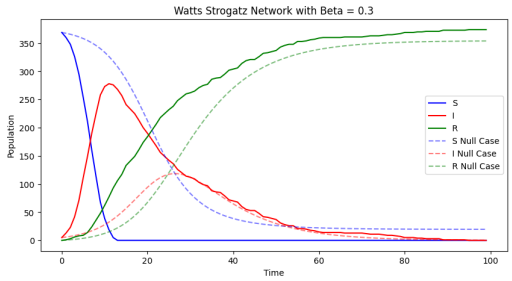
This feature is also apparent in the Barabási-Albert, Erdős-Rényi, Watts-Strogatz, and sociopatterns networks in the group of Figures 48 with the same $\beta = 0.3$, expressing a common difference between deterministic and network infections. The magnitude of this difference likely depends on the relationship between parameters and infected node selection such that clustering, average degree, or diameter might influence infection speed. By comparing infection models for a range of β for all models, it becomes more apparent how a change in parameters influences the effective Reproductive Rate on each network type.



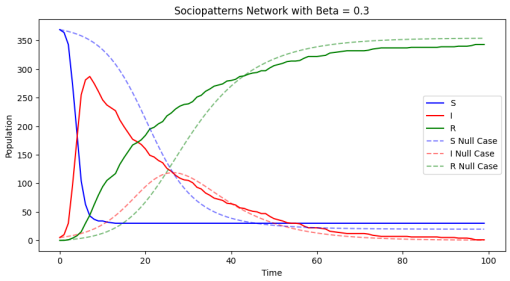
(a) Barabási-Albert



(b) Erdős-Rényi



(c) Watts-Strogatz



(d) Sociopatterns

Figure 48: SIR Infection without Demography for different networks for $\gamma = 0.05$, $\beta = 0.3$, population/nodes = 374, and 5 initial infected individuals/nodes. Larger Figures and additional β Figures are available in the Appendix B.

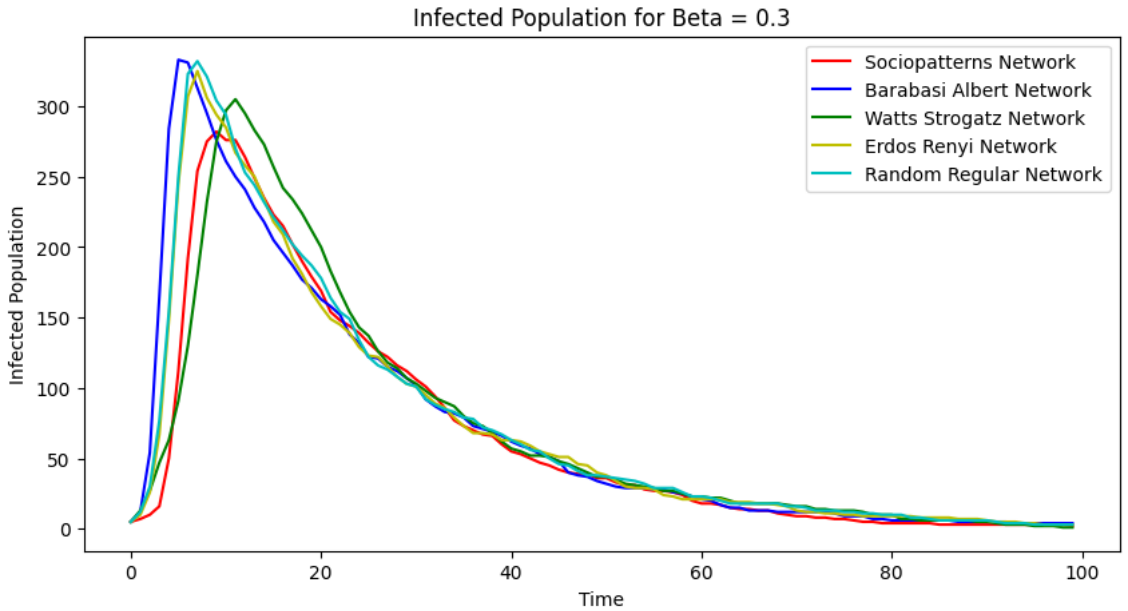


Figure 49: Infection without Demography on different networks for $\gamma = 0.05$, $\beta = 0.3$, population/nodes = 374, and 5 initial infected individuals/nodes.

Figure 49 shows how the relatively high β value creates infections that spread through networks at similar rates, minimizing the differences between network structures that might naturally mitigate spread. It also shows how each network similarly differs from the deterministic SIR model in how quickly infection spreads and peaks over the time span.

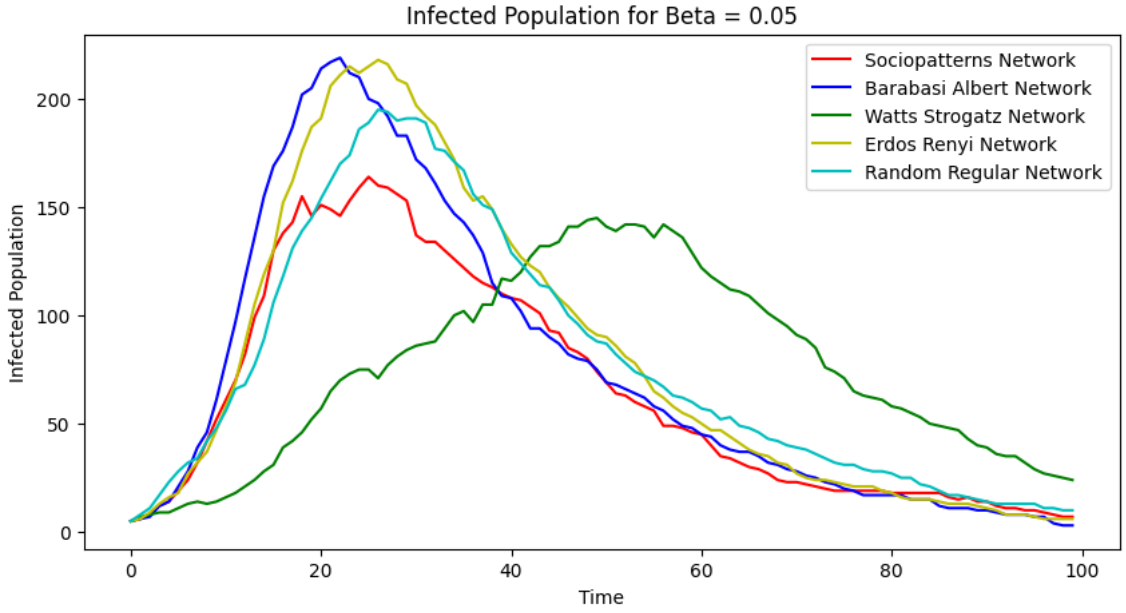


Figure 50: Infection without Demography on different networks for $\gamma = 0.05$, $\beta = 0.05$, population/nodes = 374, and 5 initial infected individuals/nodes.

By choosing a lower $\beta = 0.05$ in Figure 50, the influence of network structure becomes more apparent with the Erdős-Rényi, Barabási-Albert, and Random Regular networks retaining a higher infection peak, followed by the sociopatterns network, and the dampened Watts-Strogatz network.

Given what is known about the scale-free Barabási network and the "hub" structure, the preferential node attachment towards hub nodes makes infections on these nodes catastrophic to the network and makes full network infection quick due to the lower diameter of these networks as per Figure 46.

The Erdős-Rényi Random network and the Random Regular network behave similarly to the Barabási network with a more baseline structure of randomly allocated nodes which cause their average clustering, diameter, and average degree to behave similarly in Figures 46, 44, and 45. The Random Regular Network, however, exhibits a slight dampening compared to the Erdős-Rényi network, which is likely an effect of the degree distribution between nodes wherein the Erdős-Rényi network exhibits a binomial distribution of degrees in Figure 36 while the Random Regular network has a singular distribution in Figure 39. This difference in distribution likely influences the speed of infection depending on if the higher degree nodes of the Erdős-Rényi network are infected earlier in the process.

Additionally, the dampening on both the Sociopatterns network and the Watts-Strogatz network correlate with their average clustering coefficient and diameter in Figures 45 and 46 such that a higher clustering coefficient and higher diameter may see decreased disease infection rates. This may also be due to the estimations required to create the network edges causing a slight difference in average node degree between networks in Figure 44 such that the correlation may instead be between higher infection rate and higher average degree.

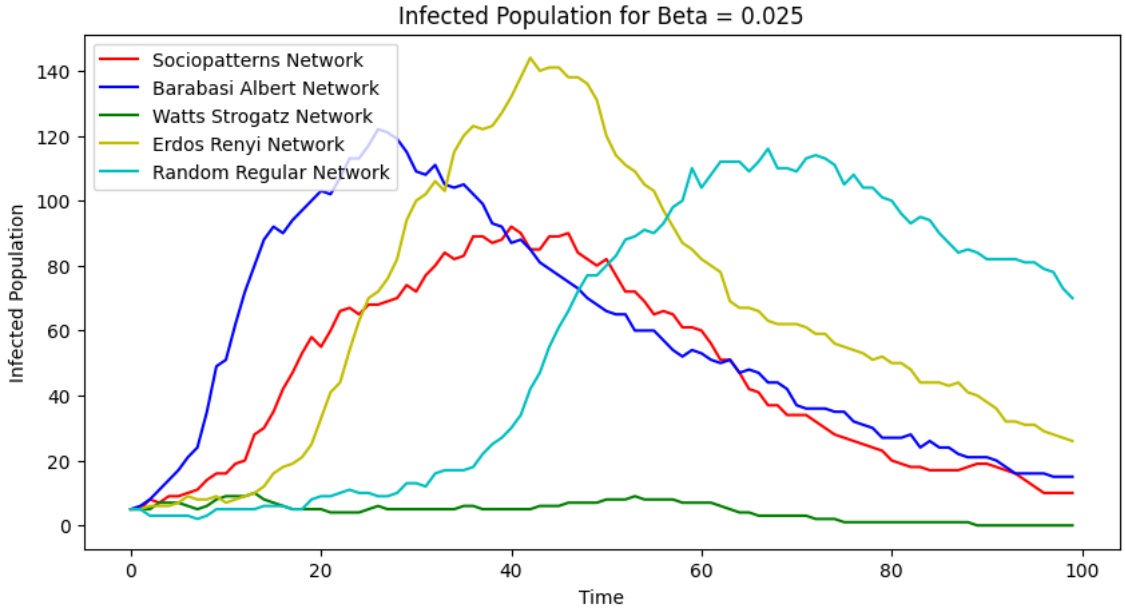


Figure 51: Infection without Demography on different networks for $\gamma = 0.05$, $\beta = 0.025$, population/nodes = 374, and 5 initial infected individuals/nodes. Additional Figures with different parameters are available in the Appendix B.

This idea is further supported by adjusting the parameters of the infection such that $\beta = 0.025$, causing all network infections to dampen, but for Watts-Strogatz and Random regular networks to dampen the most with a less pronounced effect on the Sociopatterns network. The Watts-Strogatz network in both Figures 50 and 51 show how the small-world structure of the network creates communities of nodes that may be isolated if infection pressure is not high enough to travel through the community-connecting edges.

The Random Regular network retains a peak similar to that of the Barabási or Erdős-Rényi networks, but sees its infection spread begin much later in the time span compared to any other network in Figure 51. This is likely attributed to the equal distribution of connectivity and degree that sees a baseline transmission rate dependent on the singular degree number per node rather than any specific high-degree nodes in the network. This idea also points to the influence of node selection that may cause the networks with wider degree distributions to have a faster or slower spread of infection depending on the degree of an infected node.

To view animated network infection spread videos for different network models and parameters, refer to the project drive.

5.8 Vaccination Strategies

With the concept of infection spread dependent on the degree of the randomly selected infected nodes, it becomes important to gain an understanding of how infections on these networks might be mitigated using intervention strategies. One such intervention strategy is the act of vaccination such that susceptible individuals might be removed from possible infection, thus blocking the possibility for further spread between the vaccinated node and its neighbors.

In an ideal world, it is possible to vaccinate every individual in the network immediately and thus stop any infection from ever occurring, but this is not usually logistically feasible to do and as such, vaccination must be made selectively to contain infections within networks. As such, it also becomes important to know who must vaccinated given that infected individuals are not always obvious amongst a large group, making testing an important aspect of vaccination strategy.

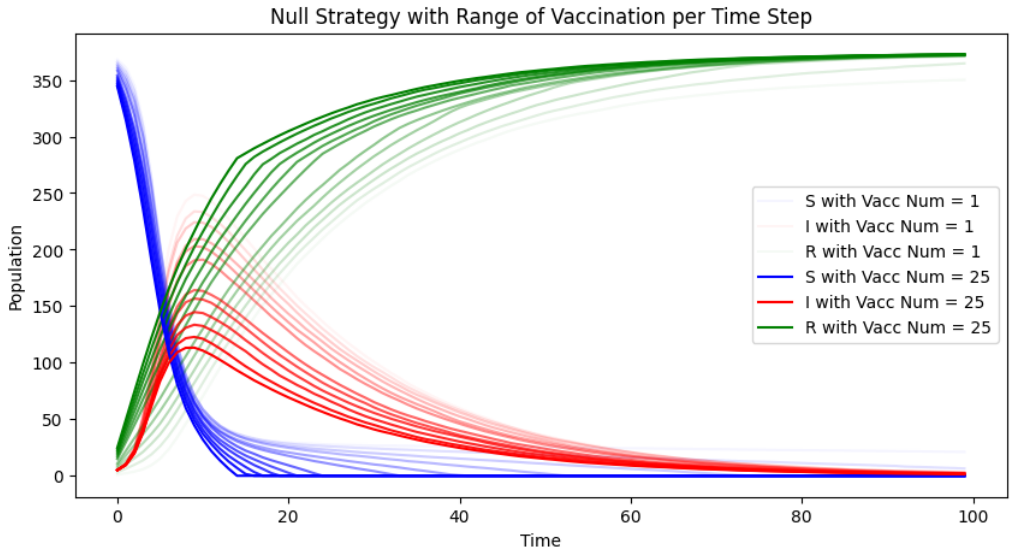


Figure 52: Null Random Vaccination Strategy with no Testing on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, and a range of vaccinations per time step = $[1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$

However, given no tests and no vaccination strategy for an infection on the Sociopatterns network dataset where $\beta = 0.175$ and $\gamma = 0.05$, the best possible option is to vaccinate randomly assuming that everyone is a viable susceptible candidate. This process in Figure 52 across 500 runs with different infected nodes would see both susceptible individuals vaccinated and added the recovered population, as well as infected individuals vaccinated without effect. The trend of dampening in Figure 52 shows how this strategy would deter infection with an increasing number of vaccination per time step, but would not stop an infection from spreading even as the number of vaccinations reached 7% of the nodes at at each time step.

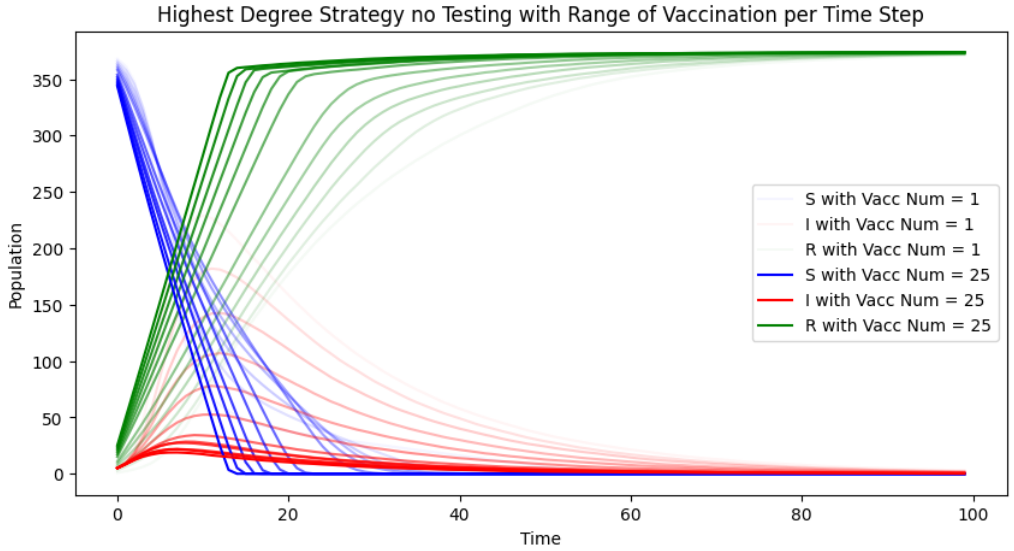


Figure 53: High Degree Vaccination Strategy with no testing on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, and a range of vaccinations per time step = $[1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$

Given the idea that infections spread more quickly through high degree nodes than low degree nodes, the vaccination strategy may be updated to include a prioritization of these nodes

using what is known about the network structure at any given time. By creating a stack of nodes prioritizing highest degree, a new strategy for a range of vaccinations per iteration = [1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25] may be plotted on Figure 53. This new figure expresses the the downward trend of the infection peak in the Sociopatterns network for 500 new averaged runs such that infections reduce to near-zero and the effect of vaccination is amplified, requiring fewer vaccinations to create a stronger dampening effect.

With the opportunity to test individual nodes for infections comes the opportunity to develop a dynamic strategy by which vaccines may be delivered – dependent on both high degree priority and testing for infections. This strategy can act on the discovery of infected nodes by attempting to isolate them from connected nodes that are susceptible, all the while prioritizing nodes of the highest degree. With testing focused on the highest degree neighbors of infected nodes found previously, followed by the highest degree nodes of the full network, the order of vaccination priority becomes:

1. Highest Degree susceptible neighbors of infected nodes
2. Highest Degree susceptible nodes that were tested this time step given vaccinations are available still
3. Highest Degree untested nodes if no more testing is available

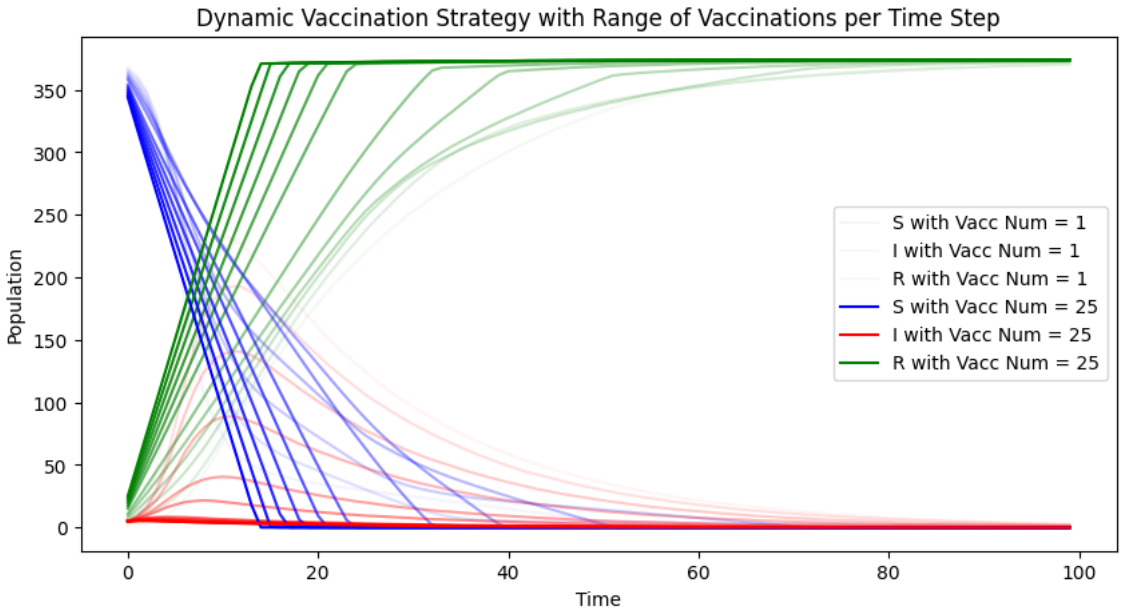


Figure 54: Dynamic Vaccination Strategy with testing on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of vaccinations per time step = $[1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$, a maximum number of total testing = 200, and a number of tests per iteration = 200

Given these priorities for vaccination and the parameters of the infection $\beta = 0.175$, $\gamma = 0.05$, the testing and vaccination parameters for the strategy may be specified to be include a maximum number of total testing = 200 and an initial number of tests per iteration = 200 while simulating vaccinations per time step = $[1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$. The Figure 54 shows the continued trend of vaccination that dampens infections as the number of vaccinations increase with an apparent improvement of the intervention's effect compared to the highest degree vaccination strategy without testing.

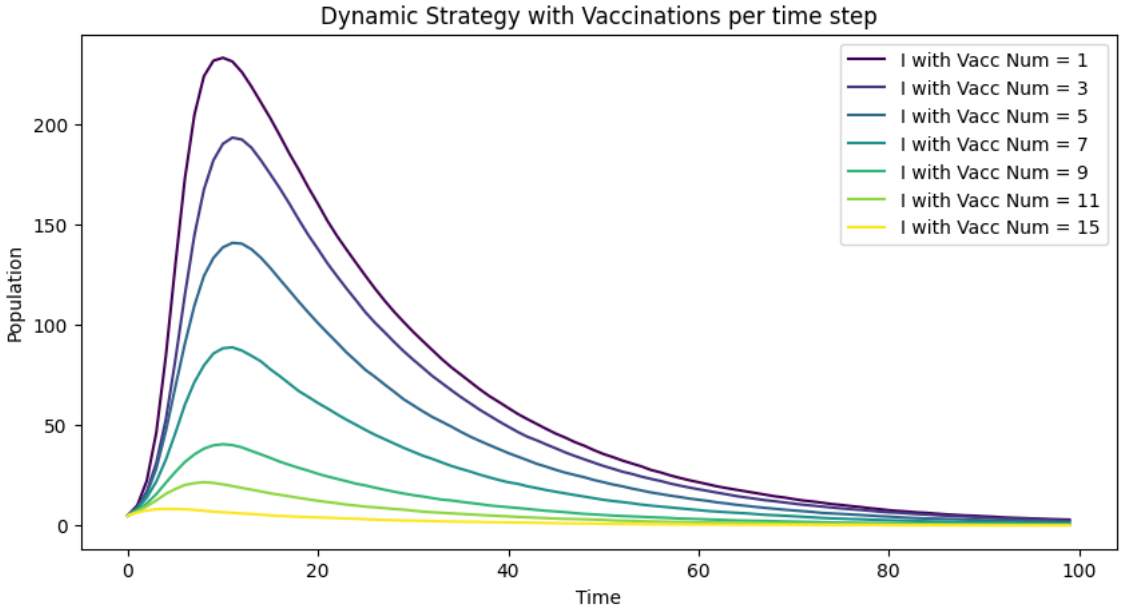


Figure 55: Dynamic Vaccination Strategy with testing on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of vaccinations per time step $= [1, 3, 5, 7, 9, 10, 15]$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

This trend becomes more apparent by plotting the infected node population on its own with a more limited range of vaccinations $= [1, 3, 5, 7, 9, 10, 15]$ in Figure 55, showing how the dynamic strategy mitigates the spread of infection enough that vaccinations per iteration ≥ 15 have effectively stopped infection spread completely.

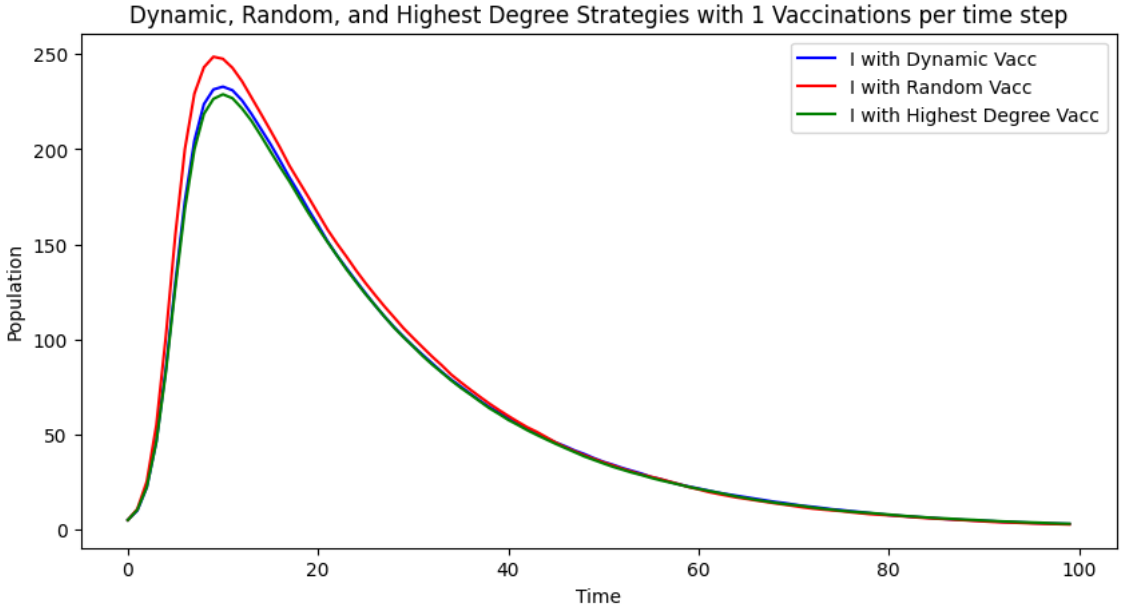


Figure 56: Comparing Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, vaccinations per time step $= 1$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

The apparent improvement of the dampening effect of this dynamic strategy compared to

the Highest Degree no testing strategy may also be further explored through comparison beginning a singular vaccination per time step in Figure 56. This visualization of the infections between the three vaccination strategies shows minor differences between them due to the limited opportunities to minimize infection risk at each time step in a situation where infections on the network grow quickly.

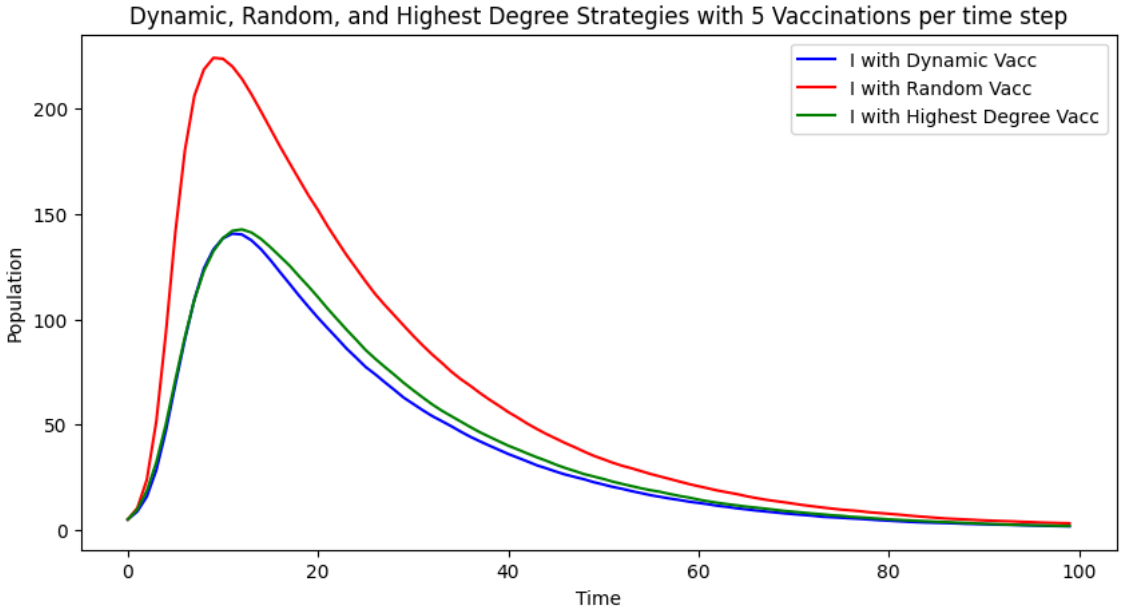
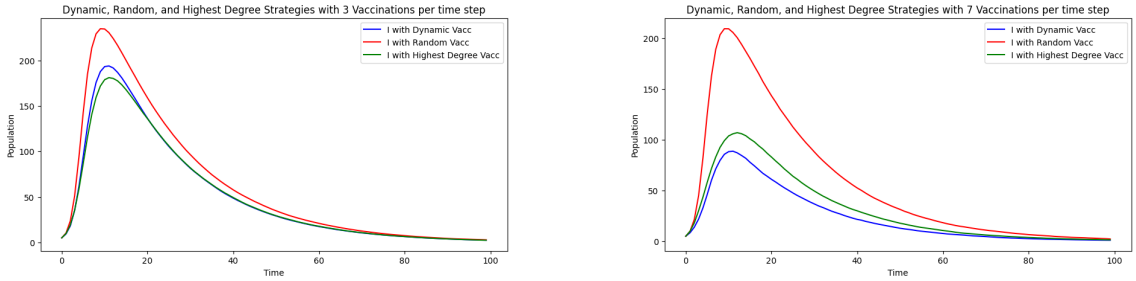


Figure 57: Comparing Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, vaccinations per time step = 5, a maximum number of total testing = 200, and a number of tests per iteration = 200

By choosing a number of vaccinations per iteration = 5 in Figure 57, it's possible to see the stark difference between the null vaccination strategy with completely random vaccine targeting and the two strategies that focus on the highest degree nodes. This difference supports the previous consideration that infection spread depends heavily on infections of high degree nodes and thus the strategy to vaccinate these nodes preemptively effectively reduces the possible options for infections to spread through.



(a) Vaccinations per time step = 3

(b) Vaccinations per time step = 7

Figure 58: Comparing Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a maximum number of total testing = 200, and a number of tests per iteration = 200. Larger Figures and additional β Figures are available in the Appendix B.

What might be noticed in Figure 57 is that both the dynamic strategy and the highest degree no testing strategy are nearly identical in their dampening effect on their respective infected

populations. By plotting infections for the number of vaccination = 3 in Figure 58b and the number of vaccinations = 7 in Figure ?? just above and below the vaccinations = 5, it's possible to see the effectiveness of these two strategies more clearly. When vaccinations per iteration are lower than 5, the number of infections is dampened more by selecting the highest degree nodes without testing compared to a dynamic vaccination strategy with testing.

It's likely that this occurs because the targeted vaccinations of the dynamic strategy focus on high-degree neighbors of high degree infected nodes first, thus causing the limited vaccinations per iteration to be applied to many neighbor nodes, thus distracting testing and vaccination from other possible outbreaks in the network that would be easier to handle with limited resources. This vaccination value of 5 might also be related to the average number of degrees per node of the sociopatterns network as per Figure 44, where the average degree of the network is 6 7 and the vaccination strategies possibly diverge at 5 6, although this is a stretch of correlation more so than quantative evidence.

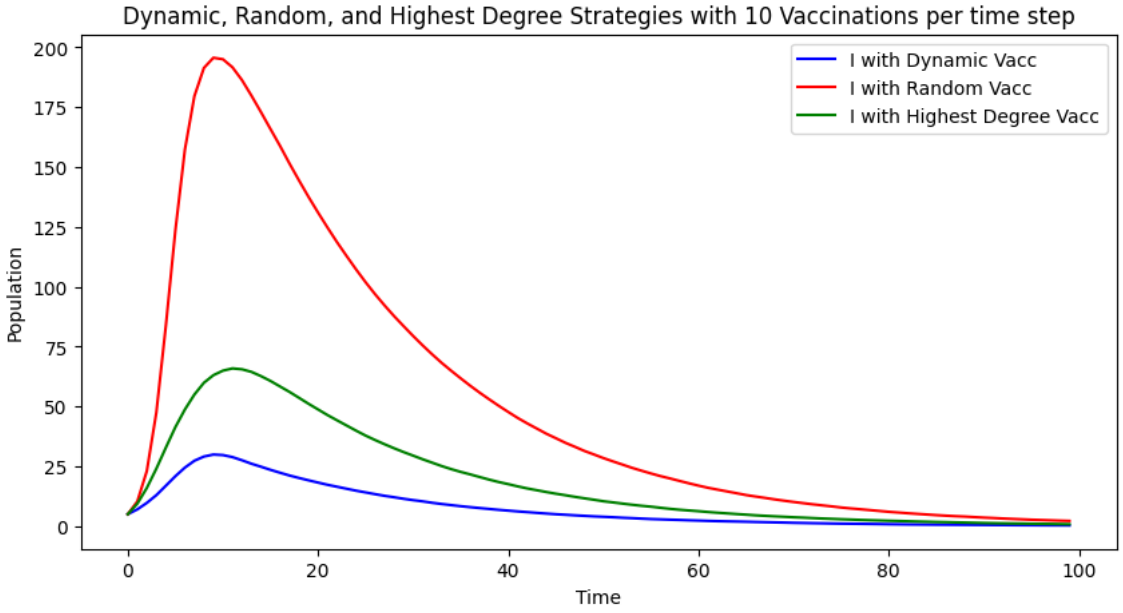


Figure 59: Comparing Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, vaccinations per time step = 10, a maximum number of total testing = 200, and a number of tests per iteration = 200

The relationship of effectiveness for the two strategies reverses after the number of vaccinations per iterations is > 5 in Figures 58b and 59 such that the dampening effect of the dynamic strategy grows considerably as the number of vaccinations available increases. At the peak of 5 vaccinations per time step, 40% of the total nodes in the network are infected at peak, while doubling that to 10 vaccinations sees a peak of only 6% of the total nodes infected, showing a greater increase in the rate of effectiveness of vaccination as it reaches 10 in Figure 60.

Additionally, by finding the difference between the dynamic strategy and the highest degree no testing strategy for each vaccination amount in Figure 61, the comparison between the strategies becomes more apparent. This confirms the findings in relation to the effectiveness of both vaccination strategies wherein the effect flips at 5 vaccines per time step and peaks at 10 vaccinations after a non-linear increase in effectiveness.

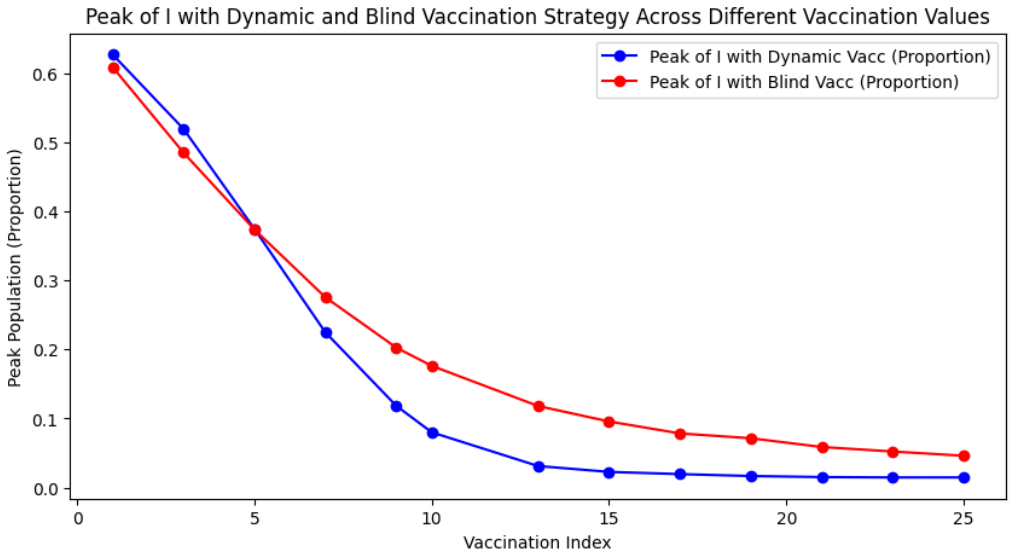


Figure 60: Comparing Normalized Infection peaks for Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of vaccinations per time step $= [1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

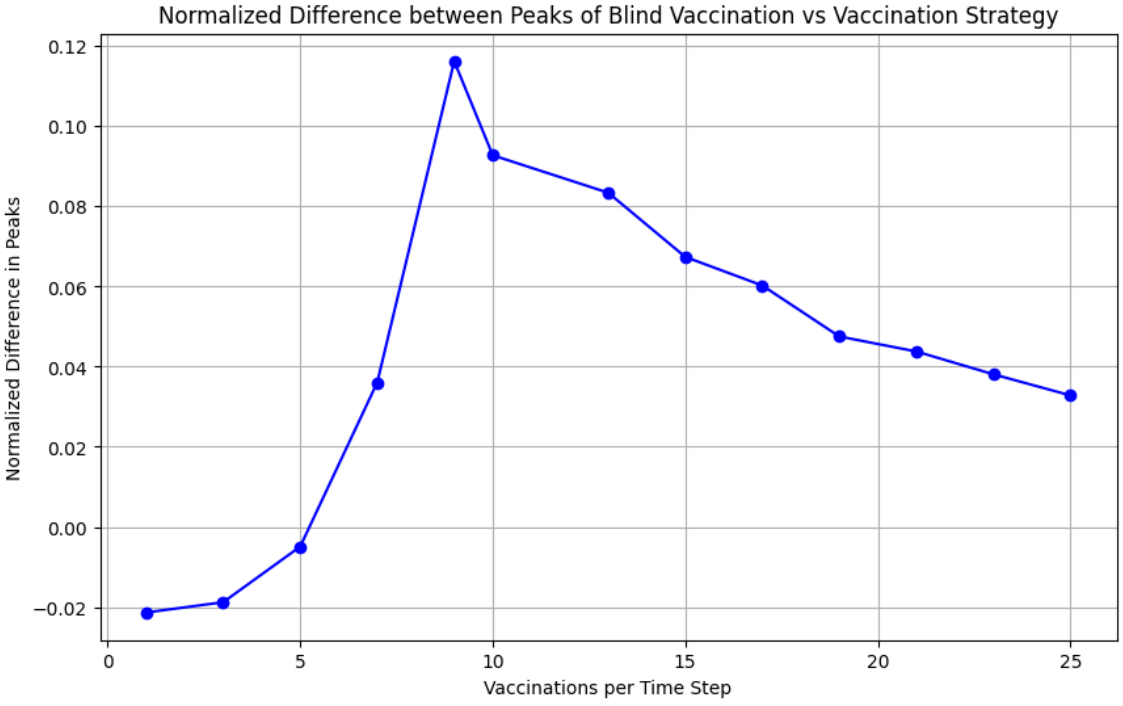


Figure 61: Difference of Infection peaks for Vaccination Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of vaccinations per time step $= [1, 3, 5, 7, 9, 10, 13, 15, 17, 19, 21, 23, 25]$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

To view animated network infection spread videos for different vaccination strategies and parameters, refer to the project drive.

5.8.1 Testing Accuracy

Ultimately, the characteristics of the dynamic strategy that make it effective are also what could be its downfall given the possibility of *False Negatives* which may originate from incorrect testing procedures, noise in the testing data, or early/late timing of test detections. These false negatives are handled in the dynamic strategy by treating infected nodes as susceptible, thus ignoring the step which tests neighbors of those infected nodes and attempting to vaccinate them if vaccines are available. If vaccines are not available, then these infected nodes get a second chance at being tested correctly in the next time step.

Given this situation where-in testing may have an accuracy rate of 0.25, 0.5, 0.75, and 1.0 while vaccinations occur at 5 per time interval for 5 random initial infected nodes, Figure 62 shows a slight difference in dampening effect as the testing accuracy changes.

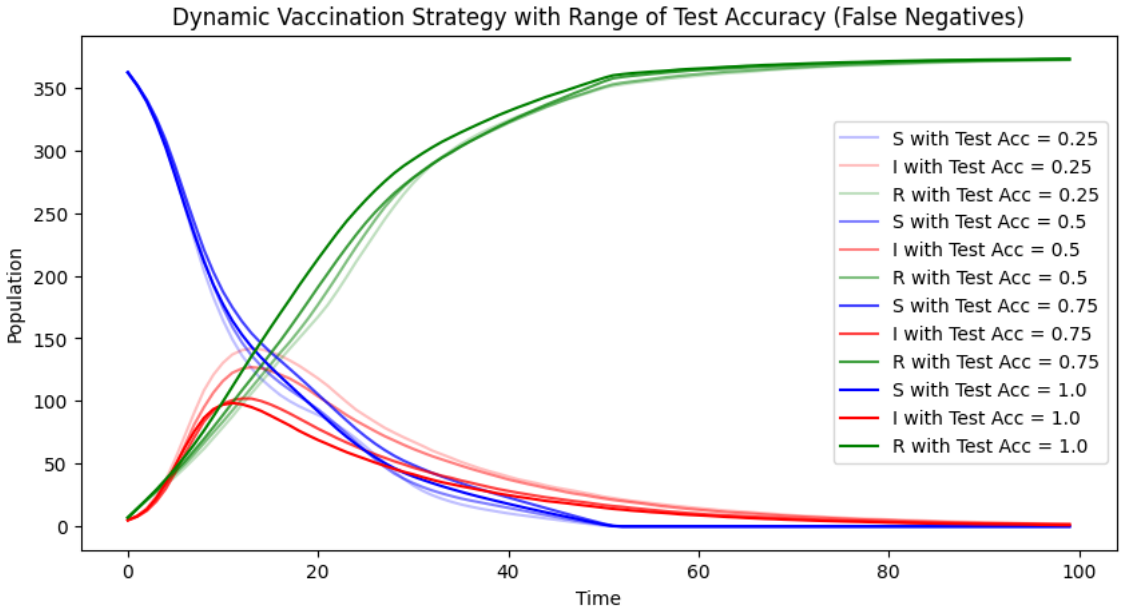


Figure 62: Testing Accuracy of Dynamic Vaccination Strategy on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of testing accuracy $= [0.25, 0.5, 0.75, 1.0]$, vaccinations per time step $= 5$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

This confirms the idea that the inclusion of possible false negatives decreases the effectiveness of the dynamic vaccination and testing strategy, but by comparing these testing accuracies to the null and highest degree, no testing, vaccination strategies in Figure 63, it becomes possible to determine just how damaging these testing accuracy changes are to vaccinations.

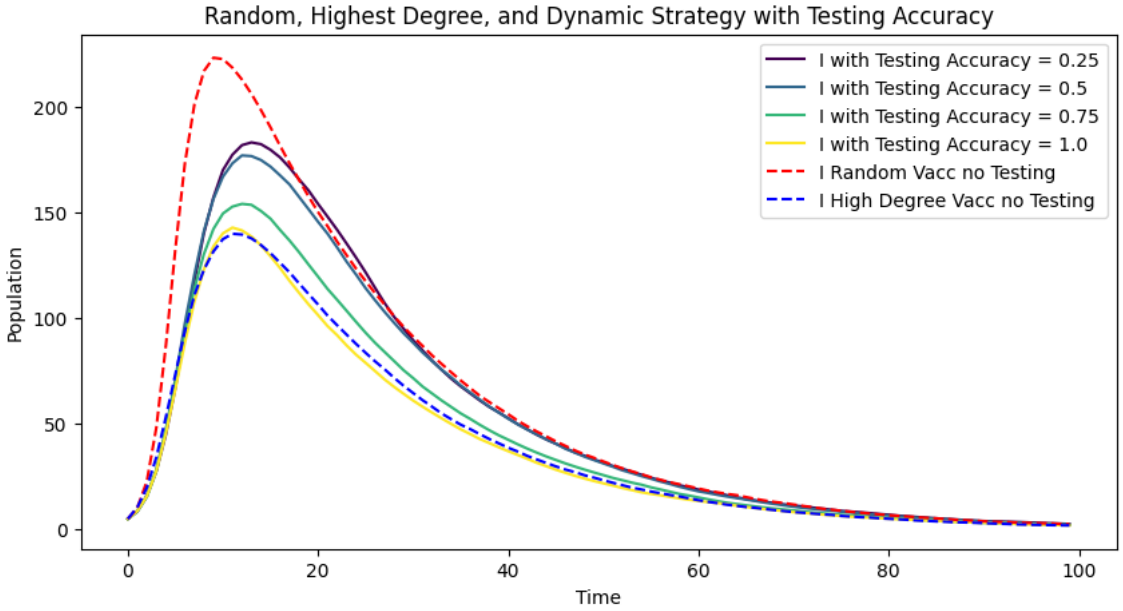


Figure 63: Testing Accuracy of Dynamic Vaccination Strategy vs Null and Blind Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of testing accuracy $= [0.25, 0.5, 0.75, 1.0]$, vaccinations per time step $= 5$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

What becomes immediately apparent when vaccinations per iteration $= 5$ in Figure 63 is that the similarity of the highest degree no testing strategy no longer holds as testing accuracy falls towards the null strategy. This ultimately does not cause the dynamic strategy to be less effective than the null strategy, but finds a network with limited vaccination resources to be more benefited without testing if any false negatives occur at all.

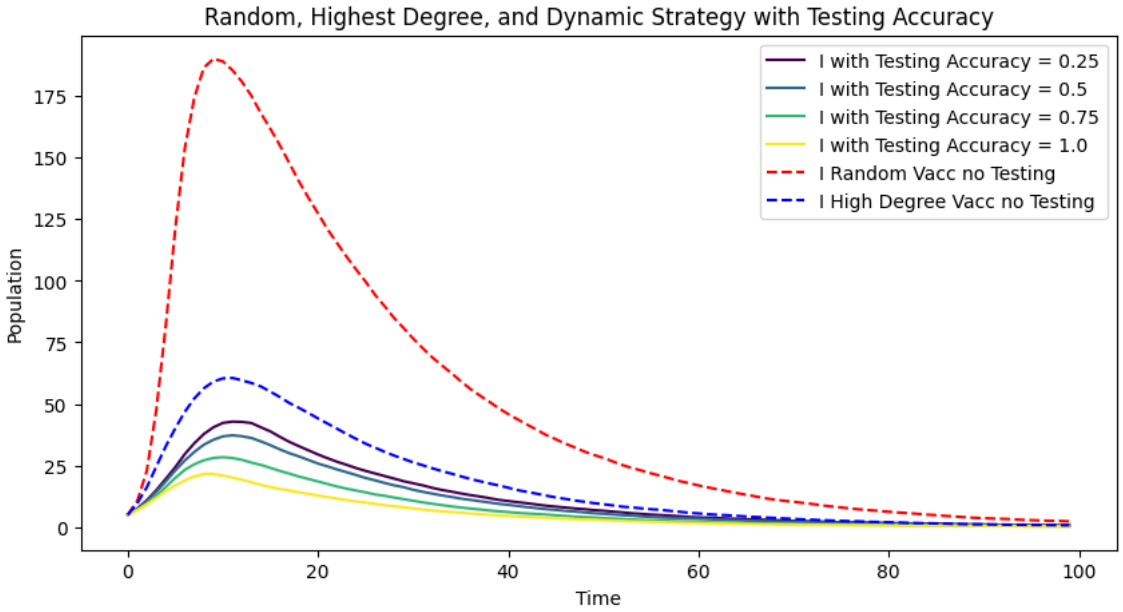


Figure 64: Testing Accuracy of Dynamic Vaccination Strategy vs Null and Blind Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of testing accuracy $= [0.25, 0.5, 0.75, 1.0]$, vaccinations per time step $= 10$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

In a network where vaccination resources are more abundant, the inclusion of false negatives is heavily outweighed by the ability to vaccinate many individuals per time step such that the effectiveness of the dynamic vaccination strategy holds in Figure 64 even with the highest degree no testing option available. By finding the middle ground of the two in Figure 65, the highest degree no testing strategy splits the testing accuracies in two such that ≥ 0.75 testing accuracy sees a more effective dynamic strategy and < 0.75 sees opposite.

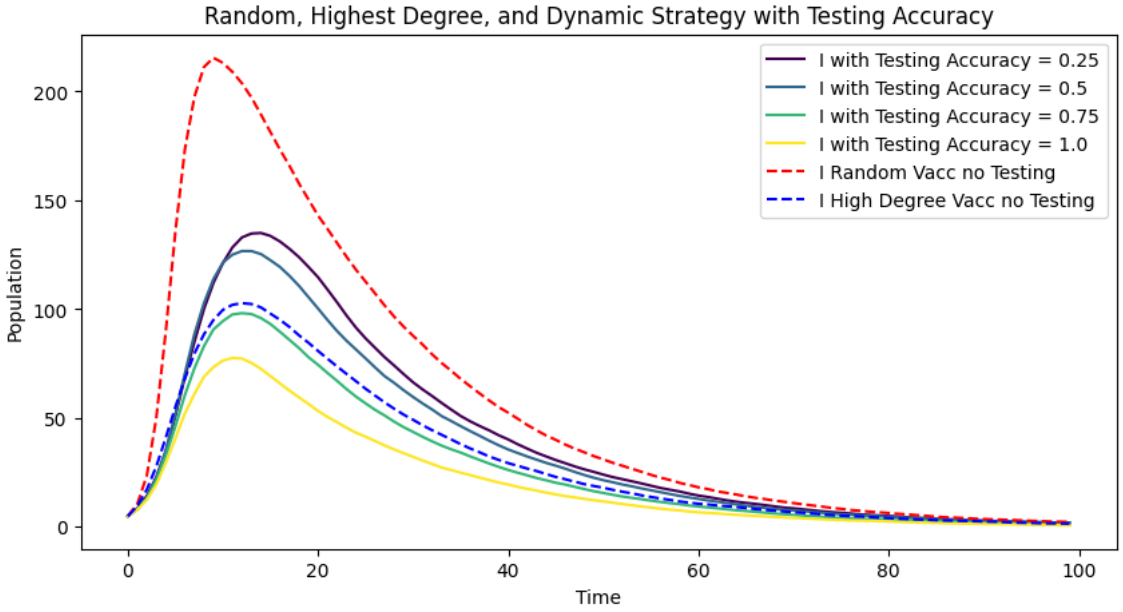


Figure 65: Testing Accuracy of Dynamic Vaccination Strategy vs Null and Blind Strategies on Sociopatterns Network for 500 runs of 5 random infected nodes per run, $\beta = 0.175$, $\gamma = 0.05$, a range of testing accuracy $= [0.25, 0.5, 0.75, 1.0]$, vaccinations per time step $= 7$, a maximum number of total testing $= 200$, and a number of tests per iteration $= 200$

6 Conclusion

To view an epidemic through numbers and models and simulations is to view the world through the infinite possibilities that make it work, that make each interaction and each individual within that system pieces to a larger puzzle. But changing these parameters to the system is only one step to the overall creation of the puzzle because the puzzle can be changed into new forms and shaped into unique models such that deterministic models become but the first step in the process of simulation.

Stochastic models such as those depicted by Gillespie's Direct Algorithm express a discrete-event based model of events according to each events probability of occurring, creating an opportunity to observe key features of stochasticity in systems like those of the SIR with demography. These features include increased variability between many different simulations that sees no two simulations almost ever having the same results; or increased transients that see spikes away from and returns to the deterministic attractor; or harmonic resonance that sees stochastic data following and amplifying deterministic oscillations; or large negative covariance that exceeds that of the deterministic such that even the mean of the stochastic SIR might be effected.

What becomes apparent is that these stochastic features are not just tied linearly to the parameters of the SIR model, rather tending to express dynamic stochastic behaviors that cause spikes of statistical difference based on γ and β. These parameters are such that equivalent Reproductive Rates between different γ and β values are not statistically equivalent and do not express the same behavior stochastically – a large range of β across a value of γ between 0.1 and 0.2 sees spikes of negative covariance and high standard deviation which is not present for other equivalent ranges of γ and β. This spike expresses itself as extinctions on the SIR model as har-

monic resonance pushes infections to zero, a behavior that is minimized as γ decreases and β increases while R_0 remains the same.

Likewise, by modifying the the population size over a range of R_0 it's possible to observe how larger population decreases the variability of the stochastic data, effectively controlling noise and having the added effect of reducing extinction probability. By tracking how population influences extinction proportion over time in relation to R_0, it becomes apparent that some critical community size exists at around 300 to 400 where-in the proportion of extinctions falls to nearly zero for 1000 time steps with the likelihood of this proportion slightly increasing with increased time steps.

By expanding this stochastic framework to spatial modeling with some stochastic elements, it's then possible to express how an infection might spread in an abstracted space of interacting individuals. The creation of networks as abstract spatial models establishes a new set of parameters by which infections must spread into an epidemic such that nodes degrees, clustering, and network diameter become important factors in the transmission between hosts. What also becomes apparent is that the type of network structure defines this transmission behavior even as edges and nodes remain similar.

A Barabási-Albert network is structured as a scale-free network in which a few nodes have most of the connections while the vast majority have very few connections, creating a small number of "hubs" which act as transfer points for the network. These networks feature a lower diameter and minimal clustering due to the lower-skewed degree distribution, causing these networks to be quickly infected by infectious diseases that spread through the centralized "hubs".

The Watts-Strogatz network features a small-world structure that expresses clusters of nodes connected by "shortcuts" between the different "communities". This network structure sees most node degrees centered on the average degree such that the degree distribution remains focused around the singular average, while the average diameter is deceptively high due to the selective 0.1 edge rewire chance, a chance that also happens to cause the average clustering coefficient to be high. These features make the transmission behavior of infectious disease an opposite of scale-free networks, creating longer paths for infections to travel and fewer edge to traverse between clusters of nodes, making this network the most effective at minimizing infection spread.

The Erdős-Rényi network is a random network that distributes it's edges based on some chance to create an edge between a new node and each of the currently existing nodes. This network structure create a binomial degree distribution around the average degree with a low average clustering coefficient and low diameter using similar node and edge characteristics of the other networks. This network structure generally retains a high transmission rate along a range of R_0, possibly due to the influence of nodes with higher degree at the right tail of the degree distribution.

This transmission behavior is partly supported by the inclusion of a Regular network that has a completely uniform degree distribution with equal degree for all nodes, but random edge connections. Given this selection of randomly connected nodes, it is found that mitigation of the infection transmission of this network performs well when R_0 is low and remains more effective than the Erdős-Rényi across a range of R_0.

By implementing a real-world network data set and using it's node number and average degree as basis for other networks, a real system may be observed for how infections might transmit throughout. With a relatively high average clustering coefficient and diameter, a dampening effect is observable on the infection transmission progress that is less effective than the Watts-Strogatz network, but more effective than the rest.

By taking this real-system sociopatterns data, it also becomes possible to apply intervention methods to mitigate infection spread in the form of vaccinations on specific nodes. This means comparing different vaccination strategies to one another based on different parameters of testing and vaccination such that a dampening effect might be observed on the network.

A null strategy of random vaccination on network nodes creates a small mitigation on the transmission of infectious disease within the network, but shows how targeted vaccinations are not very effective without large numbers of vaccinations at each time step or extremely good luck. This strategy acts as effective baseline for other strategies as an vaccine implementation should be at best better than randomly selecting nodes to vaccinate.

As such, by considering the different network structures and their individual characteristics

compared to the transmission rates on each, a new method of vaccination may conclusively be developed that prioritizes nodes in the network of the highest degree. The status of these nodes is unknown during vaccination, but even with possible ineffective vaccinations, the results of see a considerable decrease in peak infections in the network as the number of vaccines per time step increases.

Taking this implementation one step further by adding testing to determine the status of nodes at each step allows for a dynamic vaccination strategy that focuses on testing and vaccinating nodes that are confirmed high-degree neighbors of infected nodes before moving on to the highest degree nodes in the whole network. This strategy seeks to target infected nodes and isolate them by vaccinating their connected neighbors to reduce transmission further into the network.

The dynamic strategy shows apparent improvement over the highest degree no testing strategy when the number of vaccinations per time step greater than 5 such that 15 vaccinations sees en effective extinction of the infection and a vaccination number of 10 sees only 6% of the network at most being infected at peak. The lack of effectiveness when the number of vaccinations is less than 5 is likely caused by vaccinations being distracted by and stuck on neighbors of infected nodes rather than targeting higher risk high degree nodes when there aren't enough vaccines.

This makes the highest degree no testing strategy the more effective strategy when vaccination resources are minimal. This also means that if false negatives are introduced in the testing procedure, there becomes a higher number of vaccinations where it could be more effective than the dynamic strategy. Such situations in which testing accuracy is a concern see false negatives occurring when testing is done, creating situations in the dynamic strategy where vaccinations are wasted or neighbors of infected nodes are not investigated. In this situation, the key to an effective mitigation intervention is once again an increase in the number of vaccinations per iteration proportional to the rate of testing accuracy.

By adjusting the very structure of the simulation, it is possible to model systems beyond the deterministic such that randomness, space, and connection may become variables in the spread of infection. Exploring these models explores an abstracted world by which nodes are individuals and their connections are their relationships, or where-in random extinctions of an infectious disease may be caused by someone's sudden desire to stay in and watch a movie. With this model comes the responsibility find solutions and determine correlations in risk such that all individuals in these abstract simulated worlds might find solace in prediction and implementation.

References

[1] M. J. Keeling P. Rohani, *Modeling Infectious Diseases in Humans and Animals* (Princeton University Press, 2011).

[2] W. O. Kermack A. G. McKendrick, A contribution to the mathematical theory of epidemics, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* **115**, 700 (1927).

[3] G. Rossetti *et al.*, Ndlib: a python library to model and analyze diffusion processes over complex networks, *International Journal of Data Science and Analytics* **5**, 61 (2018).

[4] J. Stehlé *et al.*, Simulation of an seir infectious disease model on the dynamic contact network of conference attendees, *BMC Medicine* **9** (2011).

[5] P. L. Erdos A. Rényi, On random graphs. i., *Publicationes Mathematicae Debrecen* (1959).

[6] A. STEGER N. C. WORMALD, Generating random regular graphs quickly, *Combinatorics, Probability and Computing* **8**, 377–396 (1999).

[7] D. J. Watts S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature* **393**, 440 (1998).

[8] A.-L. Barabási R. Albert, Emergence of scaling in random networks, *Science* **286**, 509 (1999).

[9] A. A. Hagberg, D. A. Schult P. J. Swart, *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught J. Millman, eds. (Pasadena, CA USA, 2008), pp. 11 – 15.

[10] G. Rossetti *et al.*, *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2017), pp. 155–164.

A Appendix A: Python Code

A.1 Libraries

The following Python libraries are required for calculations, plotting, and visualizations of results.

```
import random
from collections import Counter
from functools import partial
import dataclasses

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.cm as cm
from matplotlib.animation import FuncAnimation

import networkx as nx # Network Analysis
import ndlib.models.ModelConfig as mc
import ndlib.models.epidemics as ep

from PIL import Image
from IPython.display import Video, display

from scipy.fft import fft, fftfreq
from scipy.signal import find_peaks
from alive_progress import alive_bar
```

Ensure that *ffmpeg* is installed on your system, as it is required for saving the video. You can install it using package managers like apt, brew, or download it from the official website.

Module/Package	Functionality/Purpose	Example Usage
`random`	Provides functions to generate random numbers and perform random sampling.	`random.randint(1, 10)`
`collections.Counter`	Utility to count hashable objects; useful for counting frequencies of elements in a list.	`Counter([1, 2, 2, 3])`
`functools.partial`	Allows partial application of functions by prefilling some arguments.	`partial(func, arg1=value)`
`dataclasses`	Provides a decorator and functions for creating data classes in Python.	`@dataclass`
`numpy (np)`	Supports large, multi-dimensional arrays and matrices along with a variety of mathematical functions.	`np.array([1, 2, 3])`
`pandas (pd)`	Data manipulation and analysis library, especially for tabular data in DataFrames.	`pd.DataFrame()`
`matplotlib.pyplot (plt)`	Plotting library for creating static, animated, and interactive visualizations.	`plt.plot()`
`matplotlib.cm`	Color map module in matplotlib for coloring plots.	`cm.get_cmap('viridis')`
`matplotlib.animation.FuncAnimation`	Tool for creating animations by repeatedly updating a plot.	`FuncAnimation(fig, update)`
`networkx (nx)`	Library for creating, analyzing, and visualizing complex networks.	`nx.Graph()`
`ndlib.models.ModelConfig (mc)`	Part of NDlib, used for configuring parameters of simulation models.	`mc.Configuration()`
`ndlib.models.epidemics (ep)`	Provides epidemic simulation models for networked systems.	`ep.SIRModel()`
`PIL.Image`	Module for image processing, allowing operations such as image resizing and filtering.	`Image.open('image.jpg')`
`scipy.fft.fft`	Fast Fourier Transform (FFT) function from SciPy, used for spectral analysis.	`fft(data)`
`scipy.fft.fftfreq`	Computes the sample frequencies for FFT.	`fftfreq(len(data))`
`scipy.signal.find_peaks`	Finds peaks in data; useful for identifying local maxima in signals.	`find_peaks(data)`
`alive_progress.alive_bar`	Progress bar utility for tracking iterations in loops interactively.	`with alive_bar(total):`
`IPython.display (Video, display)`	Used for displaying video content and other rich media outputs in Jupyter notebooks.	`display(Video('video.mp4'))`

Table 1: Python Imports and their Descriptions

A.2 Gillespie's Direct Algorithm

The following code, imported from `gillespiesDirect.py` (Listing **??**), implements Gillespie's Algorithm. This stochastic simulation algorithm is often applied in modeling systems where random reaction events determine state changes over time.

```
1  def gillespiesAlgorithm(self):
2          events = self.calcEvents()
3          rateTotal = sum([i.rate for i in events])
4
5          deltaT = -1/rateTotal * np.log(np.random.rand())
6
7          probabilities = np.random.rand() * rateTotal
8
9          upperRate = 0
10
11         for i in events:
12             lowerRate = upperRate
13             upperRate += i.rate
14
15             if lowerRate < probabilities <= upperRate:
16                 return i, deltaT
```

1. **Calculate Event Rates (lines 2-3):** The function begins by calling `calcEvents()` to obtain a list of events, each with a `rate` attribute. The variable `rateTotal` is calculated as the sum of all event rates.

2. **Determine Time Increment (line 5):** The time until the next event, `deltaT`, is drawn from an exponential distribution with parameter rateTotal. This is calculated as:

$$\text{deltaT} = -\frac{1}{\text{rateTotal}} \ln(\text{random number})$$

where the random number is generated by `np.random.rand()`.

3. **Select Event Based on Probability (lines 7-15):** A random probability threshold, `probabilities`, is generated and scaled by `rateTotal`. The function iterates through each event, checking cumulative rate bounds (`lowerRate` and `upperRate`) to determine which event satisfies the condition:

$$\text{lowerRate} < \text{probabilities} \leq \text{upperRate}$$

The selected event is then returned along with the calculated time increment, `deltaT`.

A.3 Dynamic Vaccination Strategy

The `simulateVaccinationStrategy` function simulates a targeted vaccination approach on a network to control infection spread over a specified time period. The function prioritizes testing and vaccinating nodes based on their proximity to infected nodes, using a probabilistic testing success rate and a limited number of tests and vaccines per iteration.

```
1  def simulateVaccinationStrategy(infectedNetwork,
2                                  network,
3                                  timeSpan,
4                                  testMax,
5                                  testIterationMax,
6                                  lowPriorityTestingNodes,
7                                  vaccineNumPerIter,
8                                  testingSuccessRate=1.0):
9      allIterations = []
10     highPriorityNeighborsOfInfected = {}
11
12     totalTestsCompleted = 0
13
14     for t in timeSpan:
15         currentVaccCompleted = 0
16         nonNeighborsSusceptible = {}
17
18         if totalTestsCompleted <= testMax:
19
20             for j in range(testIterationMax):
21                 testFailure = random.random() > testingSuccessRate
22
23                 if highPriorityNeighborsOfInfected:
24                     testNodeKey =
                       ↪  getKeyForLargestValueDict(highPriorityNeighborsOfInfected)
25                     currentTestNodeStatus = infectedNetwork.status[testNodeKey]
26
27                     removeNodeFromTestStacks(lowPriorityTestingNodes,
28                                              highPriorityNeighborsOfInfected,
29                                              nonNeighborsSusceptible,
30                                              testNodeKey)
31
32                     if currentTestNodeStatus == 0:
33                         if currentVaccCompleted < vaccineNumPerIter:
```

```
                            currentVaccCompleted += 1
                            infectedNetwork.status[testNodeKey] = 2
                        else:
                            highPriorityNeighborsOfInfected[testNodeKey] =
                            ↪ network.degree(testNodeKey)
                            break
                    elif currentTestNodeStatus == 1:
                        if testFailure:
                            if currentVaccCompleted < vaccineNumPerIter:
                                currentVaccCompleted += 1
                            else:
                                highPriorityNeighborsOfInfected[testNodeKey] =
                                ↪ network.degree(testNodeKey)
                                break
                        else:
                            for n in network.neighbors(testNodeKey):
                                if
                                ↪ searchForNodesInStacks(lowPriorityTestingNodes,
                                                         ↪ highPriorityNeighborsOfInfected,
                                                        nonNeighborsSusceptible,
                                                        n):
                                    highPriorityNeighborsOfInfected[n] =
                                    ↪ network.degree(n)
                elif lowPriorityTestingNodes:
                    testNodeKey =
                    ↪ getKeyForLargestValueDict(lowPriorityTestingNodes)
                    currentTestNodeStatus = infectedNetwork.status[testNodeKey]

                    removeNodeFromTestStacks(lowPriorityTestingNodes,
                                             highPriorityNeighborsOfInfected,
                                             nonNeighborsSusceptible,
                                             testNodeKey)

                    if currentTestNodeStatus == 0 or (currentTestNodeStatus == 1
                    ↪ and testFailure):
                        nonNeighborsSusceptible[testNodeKey] =
                        ↪ network.degree(testNodeKey)

                    elif currentTestNodeStatus == 1:
                        for n in network.neighbors(testNodeKey):
                            if searchForNodesInStacks(lowPriorityTestingNodes,
                                                      ↪ highPriorityNeighborsOfInfected,
                                                     nonNeighborsSusceptible,
                                                     n):
                                highPriorityNeighborsOfInfected[n] =
                                ↪ network.degree(n)

        while currentVaccCompleted < vaccineNumPerIter:
            if nonNeighborsSusceptible:
                currentVaccCompleted += 1
                suscNode = getKeyForLargestValueDict(nonNeighborsSusceptible)
                infectedNetwork.status[suscNode] = 2
                removeNodeFromTestStacks(lowPriorityTestingNodes,
                                         highPriorityNeighborsOfInfected,
```

```
80                                      nonNeighborsSusceptible,
81                                      suscNode)
82              else:
83                  break
84
85          for n in nonNeighborsSusceptible.keys():
86              lowPriorityTestingNodes[n] = nonNeighborsSusceptible[n]
87
88          while currentVaccCompleted < vaccineNumPerIter:
89              if highPriorityNeighborsOfInfected:
90                  blindVaccNode =
                    ↪ getKeyForLargestValueDict(highPriorityNeighborsOfInfected)
91              elif lowPriorityTestingNodes:
92                  blindVaccNode =
                    ↪ getKeyForLargestValueDict(lowPriorityTestingNodes)
93              else:
94                  break
95              currentVaccCompleted += 1
96              currentTestNodeStatus = infectedNetwork.status[blindVaccNode]
97              removeNodeFromTestStacks(lowPriorityTestingNodes,
98                                       highPriorityNeighborsOfInfected,
99                                       nonNeighborsSusceptible,
100                                      blindVaccNode)
101             if currentTestNodeStatus == 0:
102                 infectedNetwork.status[blindVaccNode] = 2
103
104         iteration = infectedNetwork.iteration()
105         iteration['status'] = infectedNetwork.status
106         allIterations.append(iteration)
107
108     return allIterations
```

Function Parameters

- **infectedNetwork**: Represents the current infection status of nodes in the network.
- **network**: The underlying network structure with nodes and edges representing connections.
- **timeSpan**: The time steps over which the simulation runs.
- **testMax**: Maximum number of tests allowed throughout the entire simulation.
- **testIterationMax**: Maximum number of tests allowed in each iteration.
- **lowPriorityTestingNodes**: Nodes with lower priority for testing, initialized before simulation.
- **vaccineNumPerIter**: Maximum number of vaccines available per iteration.
- **testingSuccessRate** (default = 1.0): Probability that a test will correctly identify an infected node.

Function Workflow

1. **Initialization (lines 9-16)**: Initializes variables for storing the simulation results across iterations, sets up high-priority nodes for vaccination, and tracks the total tests conducted.
2. **Main Loop (over timeSpan, line 14)**: For each time step:

- **Vaccination and Testing Counts (lines 15-16):** Resets vaccination and non-neighbor susceptible counts for the current time step.
- **Testing Phase (lines 18-71):** Tests nodes up to `testIterationMax` times, prioritizing neighbors of infected nodes. Depending on `testingSuccessRate`, successful tests lead to vaccinated or marked nodes, while failed tests mark nodes as lower-priority susceptible.
  - **Test neighbors of Infected (lines 23-30):** Test the neighbors of infected nodes starting with the highest degree nodes
  - **Susceptible Neighbor of Infected Node (lines 32-38):** If testing succeeds, vaccinate highest degree susceptible nodes until the `vaccineNumPerIter`, then interrupt testing to start the next time step.
  - **Infected Neighbor of infected node False Negative(lines 39-44):** If testing fails based on the specified `testingSuccessRate`, vaccinate if vaccines are available to no effect
  - **Infected Neighbor of infected node success(lines 45-52):** Add neighbors of the new infected node to the high priority testing queue.
  - **Test Highest Degree Nodes (lines 53-60):** If no neighbors of infected need to be tested, begin testing the highest degree nodes in the network
  - **Susceptible Node or False Negative (lines 62-63):** Add the susceptible node to a low-priority vaccination stack `nonNeighborsSusceptible` to be vaccinated only if no neighbors of infected need to vaccinated at the end of the testing iterations
  - **Infected Node (lines 65-71):** Add neighbors of the new infected node to the high priority testing queue.
- **Non-Neighbor Susceptibles Vaccination Phase (lines 73-83):** Vaccinates nodes with highest priority in the stack `nonNeighborsSusceptible` until reaching the `vaccineNumPerIter` limit.
- **Reset testing pool (lines 85-86):** Nodes that were susceptible in this time step that were not vaccinated are assumed to be unknown in the next time step and added back to the testing pool.
- **Blind Vaccination Phase (lines 88-102):** If testing is exhausted, blindly vaccinate nodes, making susceptible nodes recovered and keeping infected and recovered nodes the same.

3. **Iteration Logging (lines 104-106)**: At each time step, stores the network status and vaccination progress for later analysis.

Return Value

The function returns a list of dictionaries, each representing the network status and vaccination progress for a single iteration, allowing detailed analysis of the epidemic's spread and vaccination impact.

A.3.1 Possible Modifications to the Dynamic Vaccination Strategy

A modified version of this dynamic vaccination strategy could see the implementation of a smaller-than-maximum tests-per-iteration value and a removal of the break in the time step loop after the vaccination maximum is reached. This would effectively allow testing to continue up to a certain point after vaccinations in the time step are finished in an attempt to find additional infected nodes earlier. Further experimentation may be warranted to determine the effectiveness of the different dynamic strategy implementations for the most effective intervention method.

A.4 Null Vaccination

The `simulateNullVaccinationSIR` function simulates a random (null) vaccination strategy within a Susceptible-Infected-Recovered (SIR) model. In this approach, a fixed number of vac-

cines are randomly applied to susceptible individuals each iteration, regardless of proximity to infected nodes. This model can serve as a baseline for comparing targeted vaccination strategies.

```
1  def simulateNullVaccinationSIR(infectedNetwork,
2                                 timeSpan,
3                                 vaccineNum):
4      allIterations = []
5
6      vaccinationArray = list(infectedNetwork.status.keys())
7      random.shuffle(vaccinationArray)
8
9      for t in timeSpan:
10         iterationVaccinations = {}
11
12         for vacc in range(vaccineNum):
13             if len(vaccinationArray) > 0:
14                 vaccCandidate = vaccinationArray.pop()
15
16                 if infectedNetwork.status[vaccCandidate] == 0:
17                     infectedNetwork.status[vaccCandidate] = 2
18                     iterationVaccinations[vaccCandidate] = 2
19         for vacc in iterationVaccinations:
20             infectedNetwork.status[vacc] = iterationVaccinations[vacc]
21
22         iteration = infectedNetwork.iteration()
23
24         iteration['status'] = infectedNetwork.status
25
26         allIterations.append(iteration)
27
28     return allIterations
```

Function Parameters

- **infectedNetwork**: The network with the current infection status of nodes, where each node is either susceptible, infected, or recovered.
- **timeSpan**: A sequence of time steps for which the simulation will be run.
- **vaccineNum**: The number of vaccines available per iteration, applied randomly to susceptible nodes.

Function Workflow

1. **Initialization (lines 4-7)**: Sets up an empty list, `allIterations`, to store each iteration's results. A `vaccinationArray` is created, which contains all node keys from `infectedNetwork`. The array is shuffled to ensure random selection of nodes for vaccination.
2. **Main Loop (over timeSpan, line 9)**: For each time step:
   - **Vaccination Selection (lines 10-18):** - A dictionary, `iterationVaccinations`, is created to store the vaccination status for this iteration. - For each vaccine available (`vaccineNum`), a node is selected randomly from `vaccinationArray` and vaccinated if it is susceptible (status = 0). - The node's status is updated to vaccinated (status = 2).
   - **Updating Network Status (lines 19-20):** - After vaccination, `infectedNetwork.status` is updated with the new vaccinated nodes for this iteration.

3. **Iteration Logging (lines 22-26)**: Records the network status at each time step by storing it in `allIterations`. Each entry in `allIterations` represents the infection and vaccination status for a specific time step, enabling further analysis of infection spread and vaccination effects.

Return Value

The function returns `allIterations`, a list of dictionaries with each dictionary capturing the infection and vaccination statuses for each iteration, representing the time progression of the epidemic under random vaccination conditions.

A.5 Highest Degree Vaccination Strategy

The `highestDegreeVaccinationStrategy` function simulates a targeted vaccination approach that prioritizes nodes with the highest degree (number of connections) in a network. This strategy assumes that vaccinating high-degree nodes may be more effective in controlling the spread of infection due to their influence in connecting other nodes.

```
1  def highestDegreeVaccinationStrategy(infectedNetwork,
2                                       timeSpan,
3                                       testingStack,
4                                       vaccineNumPerIter):
5      allIterations = []
6
7      for t in timeSpan:
8          currentVaccCompleted = 0
9
10         while currentVaccCompleted < vaccineNumPerIter:
11             if testingStack:
12                 testNodeKey = getKeyForLargestValueDict(testingStack)
13                 currentTestNodeStatus = infectedNetwork.status[testNodeKey]
14
15                 del testingStack[testNodeKey]
16
17                 if currentTestNodeStatus == 0:
18                     currentVaccCompleted += 1
19                     infectedNetwork.status[testNodeKey] = 2
20             else:
21                 break
22
23         iteration = infectedNetwork.iteration()
24
25         iteration['status'] = infectedNetwork.status
26
27         allIterations.append(iteration)
28
29     return allIterations
```

Function Parameters

- **infectedNetwork**: The network structure with nodes and their current infection status.
- **timeSpan**: The number of time steps for which the vaccination strategy simulation is executed.
- **testingStack**: A dictionary or stack containing nodes ordered by their connectivity (degree), allowing the function to prioritize nodes with the highest degree.

- **vaccineNumPerIter**: The maximum number of vaccines that can be administered per iteration.

Function Workflow

1. **Initialization (lines 5)**: Initializes an empty list, `allIterations`, to store the results of each time step, including vaccination status changes.
2. **Main Loop (over timeSpan, line 7)**: For each time step in `timeSpan`:
    - **Vaccination Phase (lines 11-19)**: - **High-Degree Selection (lines 12-13):** Vaccination targets are selected by calling `getKeyForLargestValueDict(testingStack)` to obtain the highest-degree node. - **Vaccination Application (lines 17-19):** If the selected node is susceptible (status = 0), it is vaccinated (status updated to 2), and the vaccine count `currentVaccCompleted` is incremented. - If the testing stack is empty, the function exits the vaccination phase.
3. **Iteration Logging (lines 23-27)**: After each time step, the current infection and vaccination status of nodes are recorded in `allIterations`. This provides a snapshot of the network's state after each iteration, tracking the impact of vaccinating high-degree nodes over time.

Return Value

The function returns `allIterations`, a list containing the network's infection and vaccination status for each time step, which enables analysis of the vaccination strategy's effects on infection spread within the network.

B Appendix B: Figures

B.1 Stochastic Model with Range of β and $\gamma = 0.05$

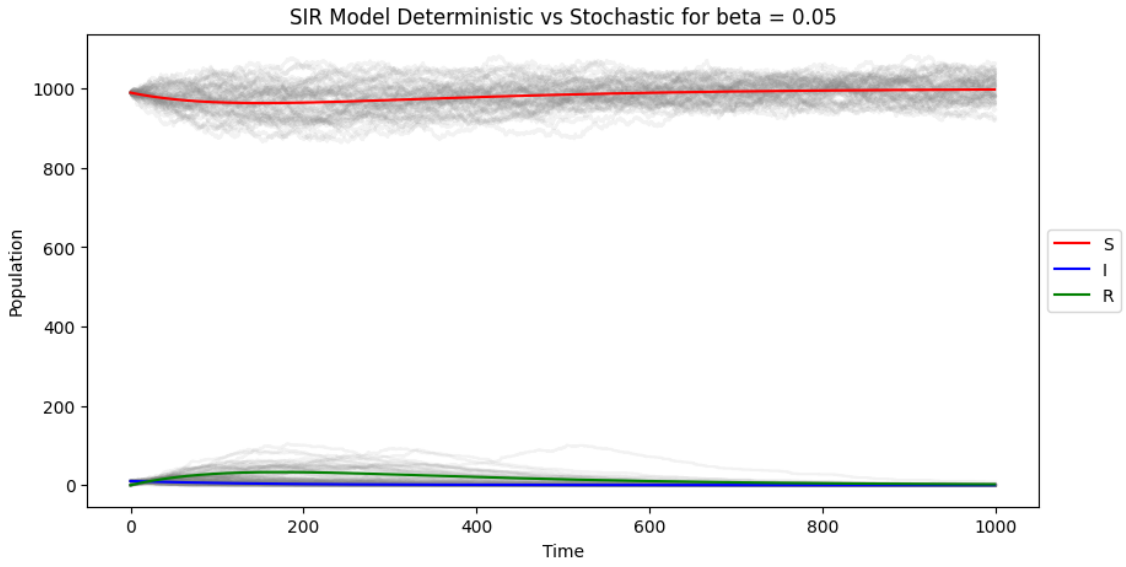


Figure 66

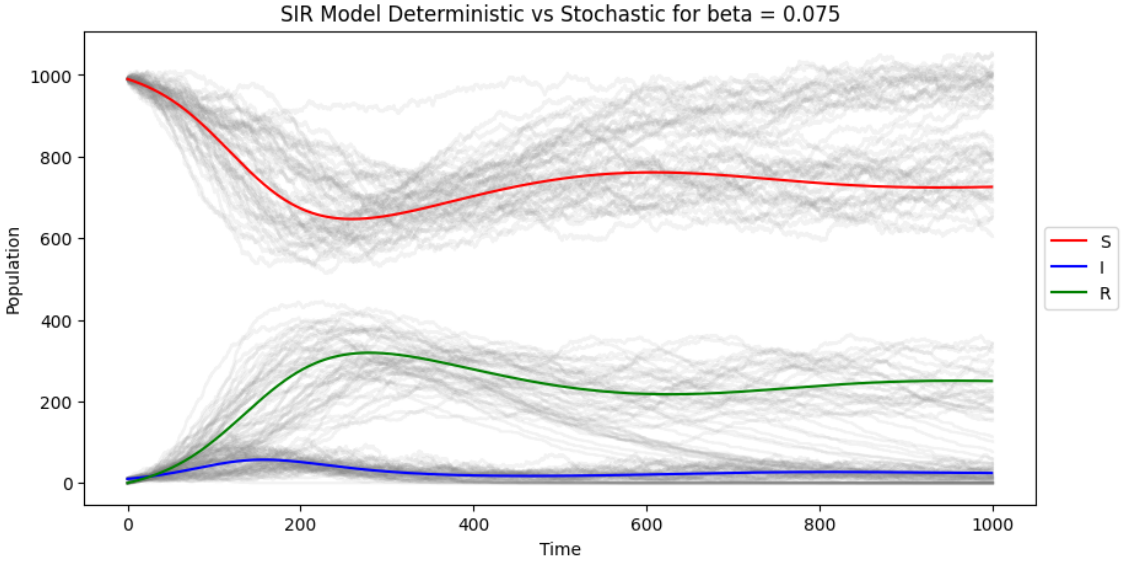


Figure 67

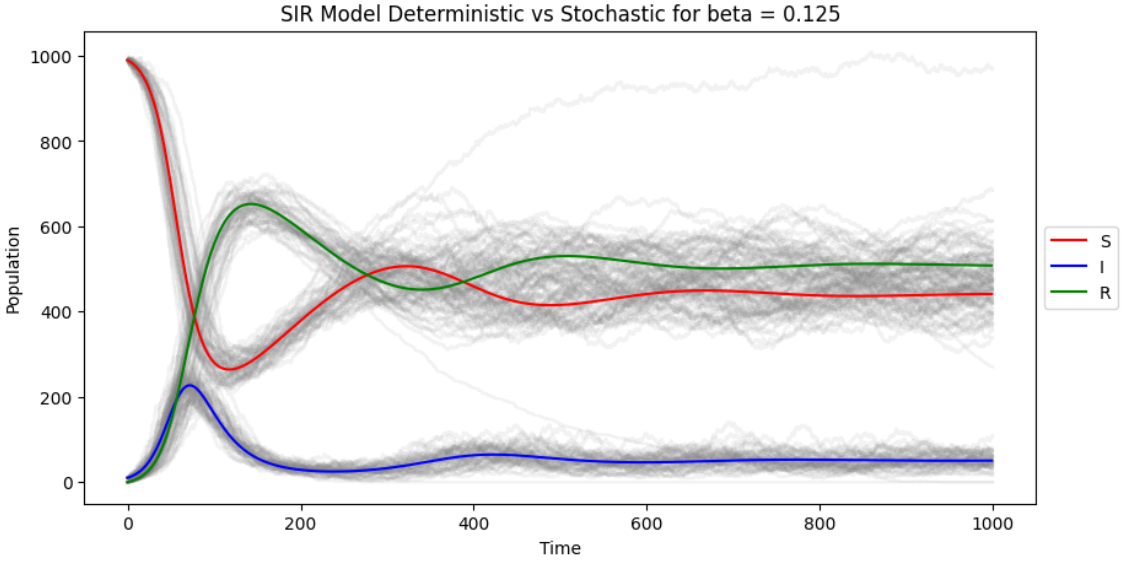


Figure 68

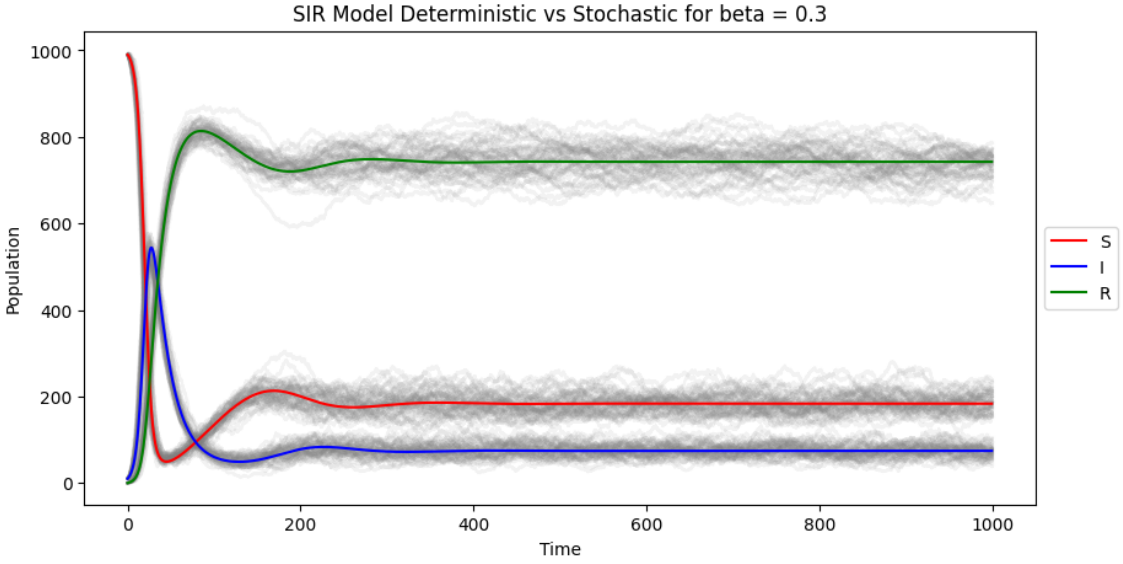


Figure 69

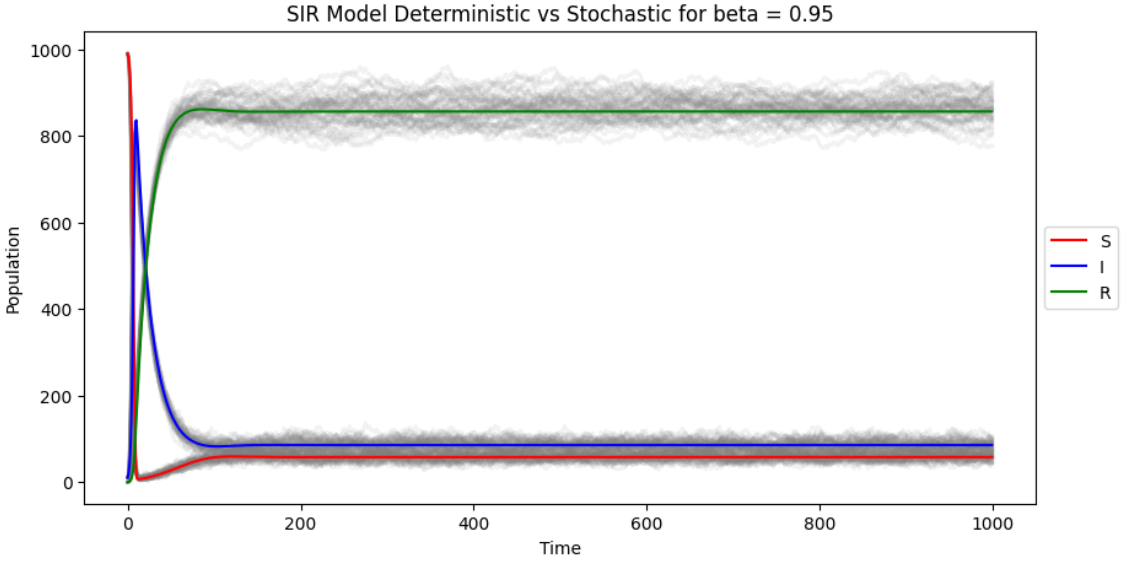


Figure 70

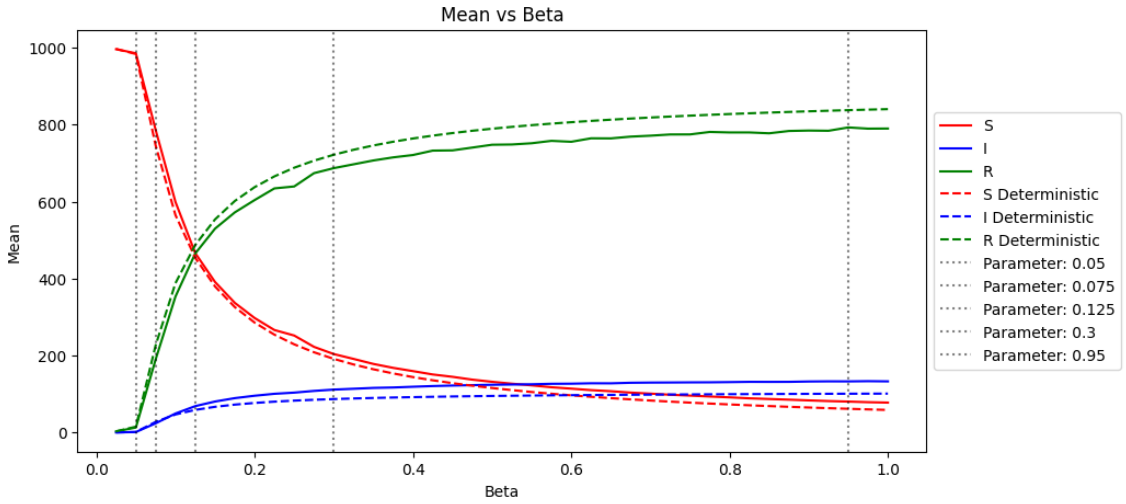


Figure 71

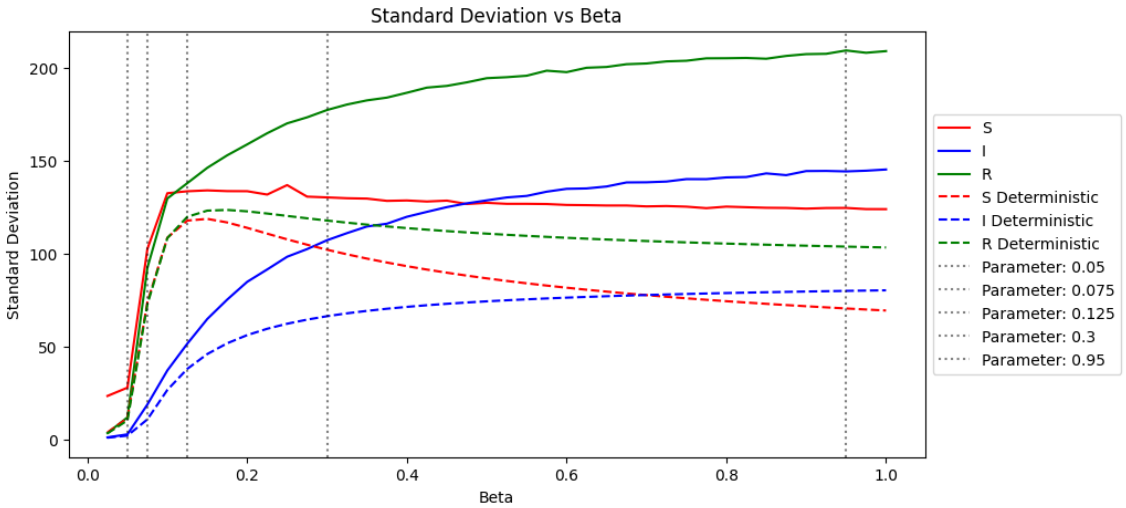


Figure 72

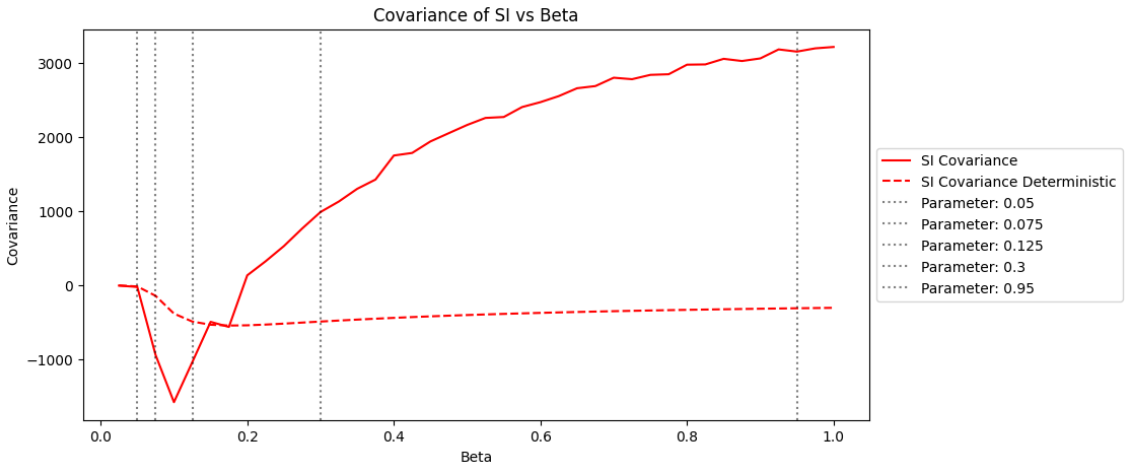


Figure 73

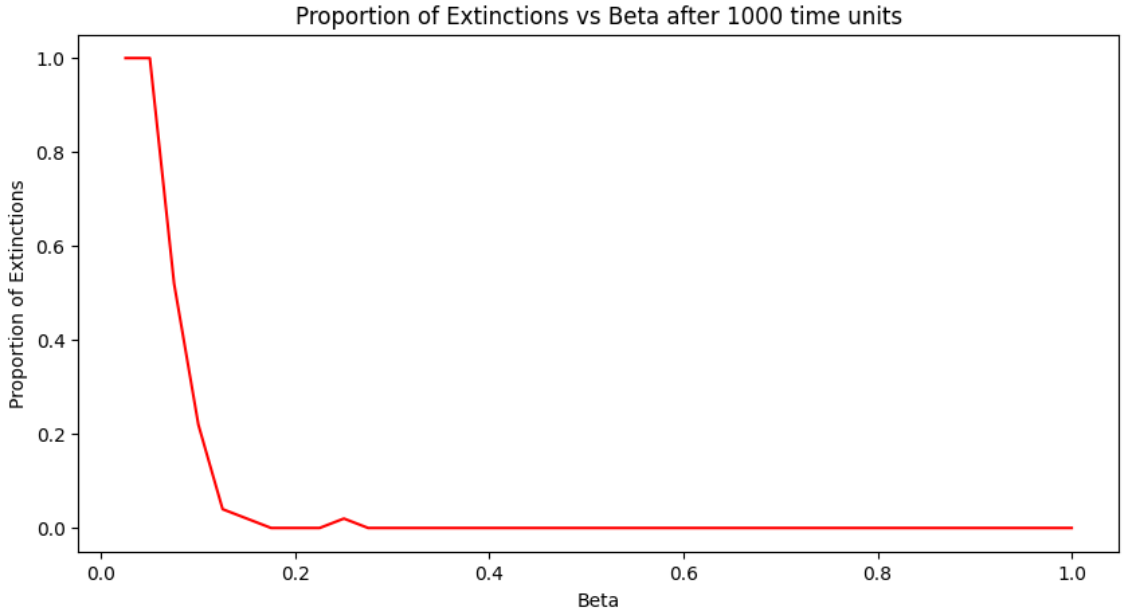


Figure 74

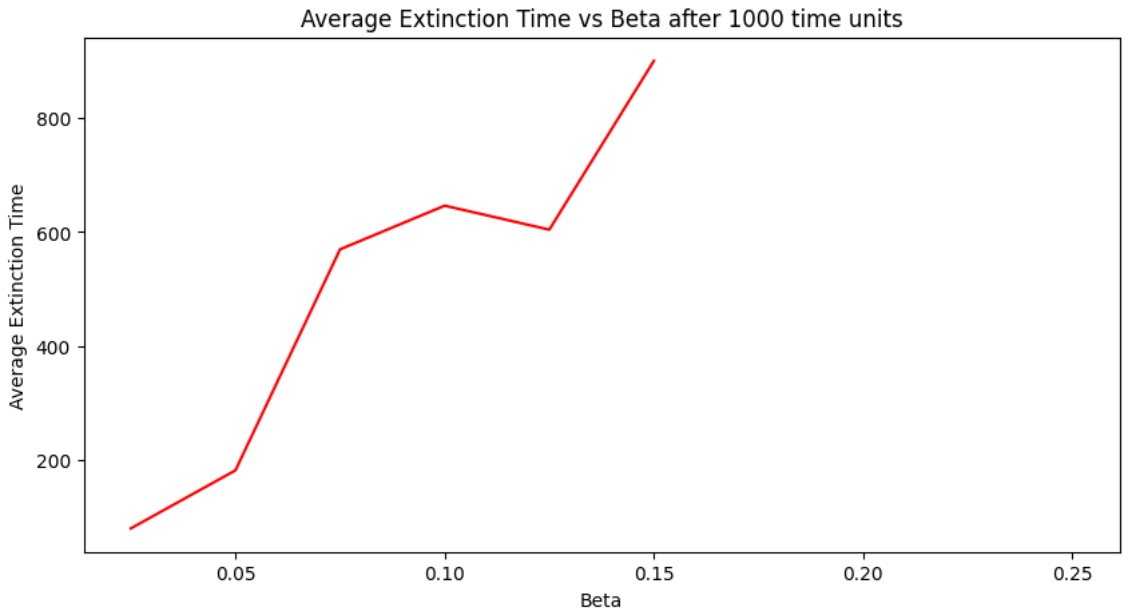


Figure 75

B.2 Stochastic Model with Range of γ and $\beta = 0.25$

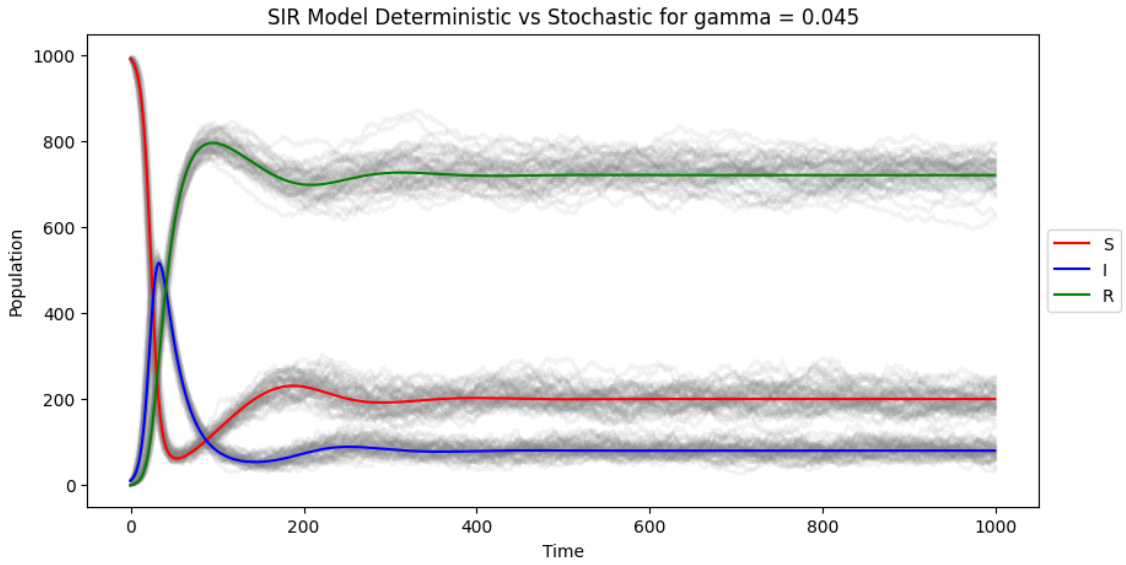


Figure 76

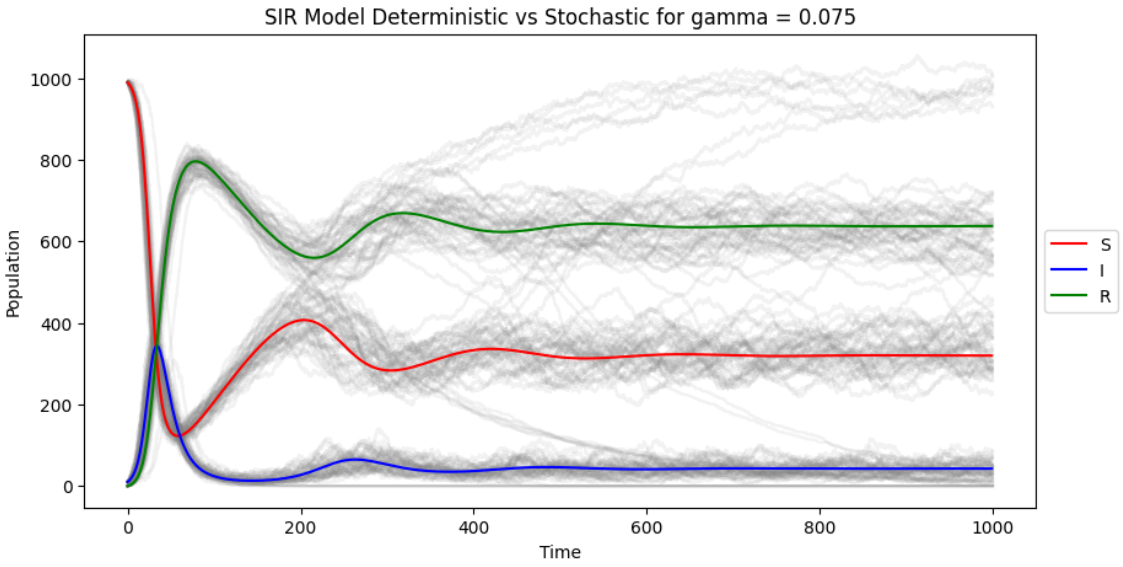


Figure 77

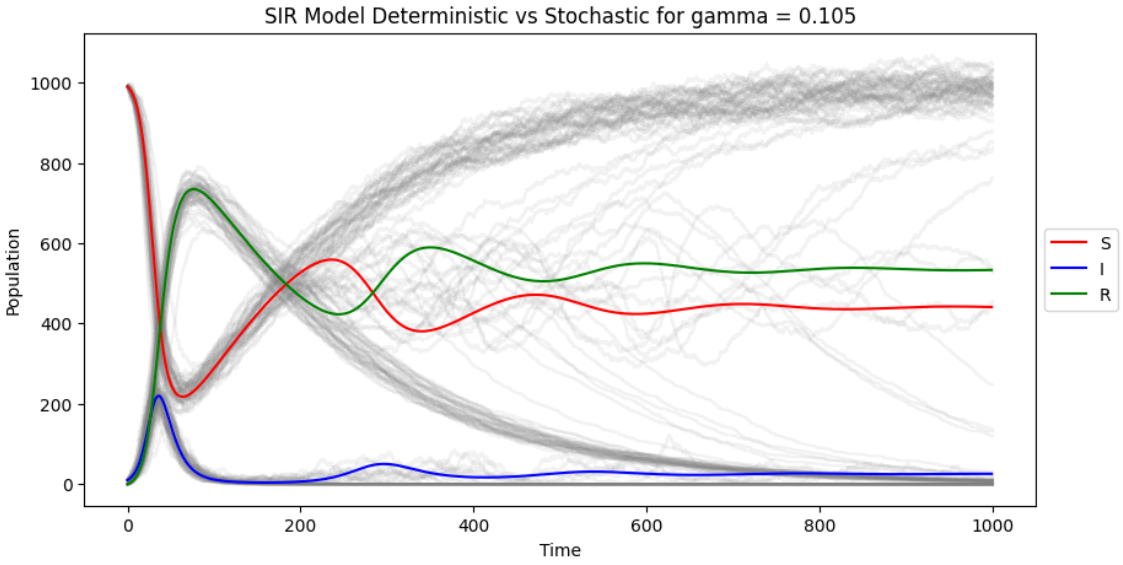


Figure 78

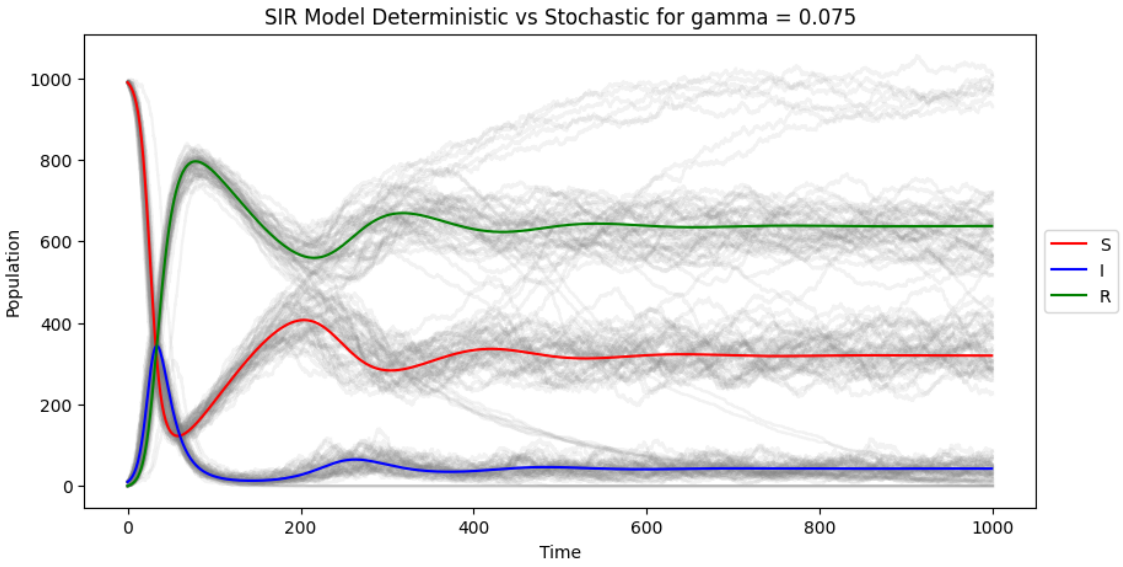


Figure 79

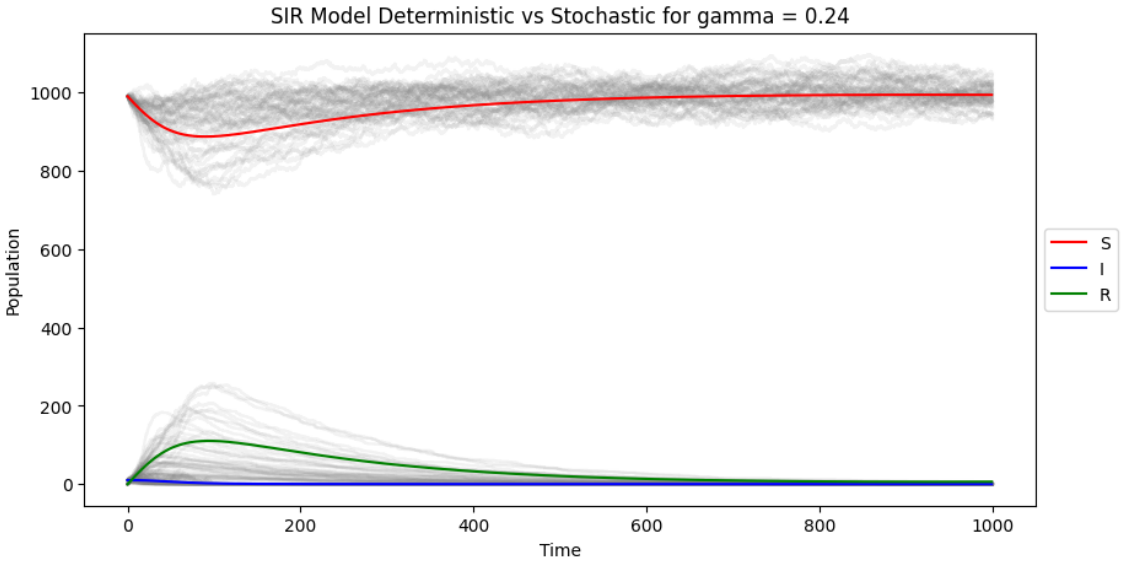


Figure 80

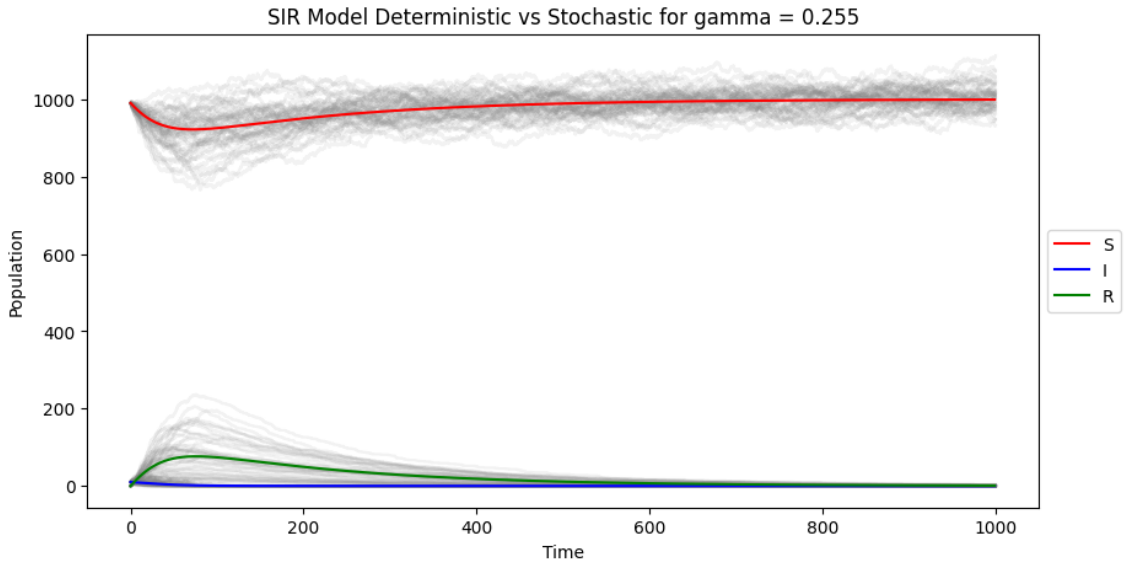


Figure 81

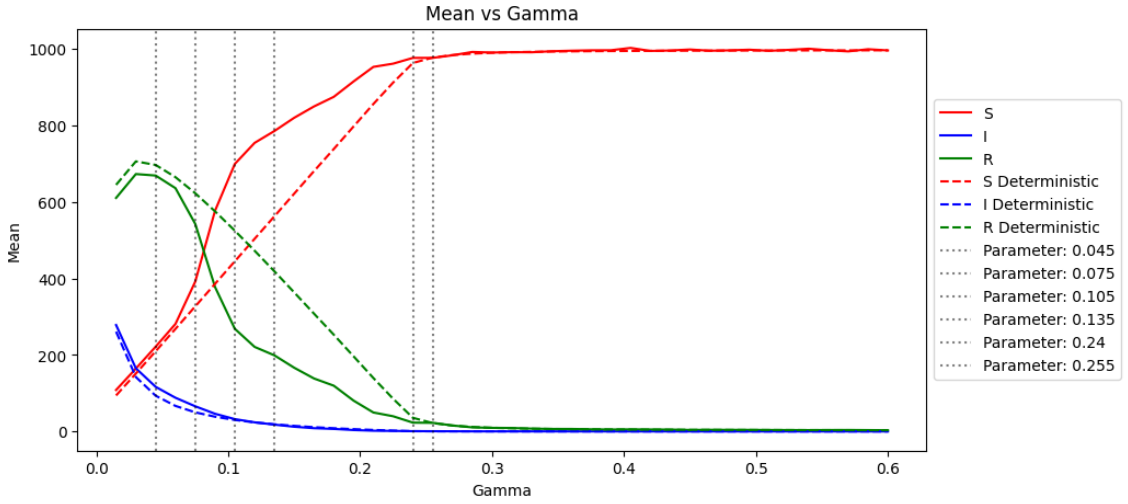


Figure 82

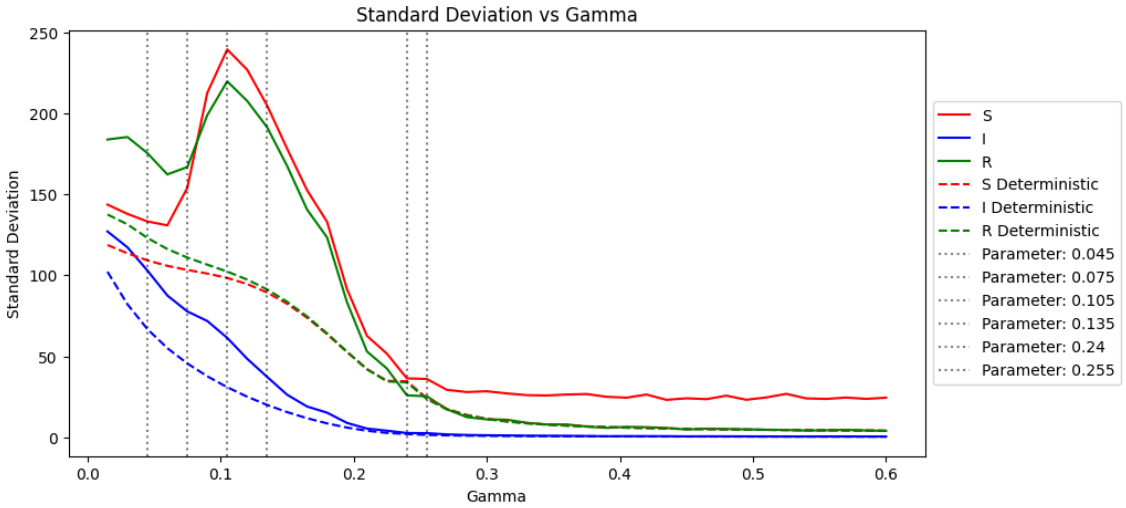


Figure 83

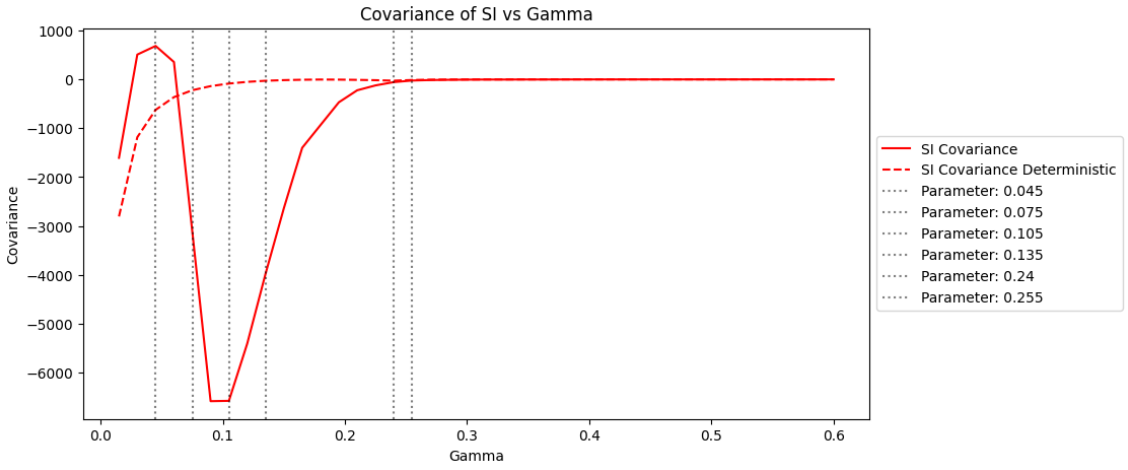


Figure 84

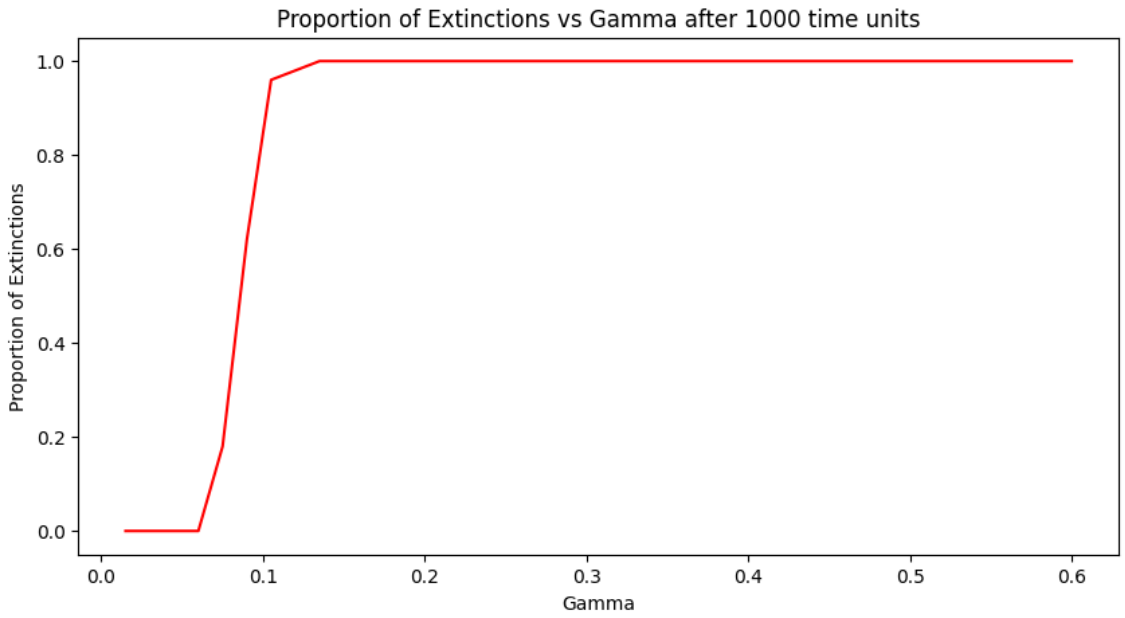


Figure 85

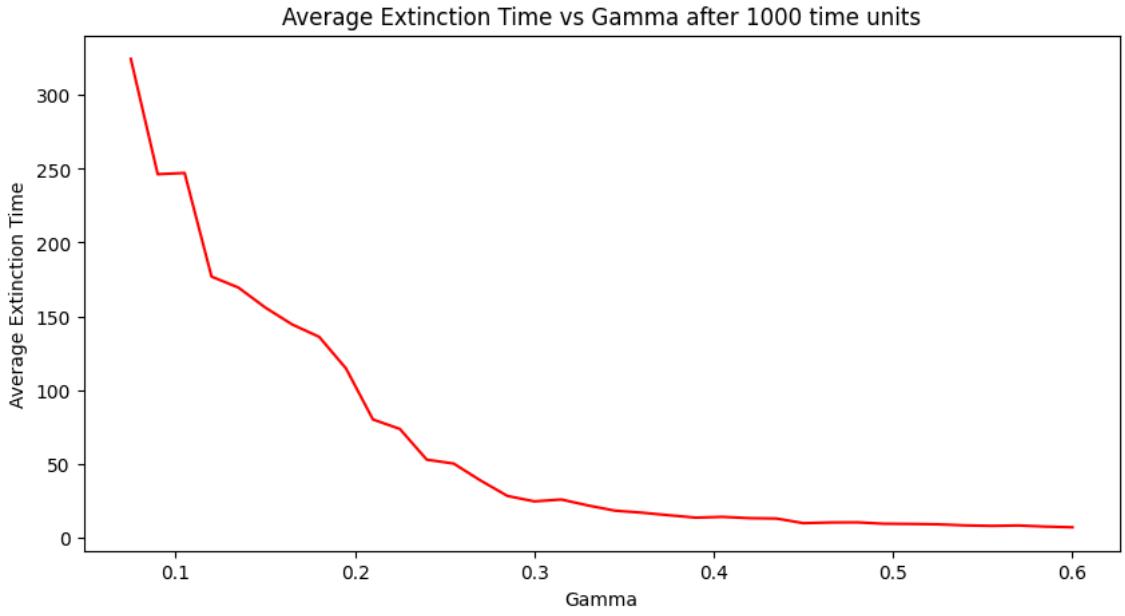


Figure 86

B.3 Fourier Analysis of Stochastic SIR model

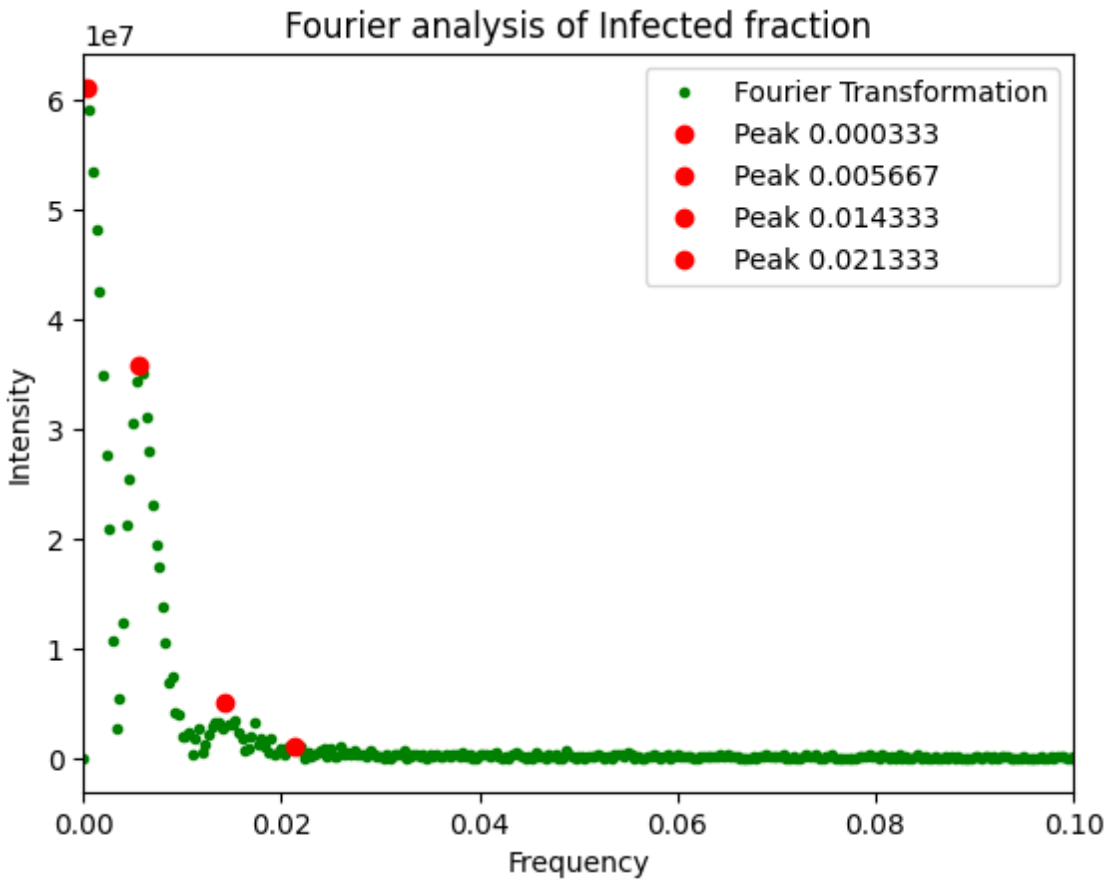


Figure 87: A Fourier transform of a stochastic SIR simulation with $\beta = 0.5$ and $\gamma = 0.1$. Four peaks have been detected and marked in this plot. The first peak has a frequency of $f = 0.000333$, or a period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.014333$, or a period of $T = 46$. The forth peak has a frequency of $f = 0.02133$, or a period of $T = 46$.

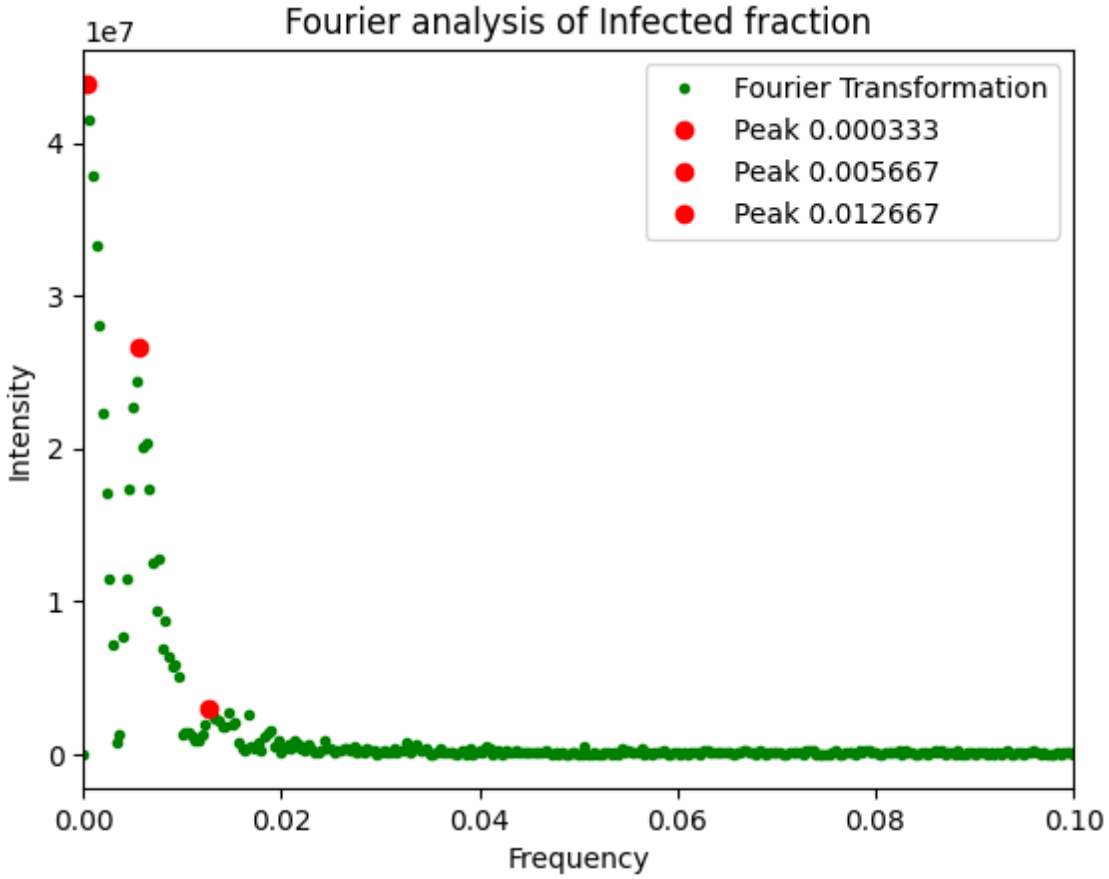


Figure 88: A Fourier transform of a stochastic SIR simulation with $\beta = 0.5$ and $\gamma = 0.15$. Three peaks have been detected and marked in this plot. The first peak has a frequency of $f = 0.000333$, or a period of $T = 3003$. The second peak has a frequency of $f = 0.005667$, or a period of $T = 176$. The third peak has a frequency of $f = 0.012667$, or a period of $T = 69$.

B.4 Stochastic Model with Range of γ and β

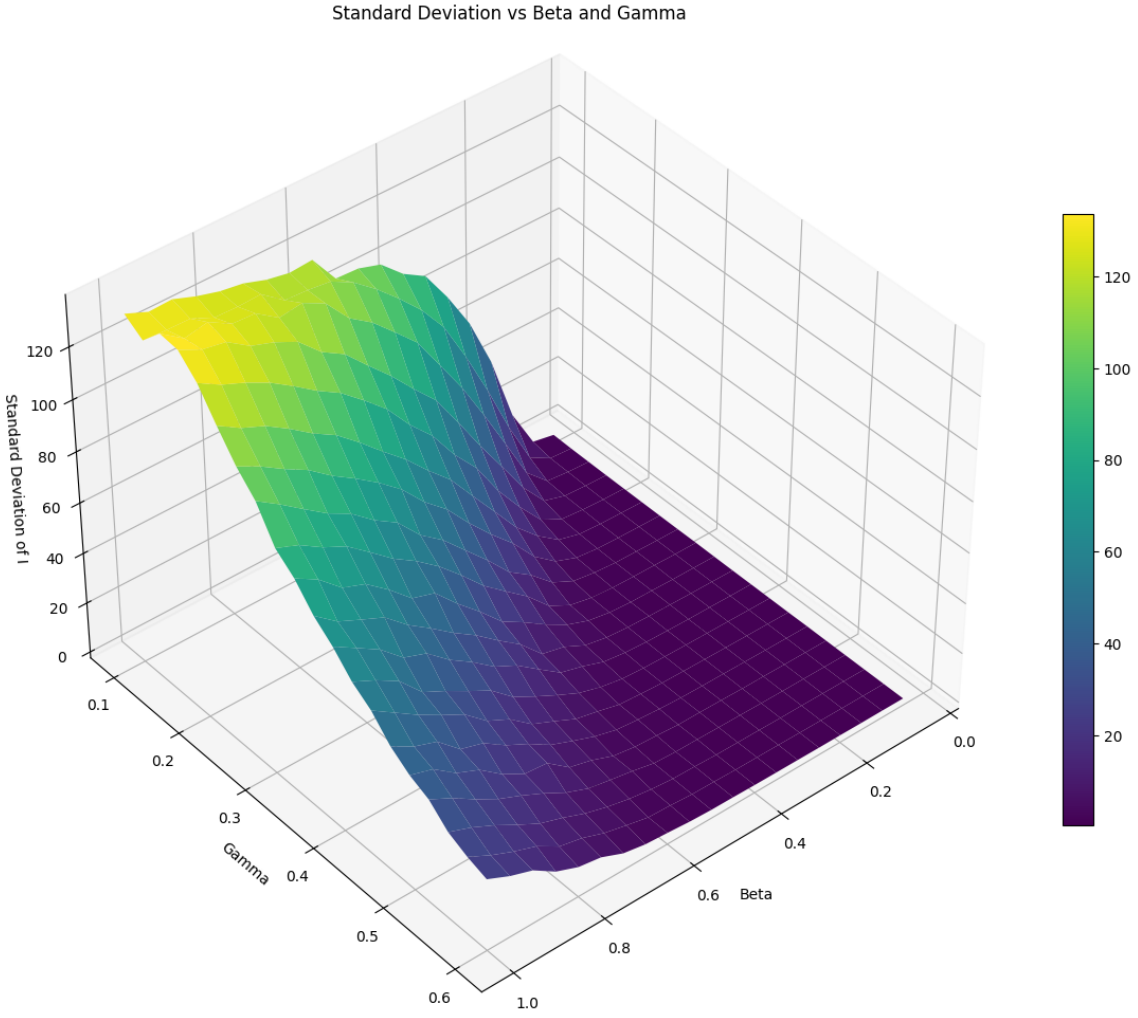


Figure 89

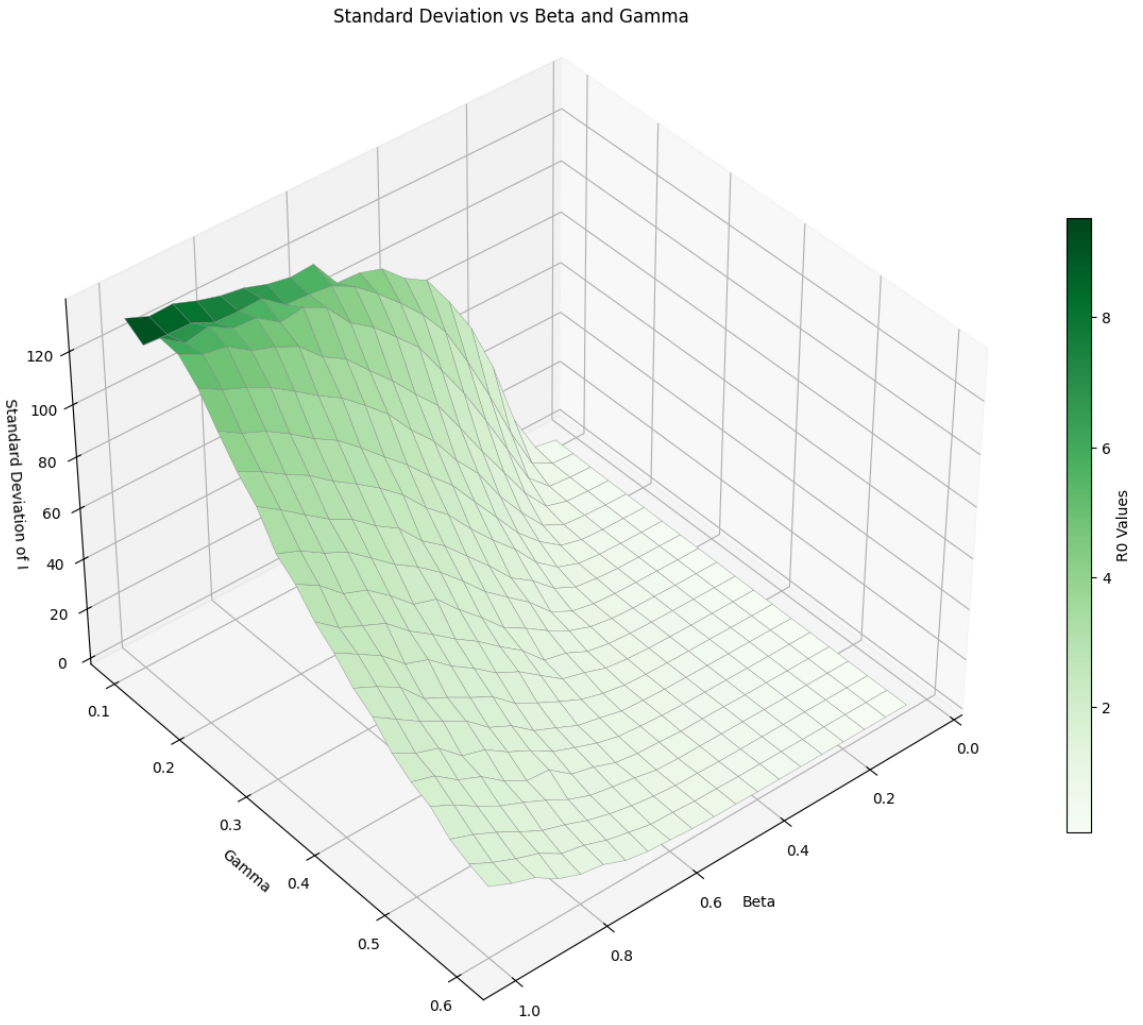

Standard Deviation vs Beta and Gamma
Standard Deviation of I
120
100
80
60
40
20
0
0.1
0.2
0.3
0.4
0.5
0.6
Gamma
1.0
0.8
0.6
0.4
0.2
0.0
Beta
8
6
4
2
R0 Values

Figure 90

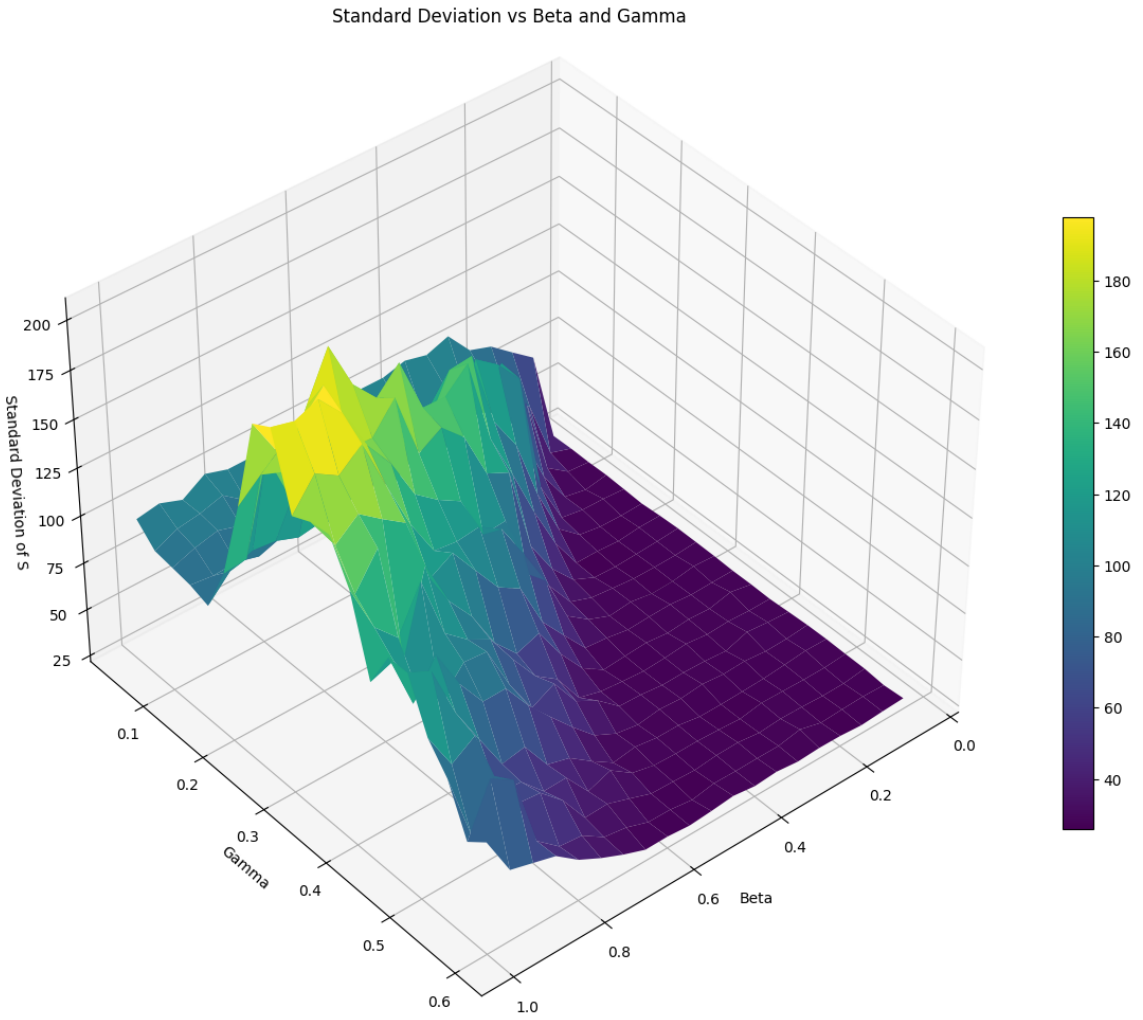

Standard Deviation vs Beta and Gamma
Standard Deviation of S
200
175
150
125
100
75
50
25
0.1
0.2
0.3
0.4
0.5
0.6
Gamma
0.0
0.2
0.4
0.6
0.8
1.0
Beta
180
160
140
120
100
80
60
40

Figure 91

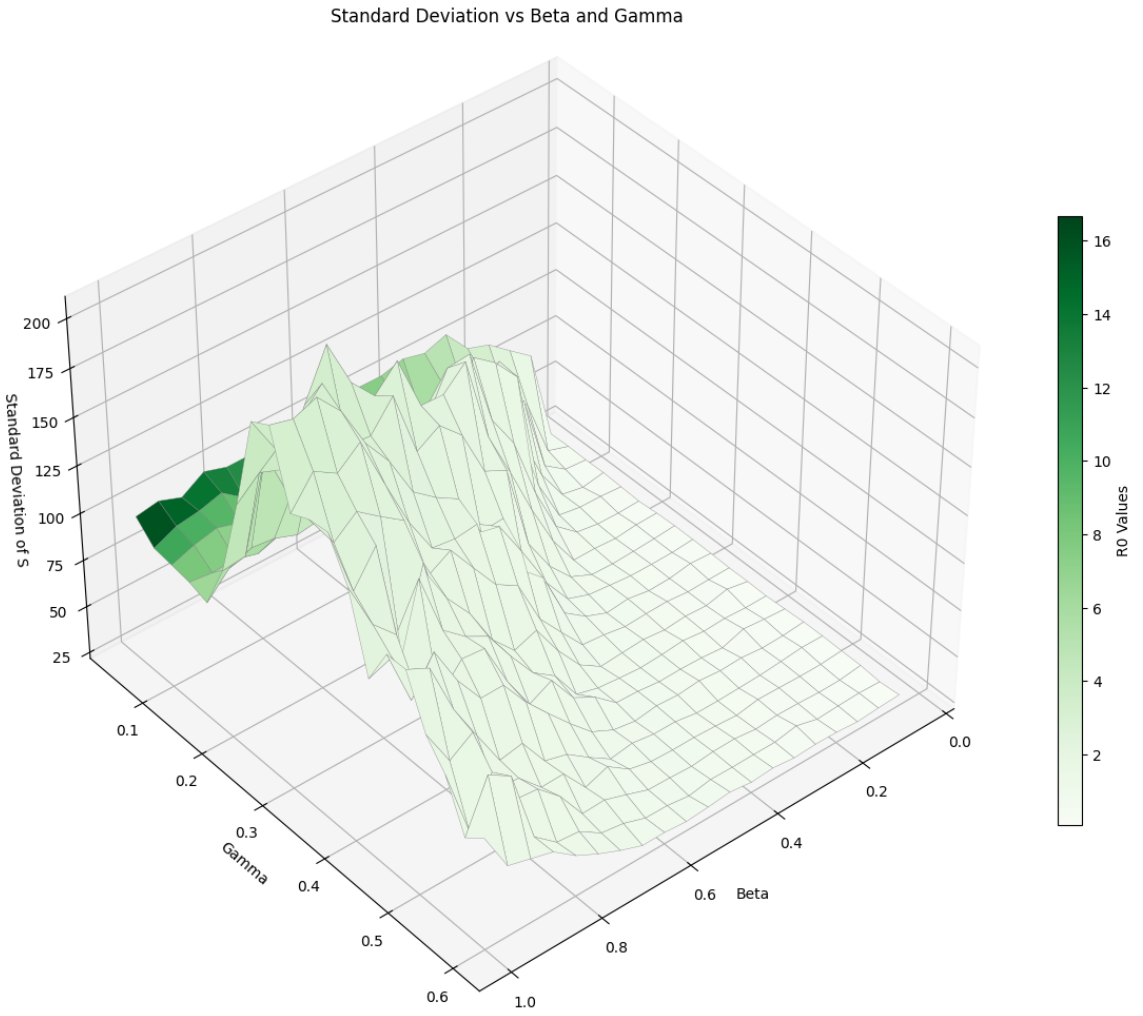

Standard Deviation vs Beta and Gamma
Standard Deviation of S
200
175
150
125
100
75
50
25
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
0.0
0.2
0.4
0.6
0.8
1.0
R0 Values
16
14
12
10
8
6
4
2

Figure 92

e

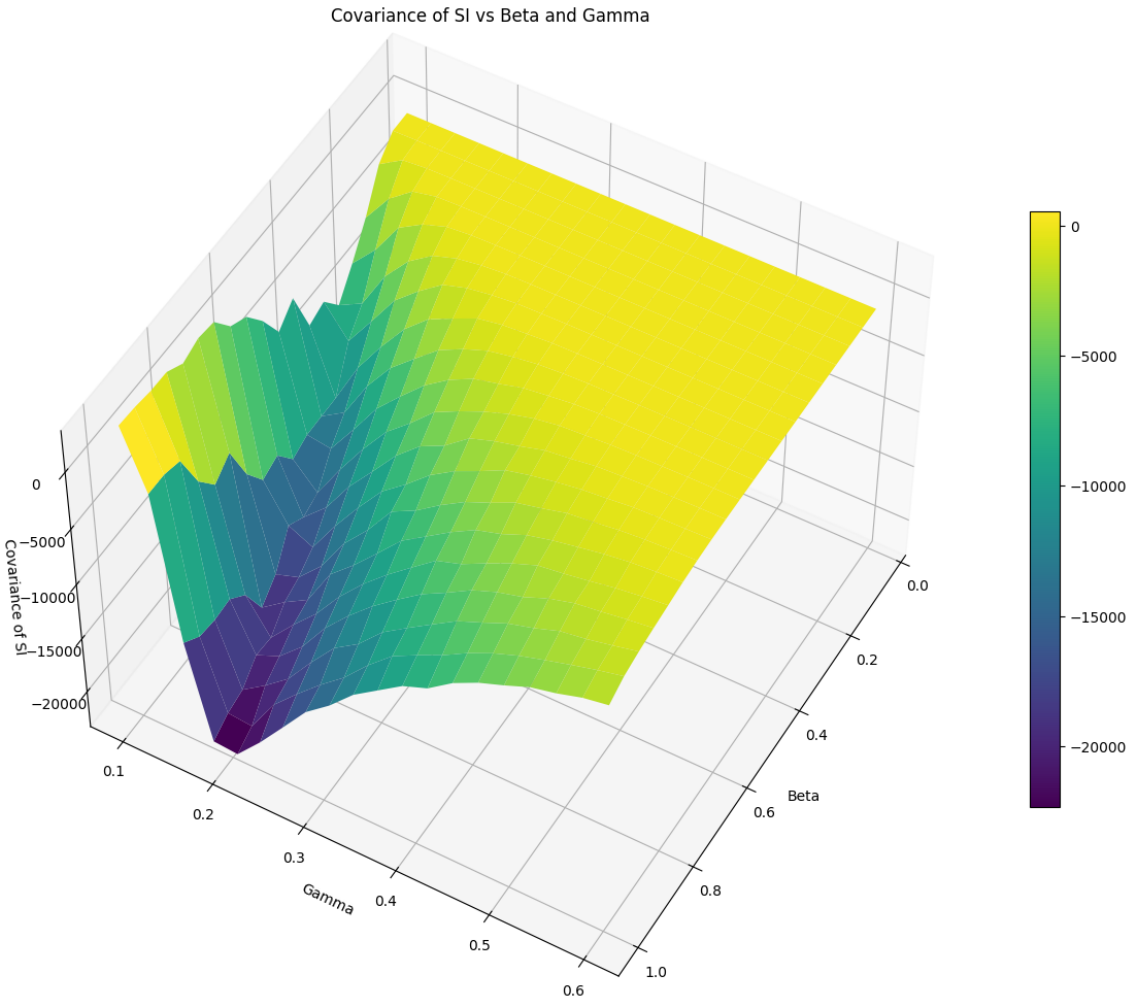

Covariance of SI vs Beta and Gamma
Covariance of SI
0
−5000
−10000
−15000
−20000
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
0.0
0.2
0.4
0.6
0.8
1.0
0
−5000
−10000
−15000
−20000

Figure 93

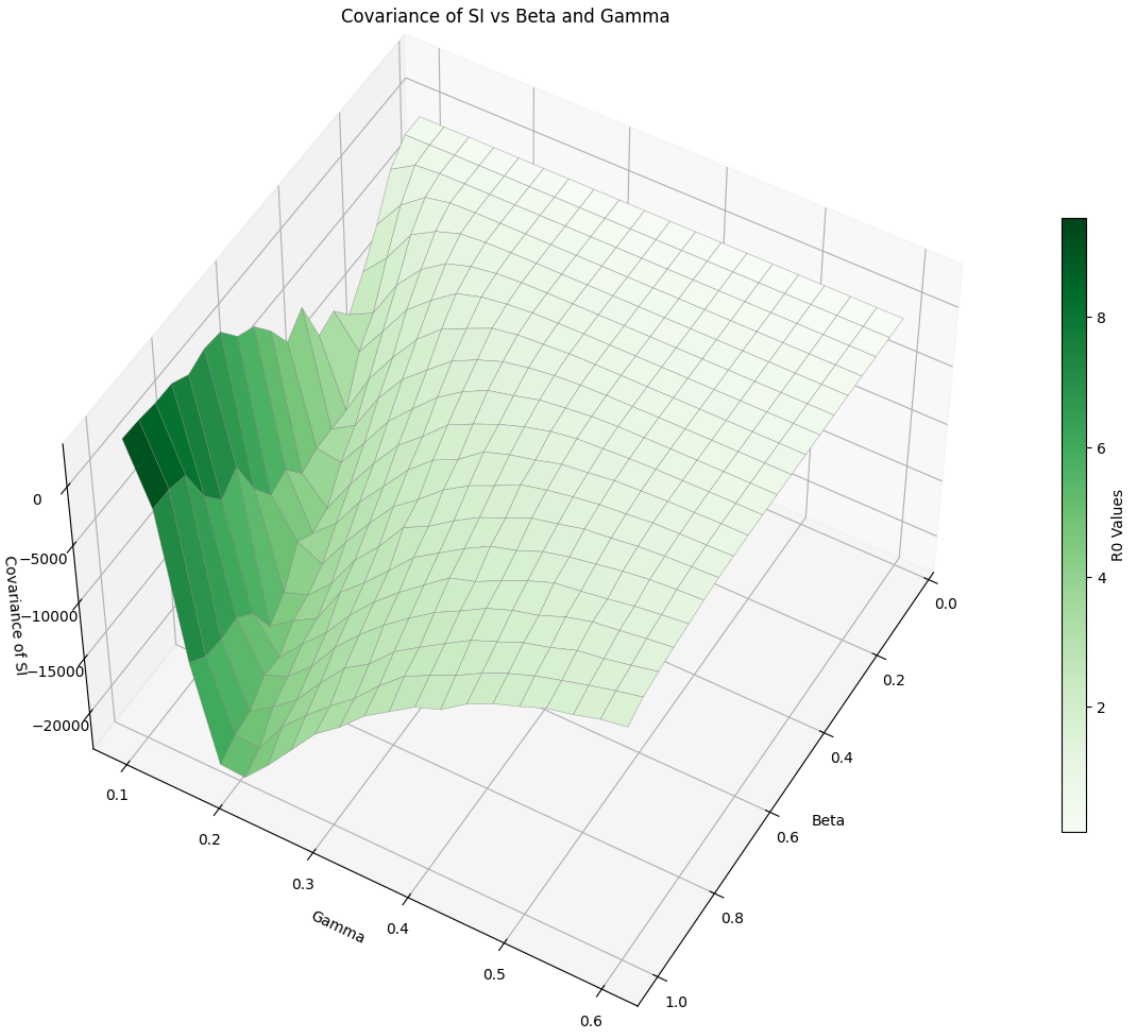

Covariance of SI vs Beta and Gamma
Covariance of SI
0
−5000
−10000
−15000
−20000
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
0.0
0.2
0.4
0.6
0.8
1.0
R0 Values
2
4
6
8

Figure 94

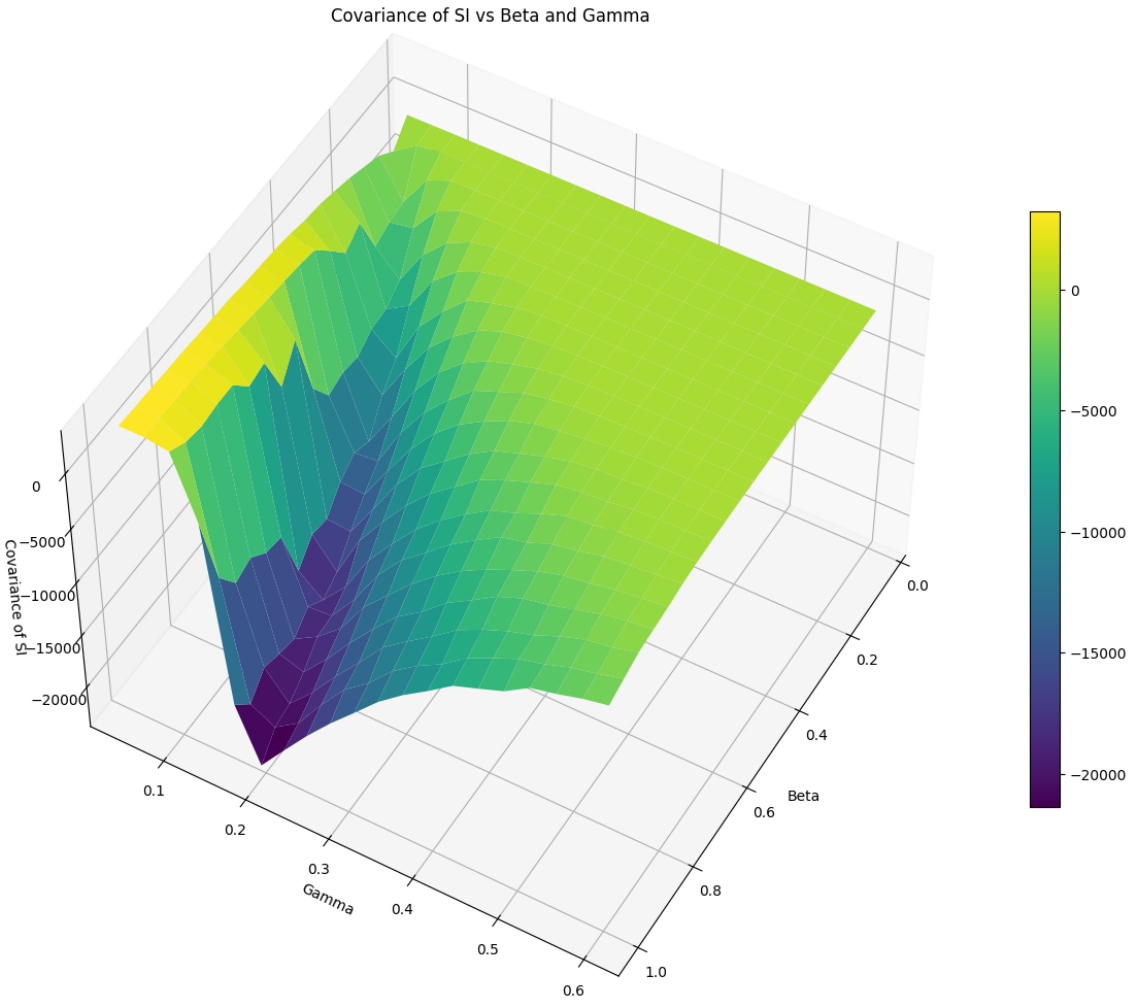

Covariance of SI vs Beta and Gamma
Covariance of SI
0
−5000
−10000
−15000
−20000
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
0.0
0.2
0.4
0.6
0.8
1.0
0
−5000
−10000
−15000
−20000

Figure 95

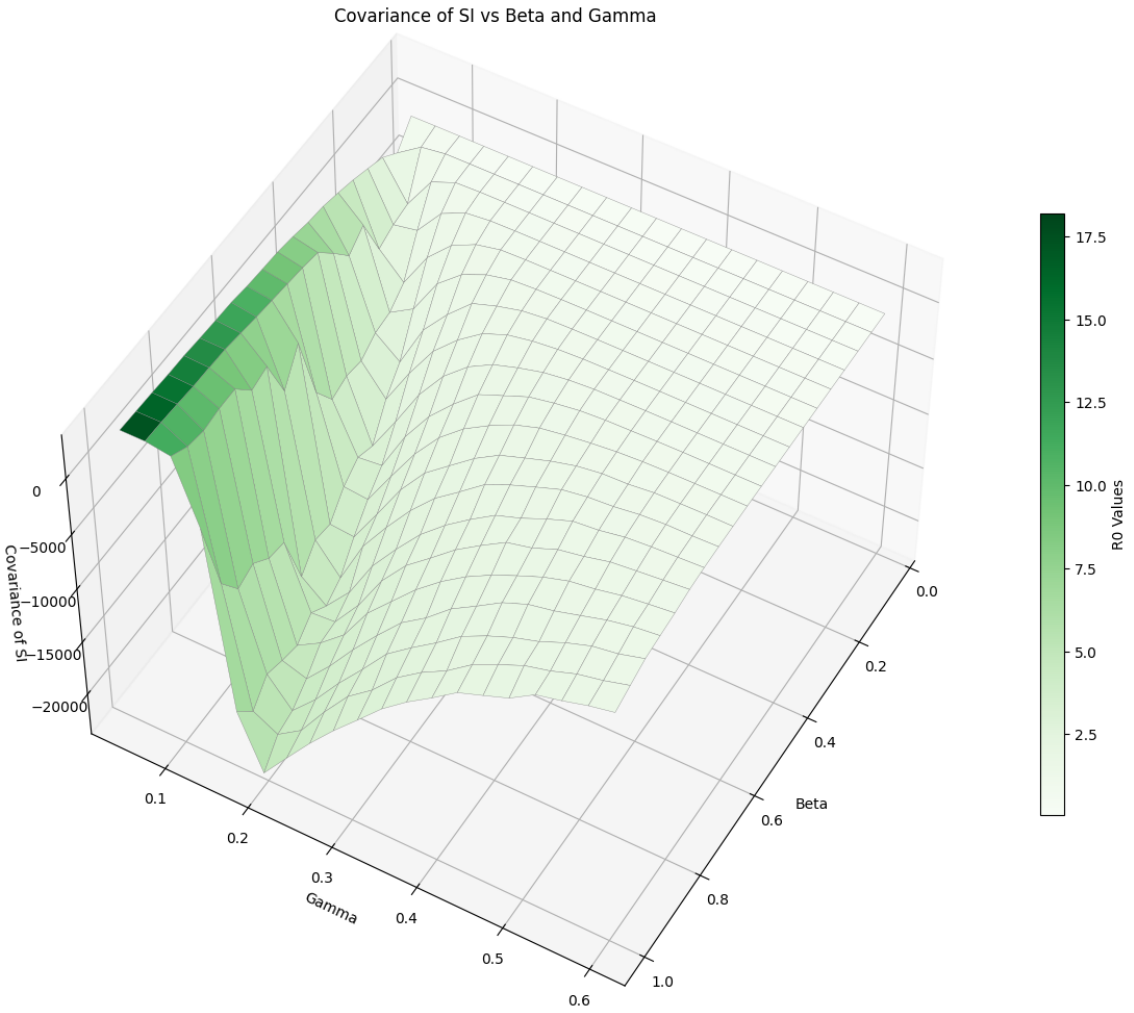

Covariance of SI vs Beta and Gamma
Covariance of SI
0
−5000
−10000
−15000
−20000
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
0.0
0.2
0.4
0.6
0.8
1.0
R0 Values
17.5
15.0
12.5
10.0
7.5
5.0
2.5

Figure 96

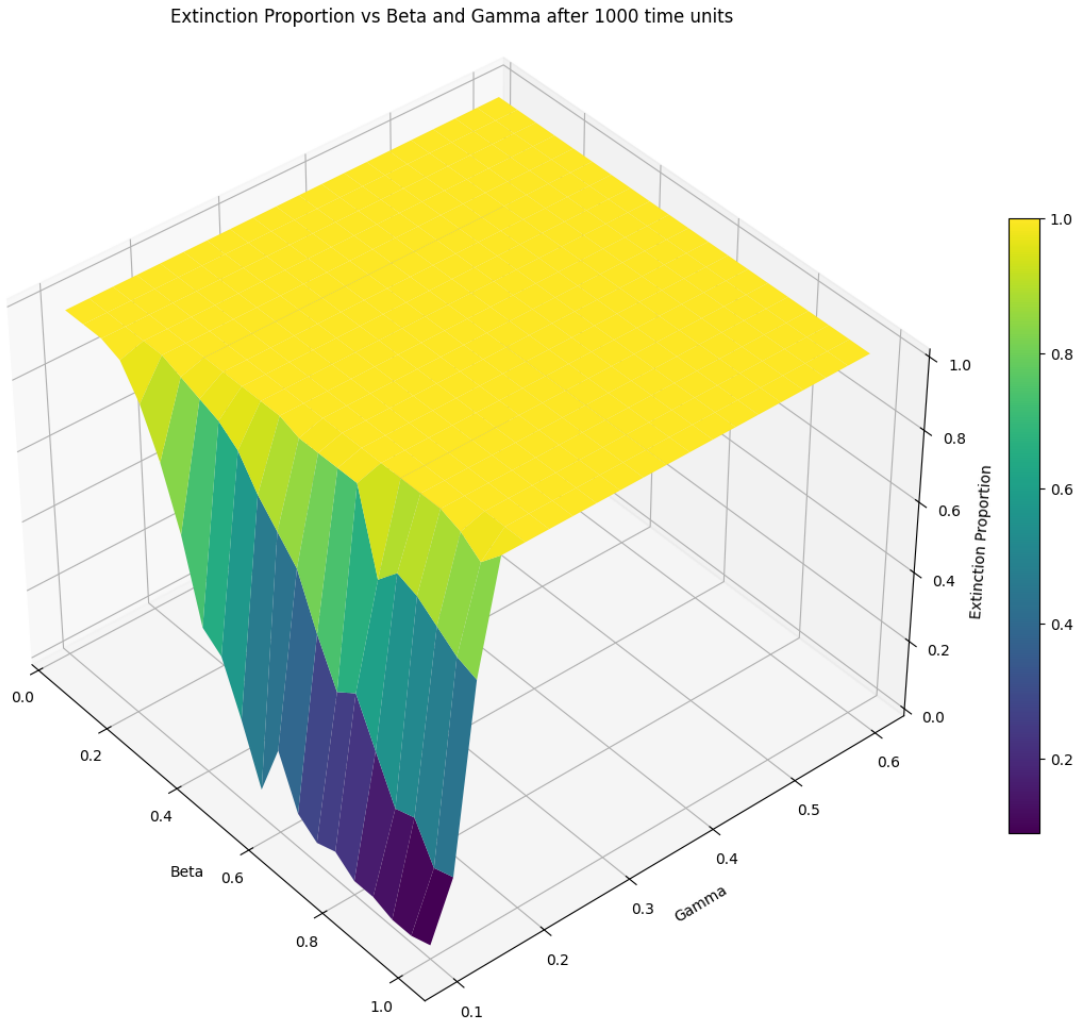

Extinction Proportion vs Beta and Gamma after 1000 time units
Beta
Gamma
Extinction Proportion
0.0
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
0.6

Figure 97

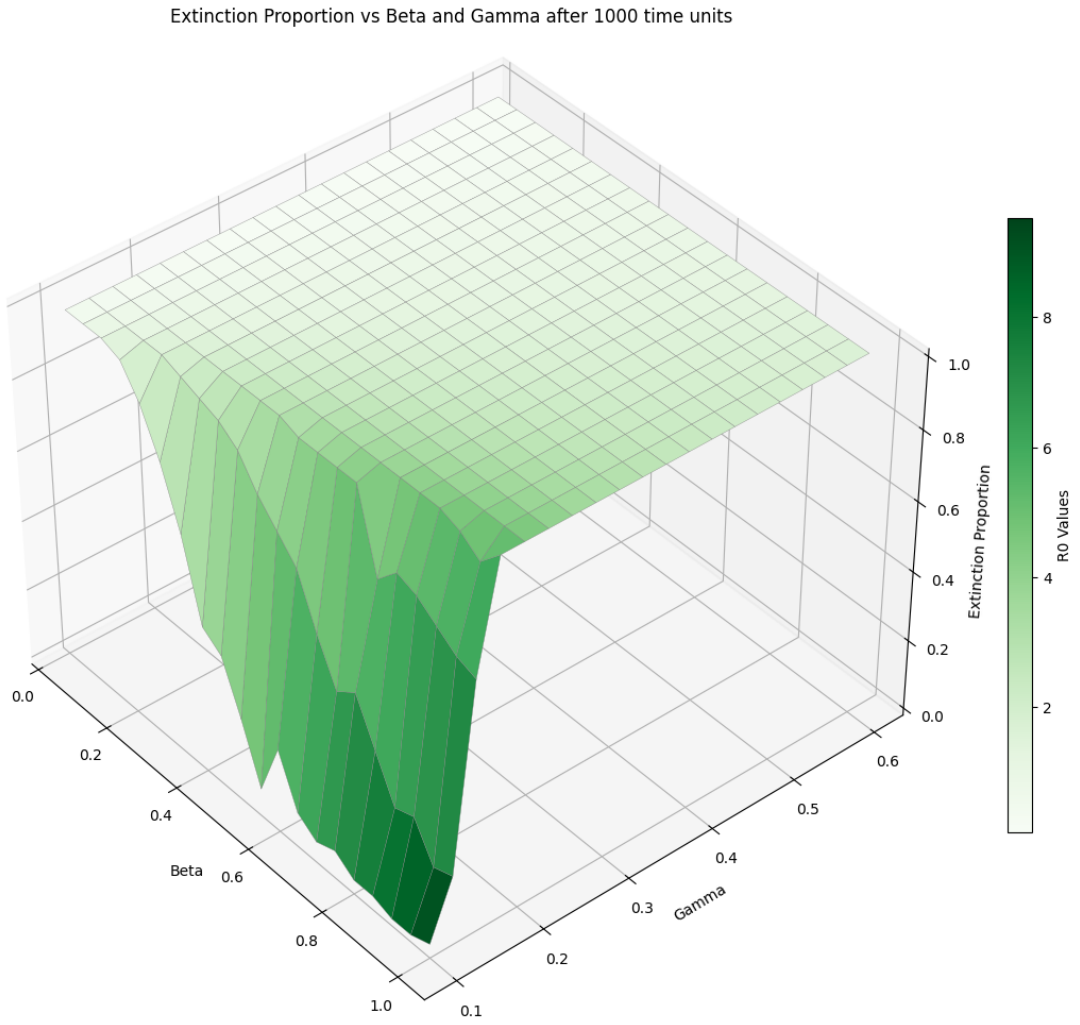

Extinction Proportion vs Beta and Gamma after 1000 time units
Beta
0.0
0.2
0.4
0.6
0.8
1.0
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Extinction Proportion
0.0
0.2
0.4
0.6
0.8
1.0
R0 Values
2
4
6
8

Figure 98

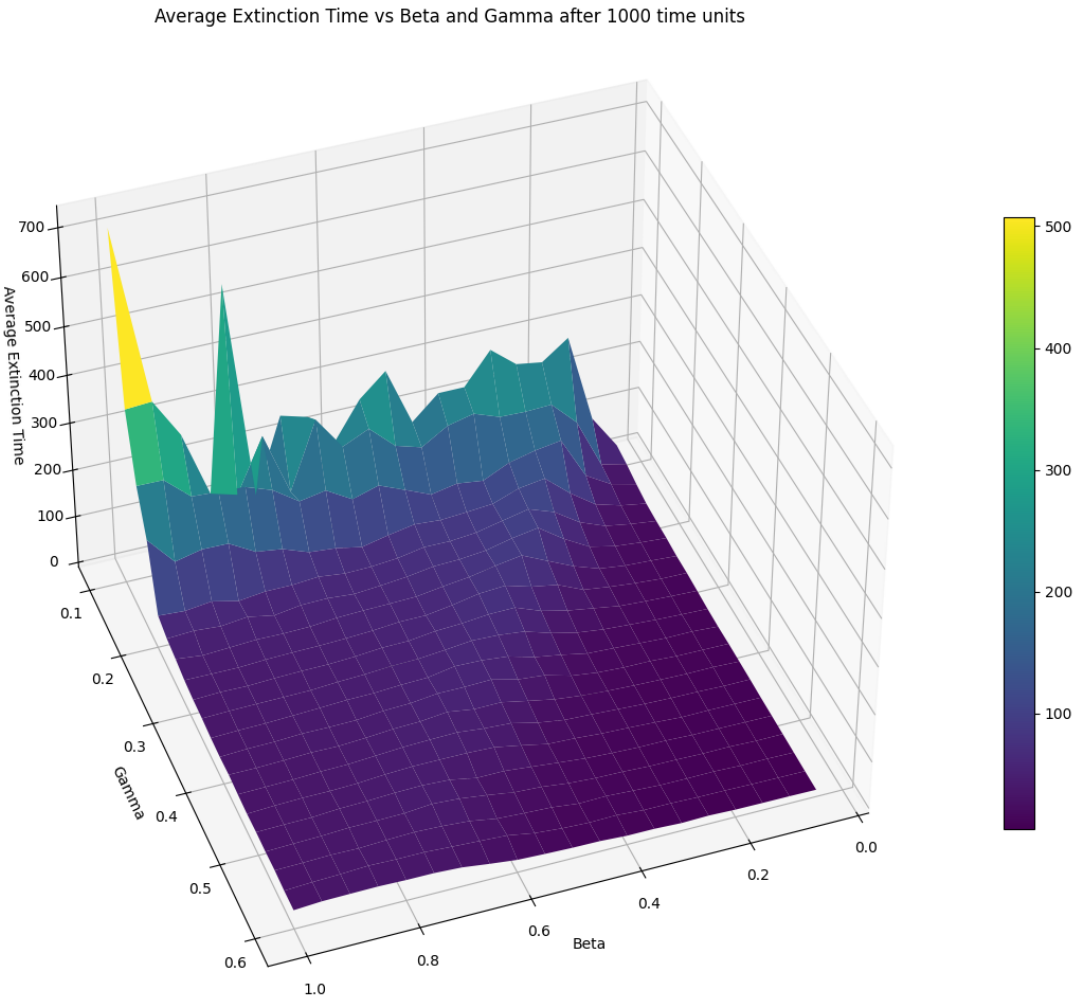

Average Extinction Time vs Beta and Gamma after 1000 time units
Average Extinction Time
700
600
500
400
300
200
100
0
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
1.0
0.8
0.6
0.4
0.2
0.0
500
400
300
200
100

Figure 99

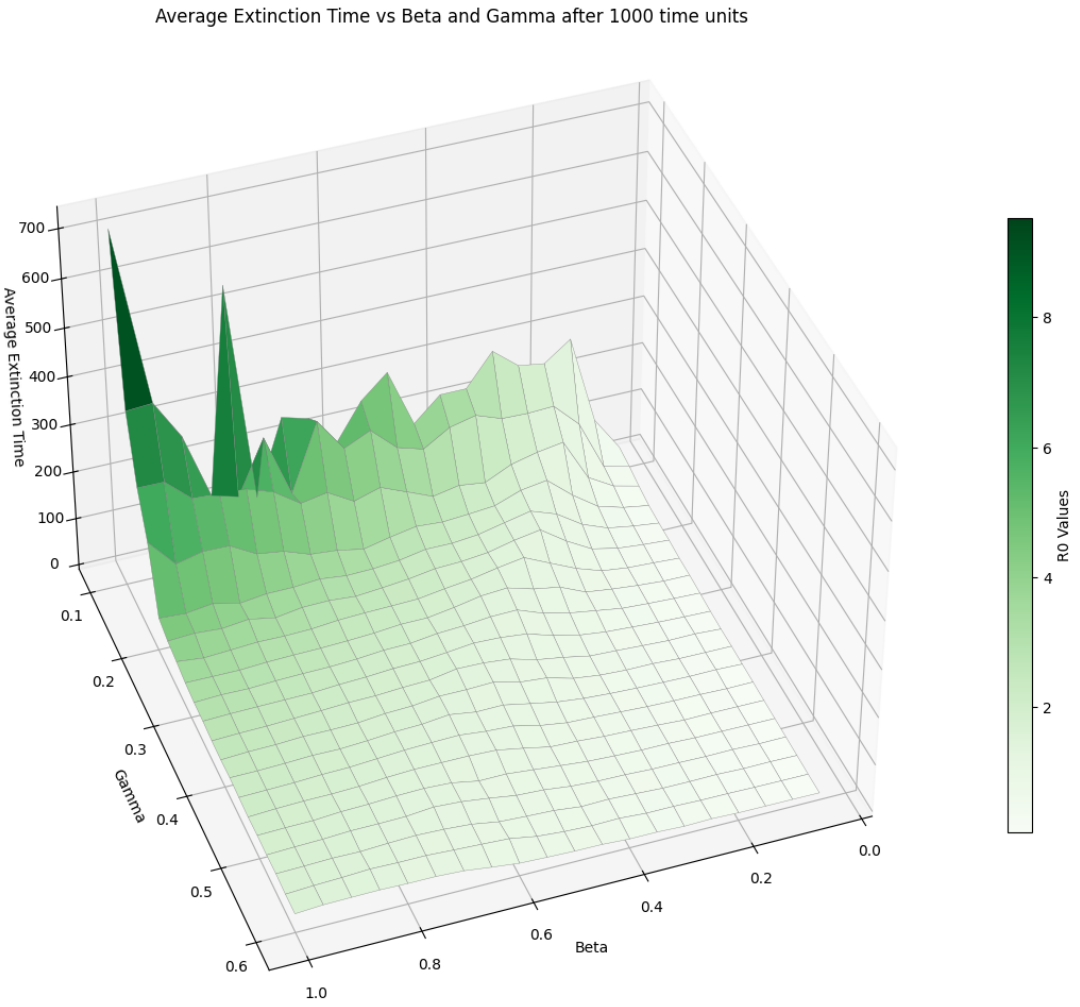

Average Extinction Time vs Beta and Gamma after 1000 time units
Average Extinction Time
700
600
500
400
300
200
100
0
Gamma
0.1
0.2
0.3
0.4
0.5
0.6
Beta
1.0
0.8
0.6
0.4
0.2
0.0
R0 Values
8
6
4
2

Figure 100

B.5 Stochastic Model with Range of Population

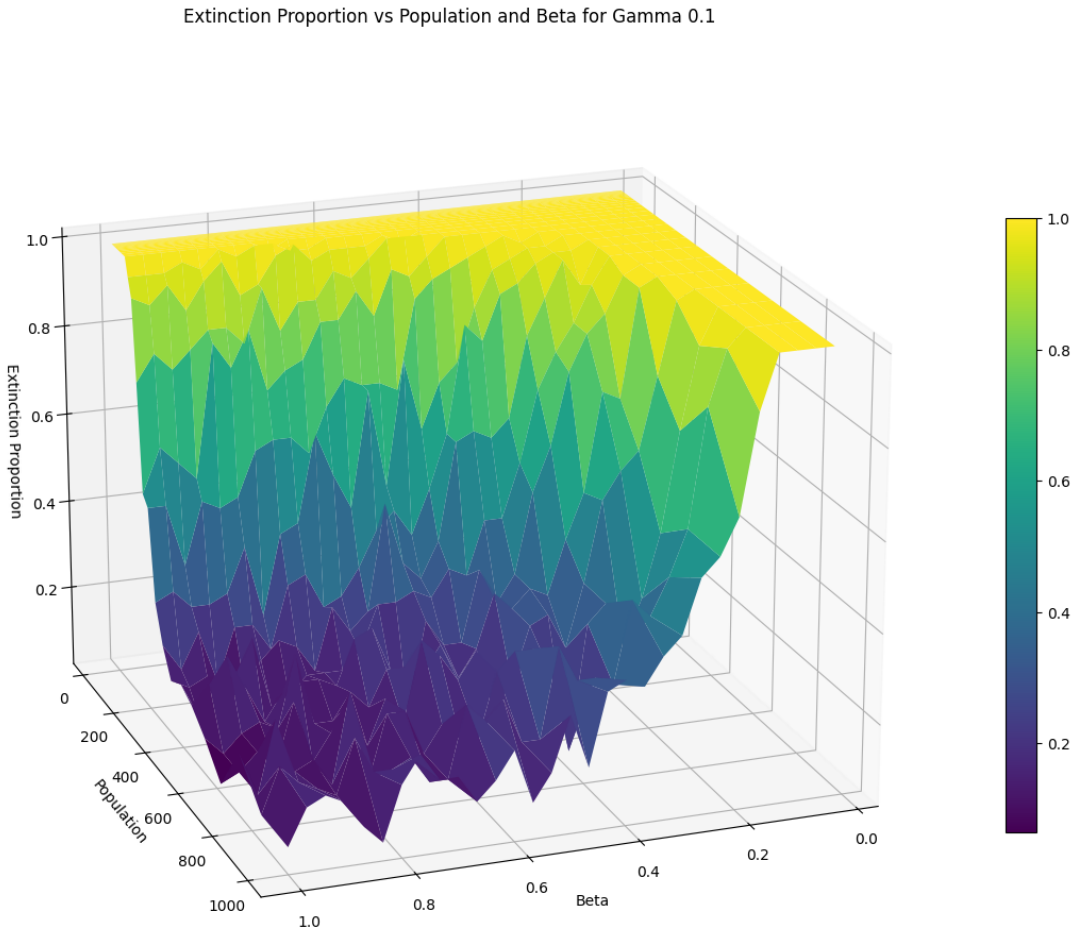


Figure 101

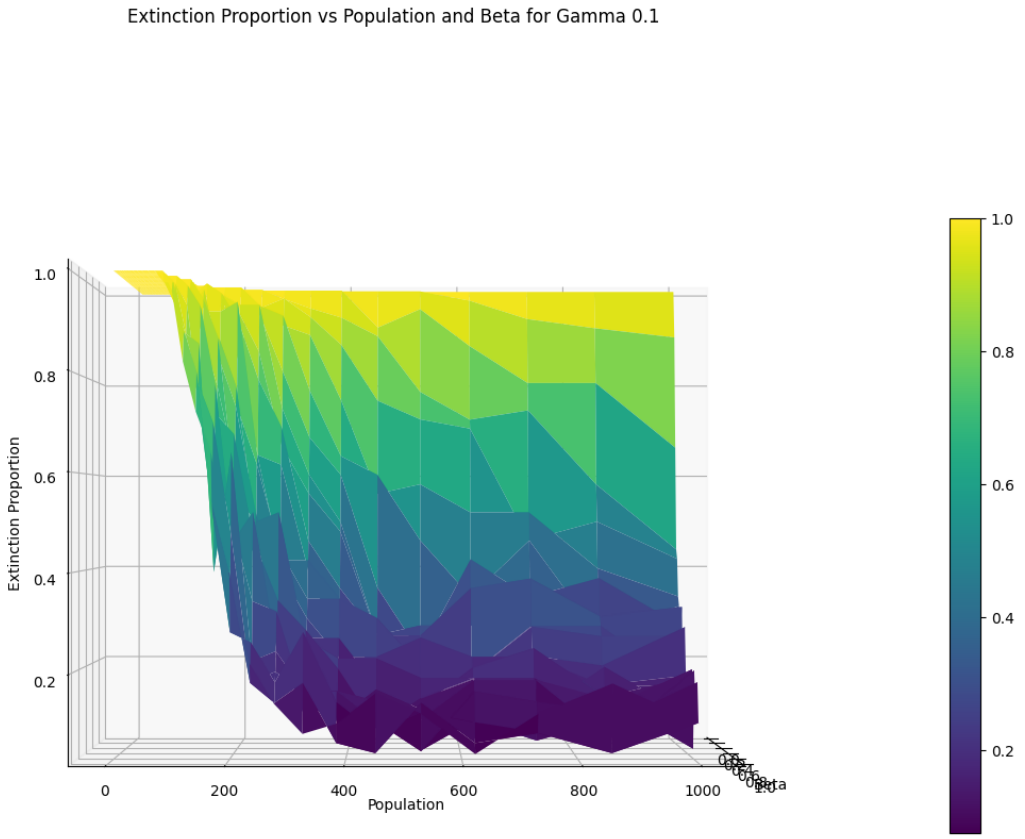

Extinction Proportion vs Population and Beta for Gamma 0.1
Extinction Proportion
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
Population
Beta
1.0
0.8
0.6
0.4
0.2

Figure 102

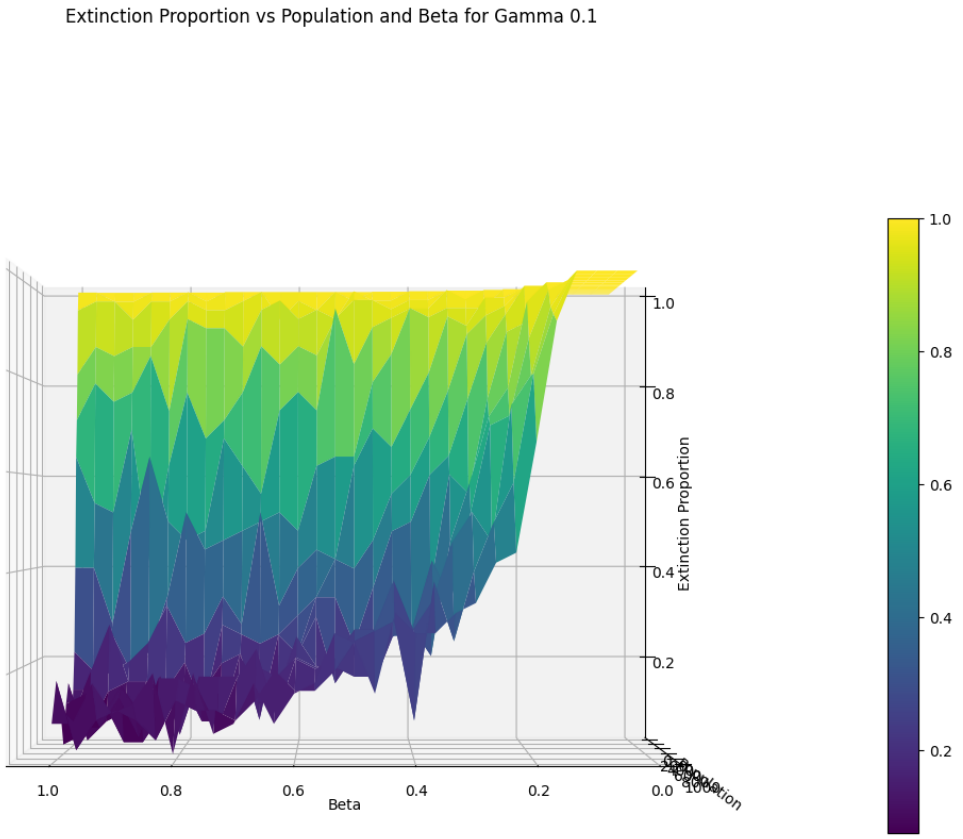
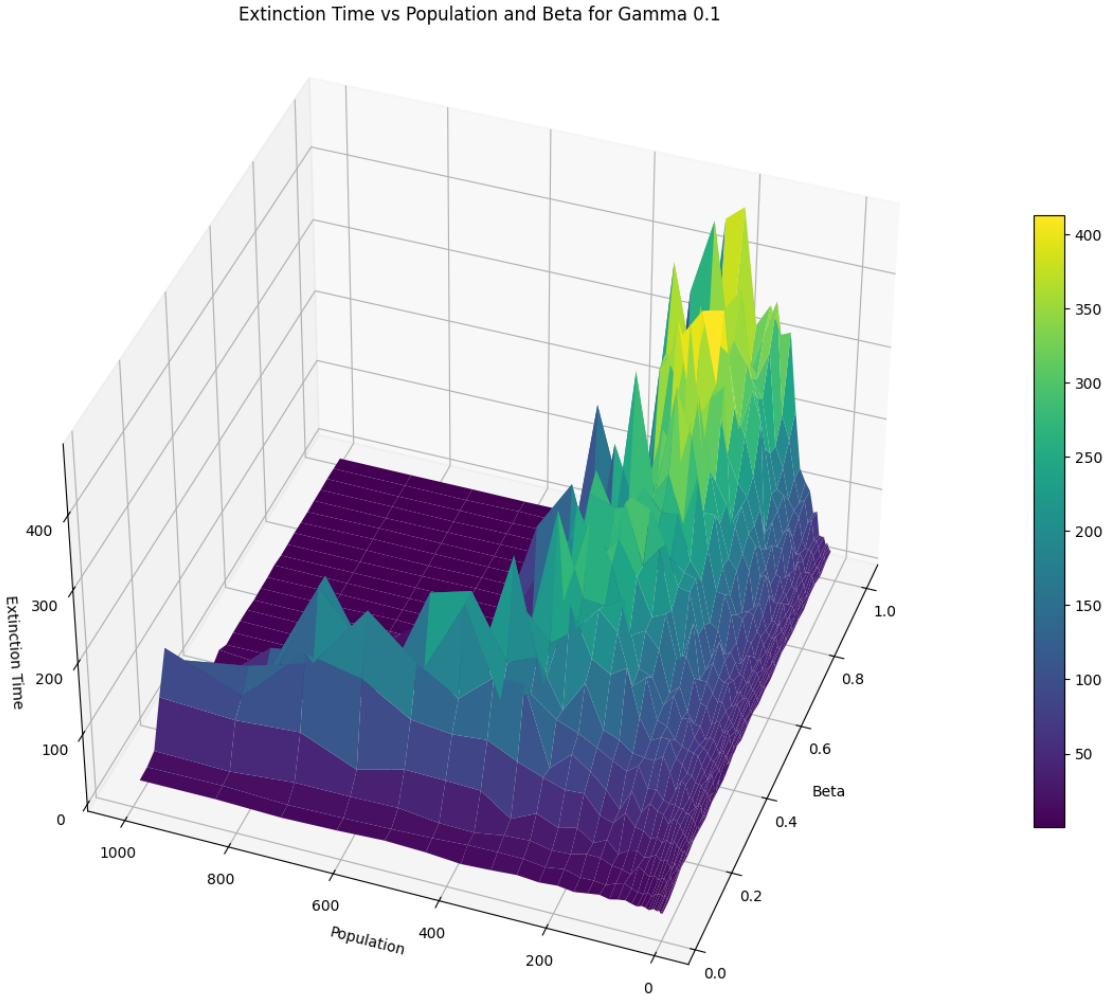

Extinction Proportion vs Population and Beta for Gamma 0.1
Beta
Extinction Proportion
1.0
0.8
0.6
0.4
0.2
0.0

Figure 103


Extinction Time vs Population and Beta for Gamma 0.1
Extinction Time
Population
Beta
0
100
200
300
400
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
250
300
350
400

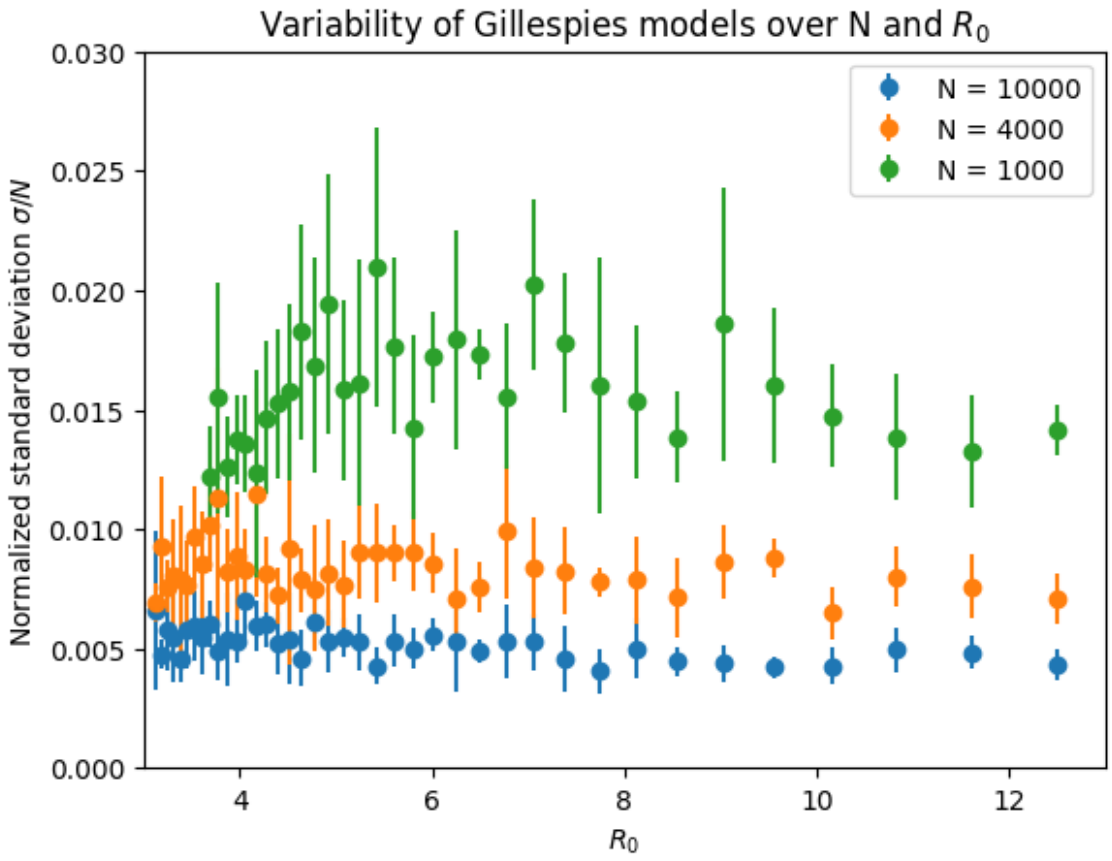


Figure 104: The normalized standard deviation of stochastic SIR models. The models were run with populations of 1000, 4000 and 10.000. The $\beta = 0.3$ and γ was varied between 0.04 and 0.16. For the $N = 1000$ models, the γ was only increased to a value of approximately 0.13, as simulations running at higher values went extinct to frequently, which caused a significant slow-down in the code. The plot shows that smaller populations have a higher normalized standard deviation, and are therefore more affected by stochastic effects. Additionally, the $N = 1000$ dataset also appears to show a peak around $R_0 = 5.5$. This might indicate that this value exhibits stochastic resonance.

B.6 Networks

B.6.1 Barabasi Albert Network

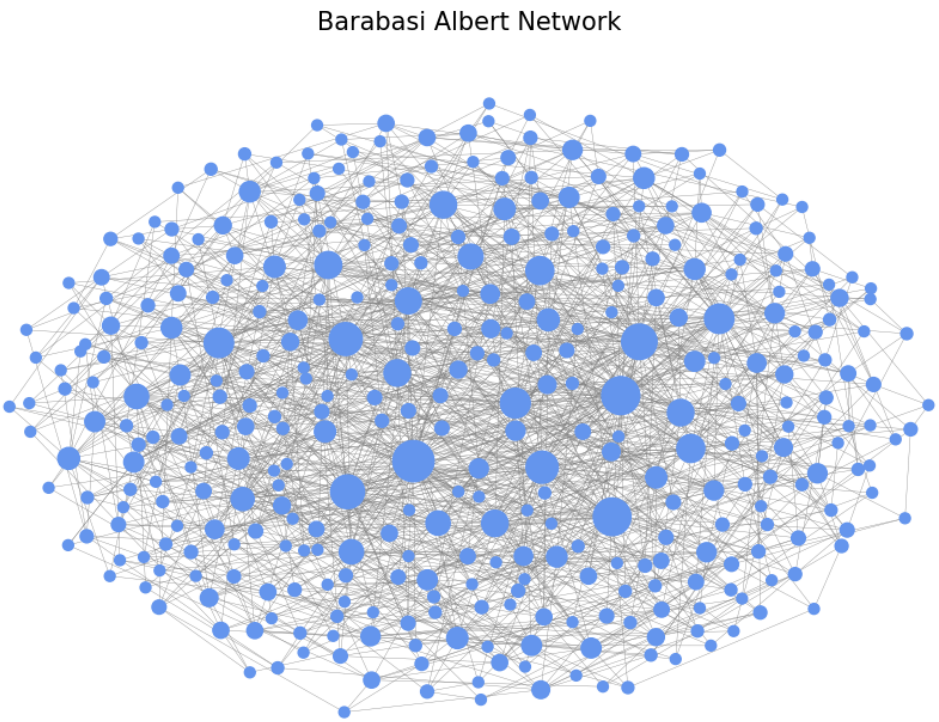


Figure 105

e

B.6.2 Watts Strogatz Network

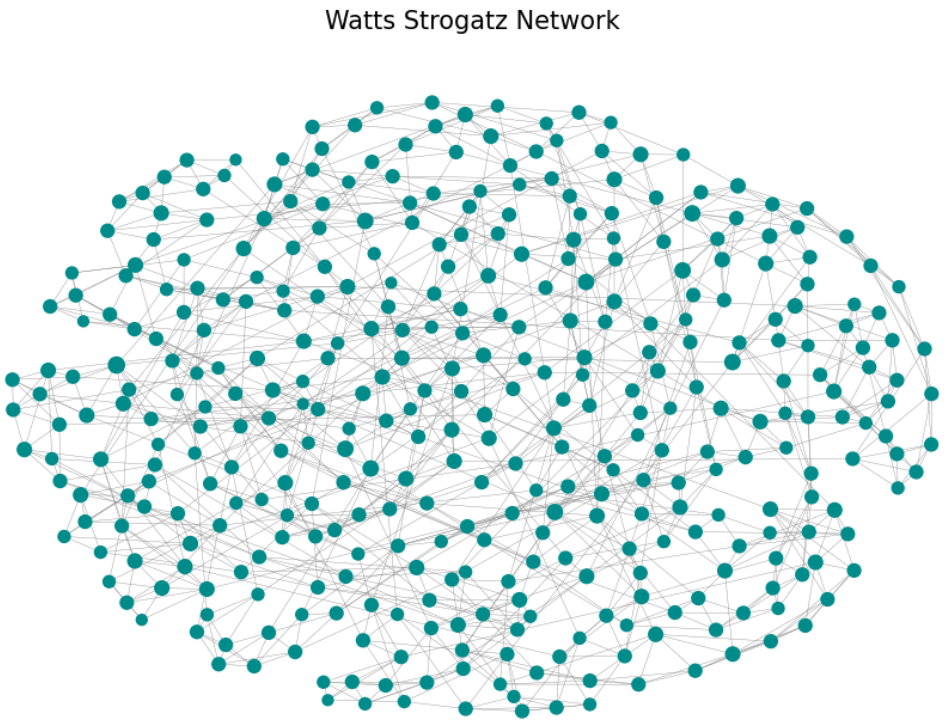


Figure 106

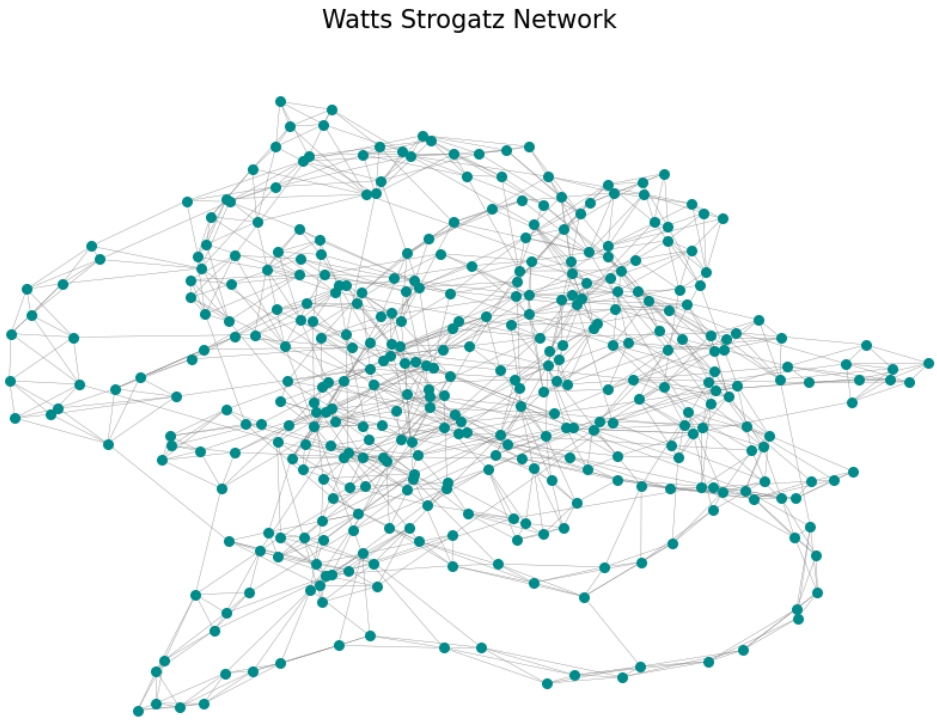


Figure 107

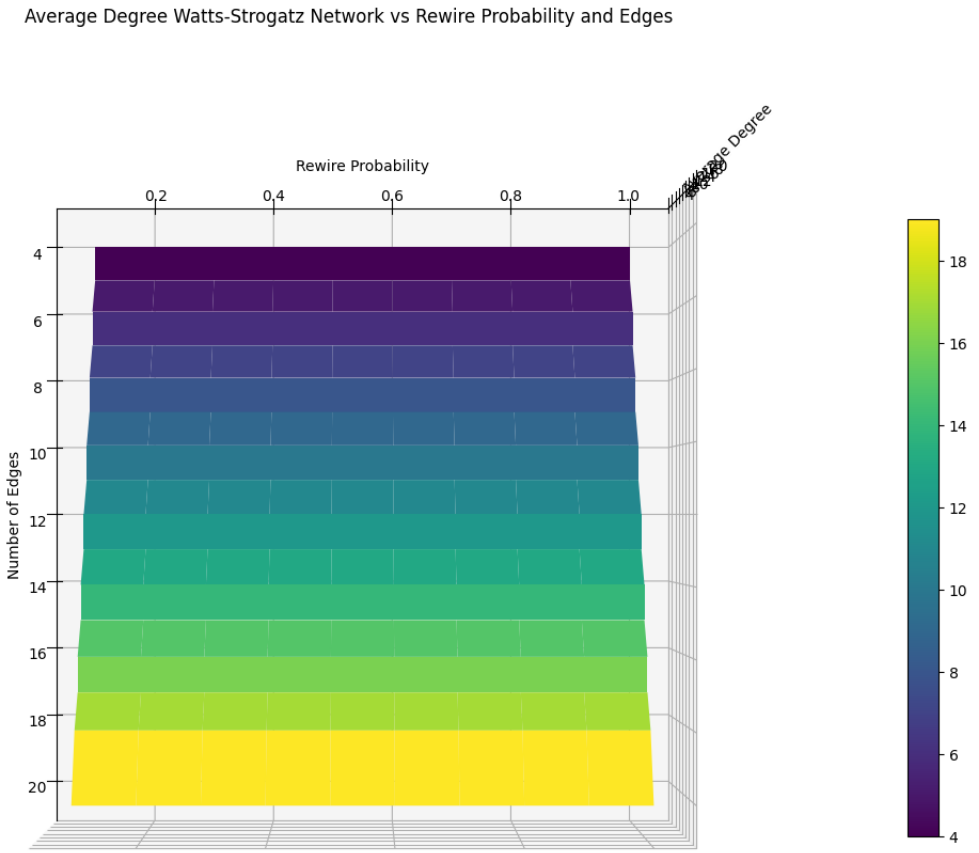

Average Degree Watts-Strogatz Network vs Rewire Probability and Edges
Rewire Probability
0.2
0.4
0.6
0.8
1.0
Number of Edges
4
6
8
10
12
14
16
18
20
Average Degree
4
6
8
10
12
14
16
18

Figure 108

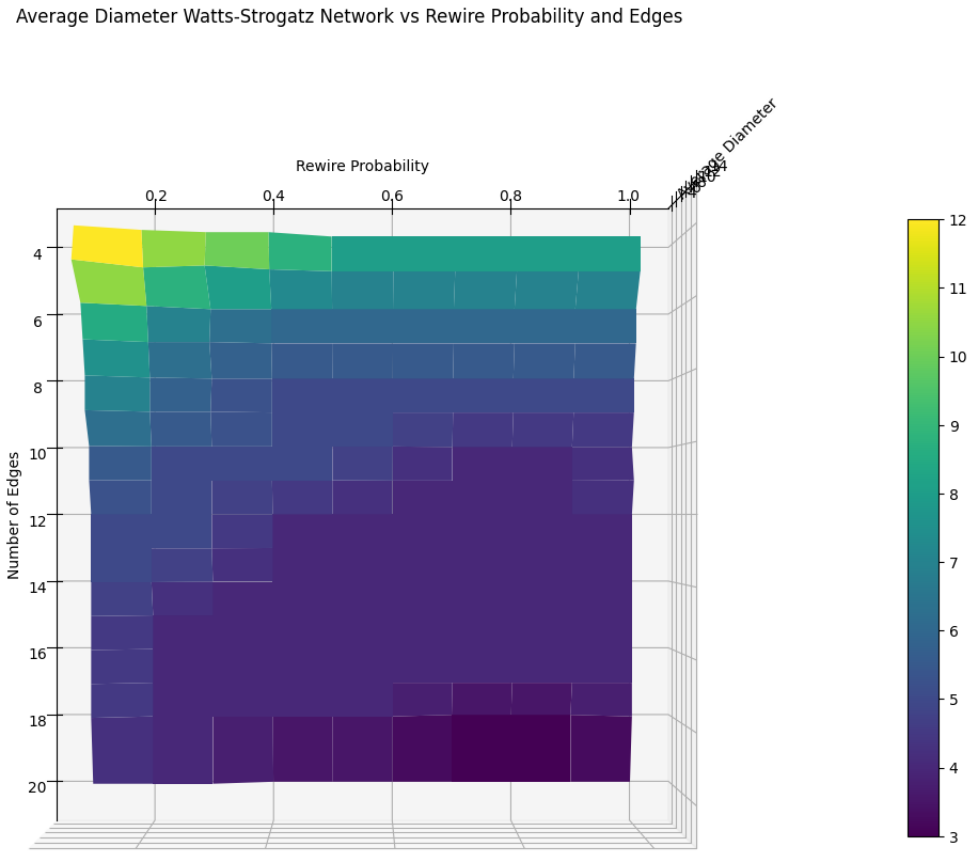

Average Diameter Watts-Strogatz Network vs Rewire Probability and Edges
Rewire Probability
0.2
0.4
0.6
0.8
1.0
Number of Edges
4
6
8
10
12
14
16
18
20
12
11
10
9
8
7
6
5
4
3

Figure 109

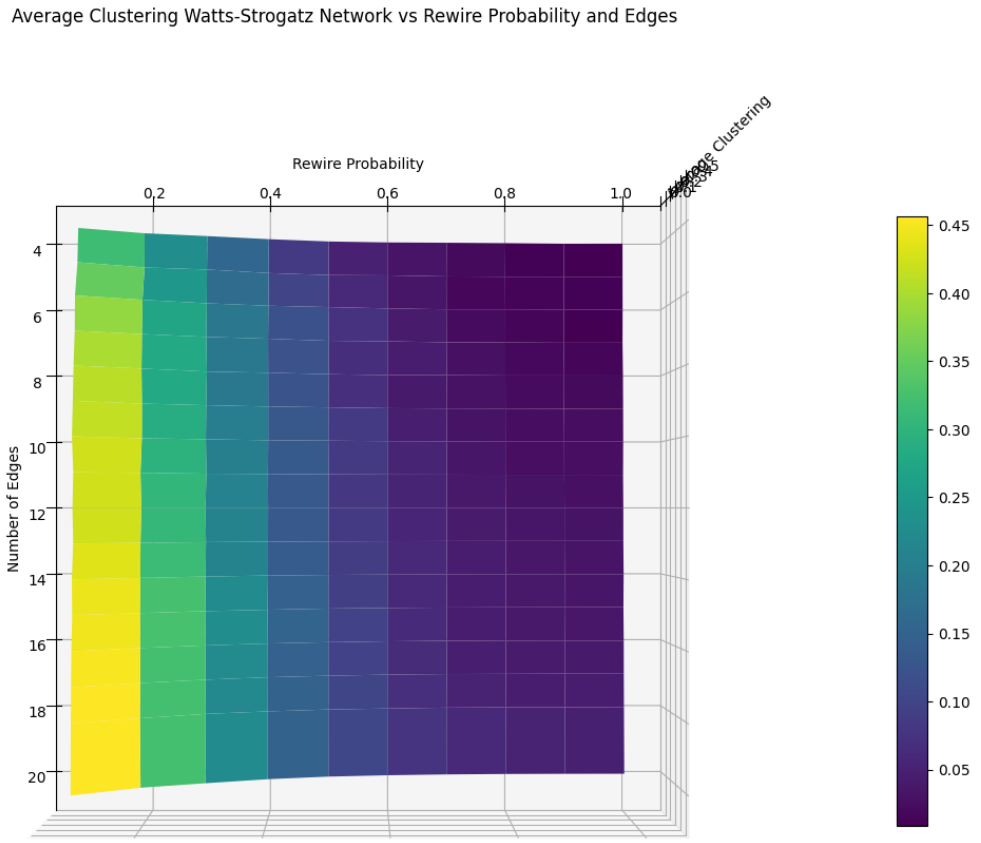

Average Clustering Watts-Strogatz Network vs Rewire Probability and Edges
Rewire Probability
0.2
0.4
0.6
0.8
1.0
Number of Edges
4
6
8
10
12
14
16
18
20
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05

Figure 110

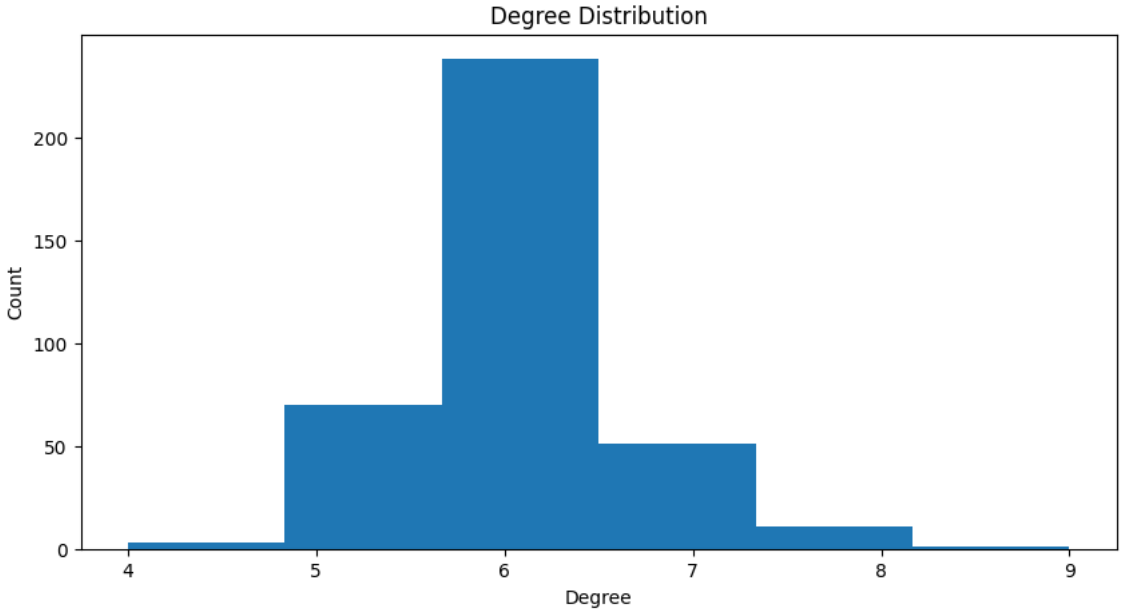


Figure 111

B.6.3 Erdos Reyni Network

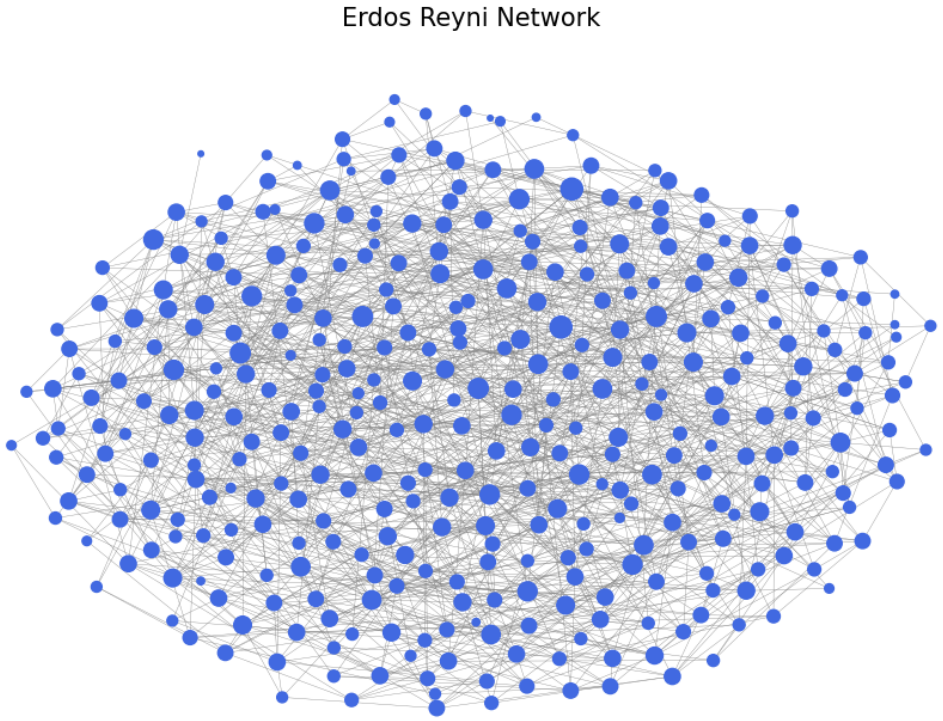


Figure 112

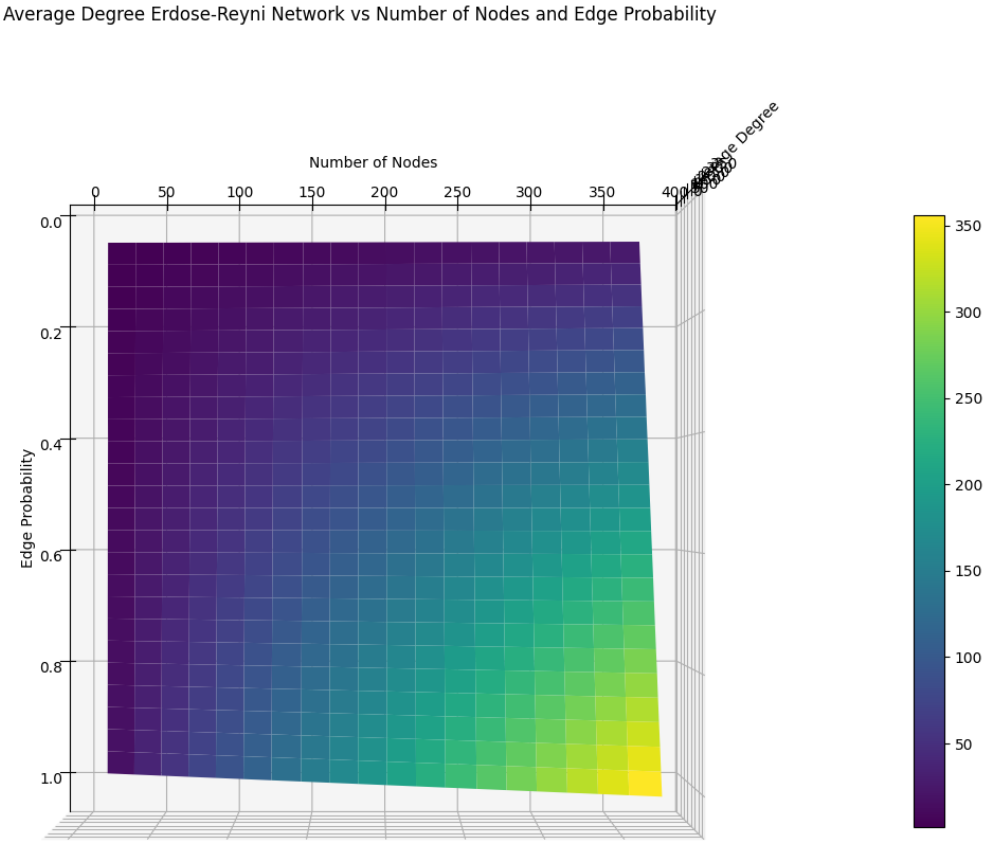

Average Degree Erdose-Reyni Network vs Number of Nodes and Edge Probability
Number of Nodes
0
50
100
150
200
250
300
350
Edge Probability
0.0
0.2
0.4
0.6
0.8
1.0
Average Degree
350
300
250
200
150
100
50

Figure 113

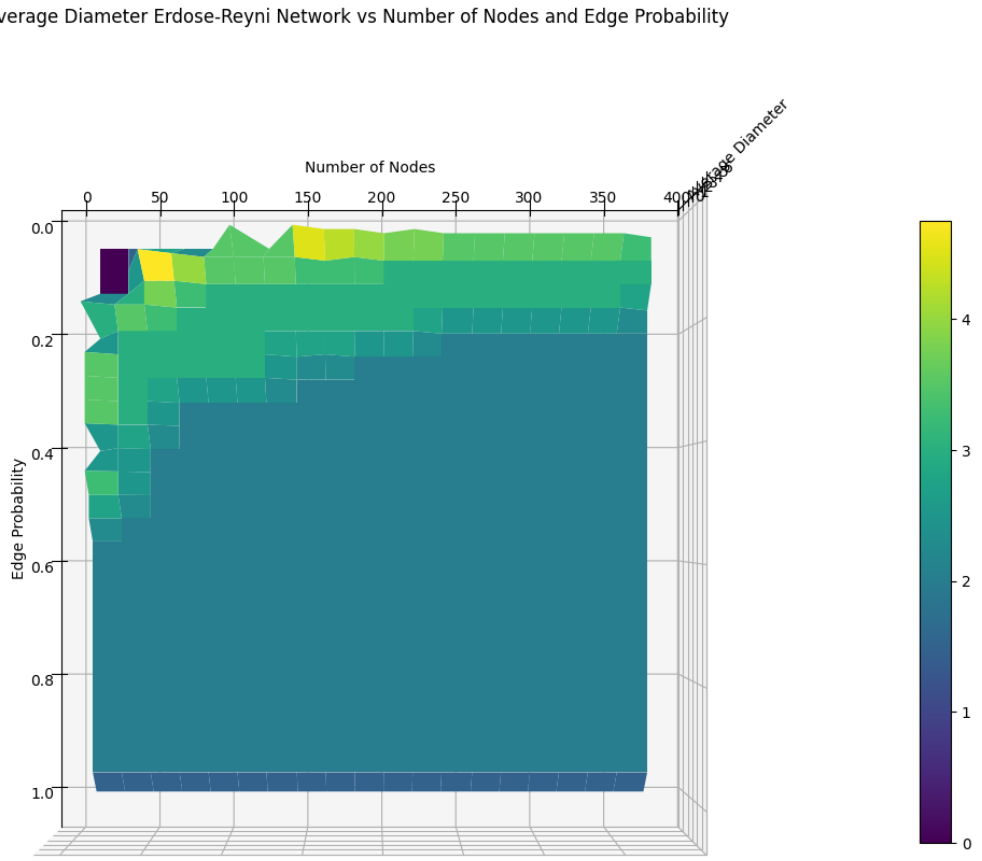

Average Diameter Erdose-Reyni Network vs Number of Nodes and Edge Probability
Number of Nodes
0
50
100
150
200
250
300
350
Edge Probability
0.0
0.2
0.4
0.6
0.8
1.0
4
3
2
1
0

Figure 114

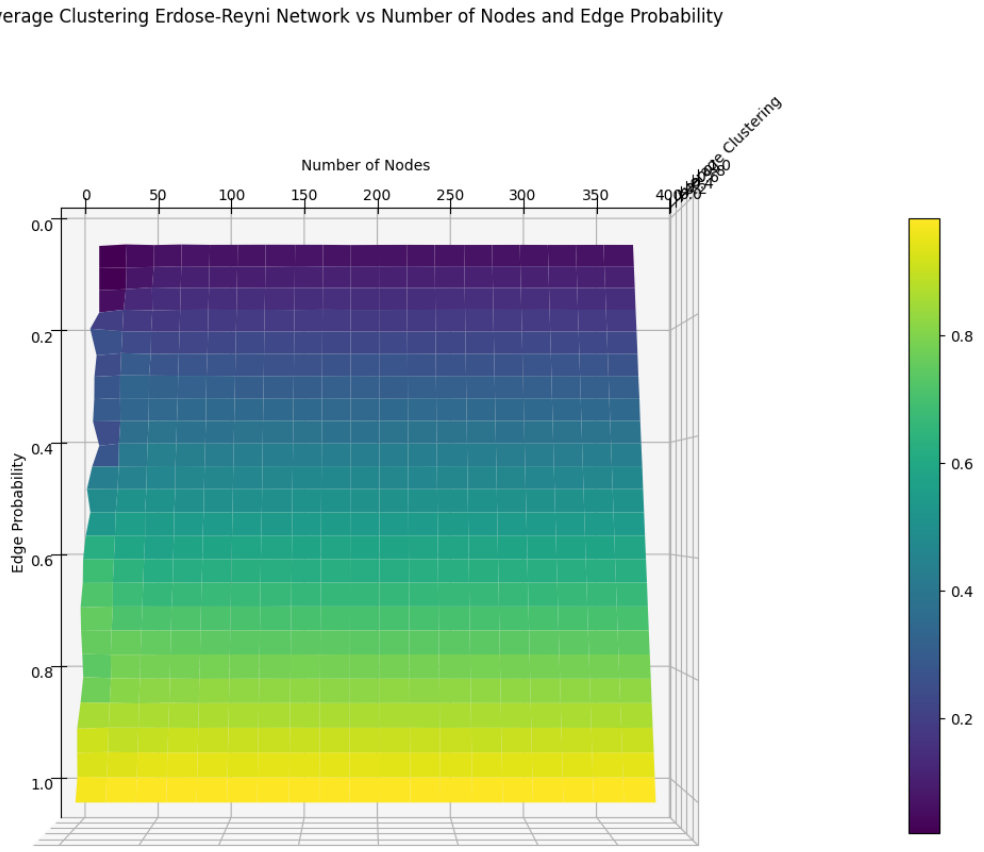

Average Clustering Erdose-Reyni Network vs Number of Nodes and Edge Probability
Number of Nodes
0
50
100
150
200
250
300
350
Edge Probability
0.0
0.2
0.4
0.6
0.8
1.0
0.8
0.6
0.4
0.2

Figure 115

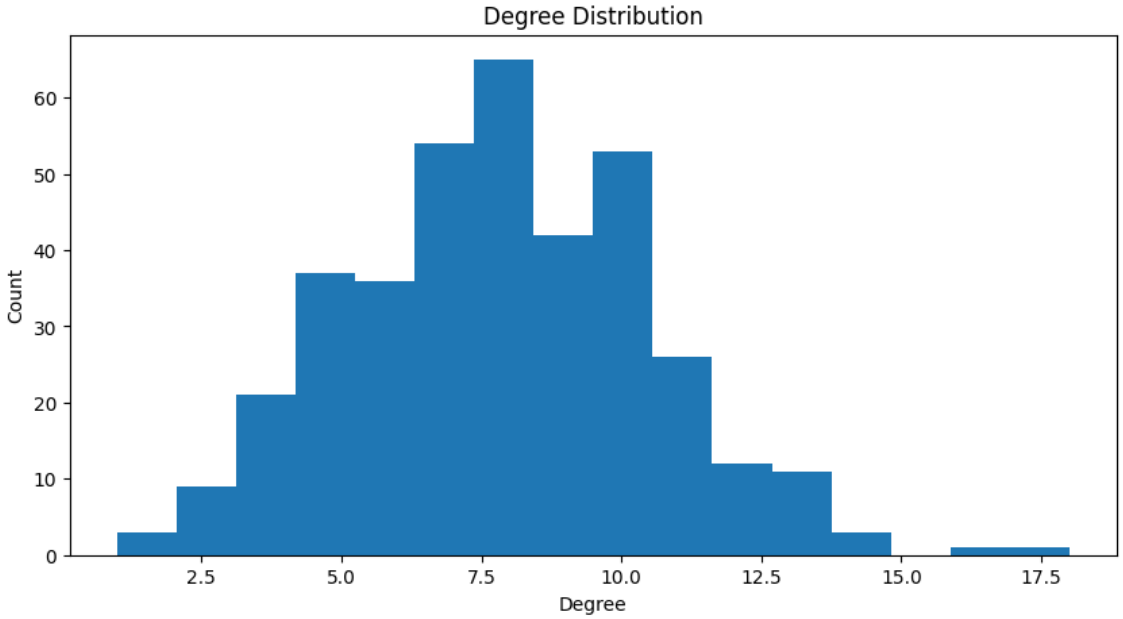


Figure 116

B.6.4 Random Regular Network

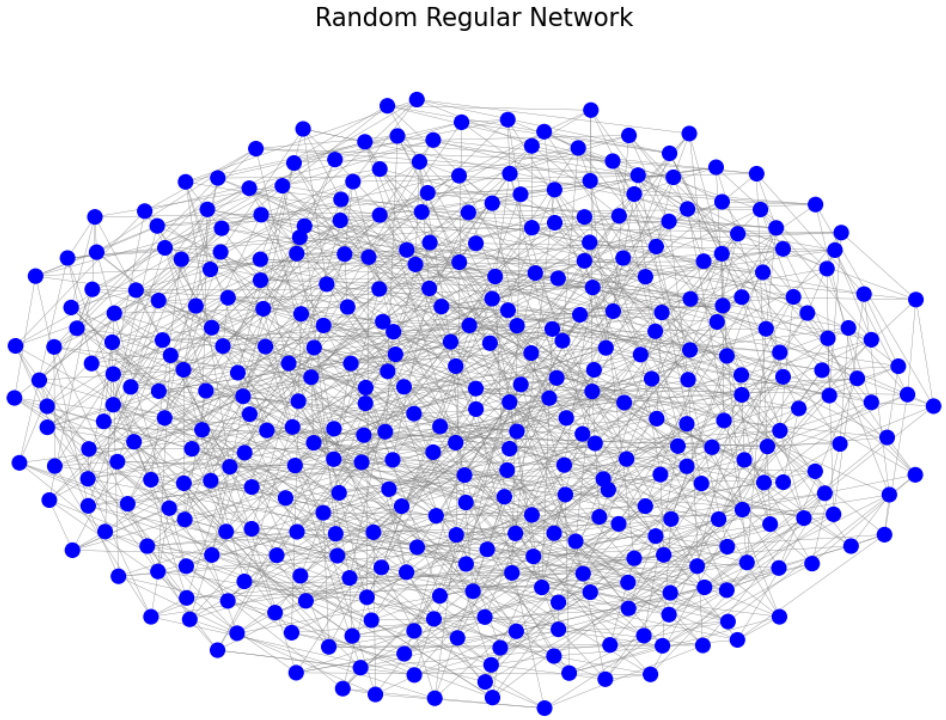


Figure 117

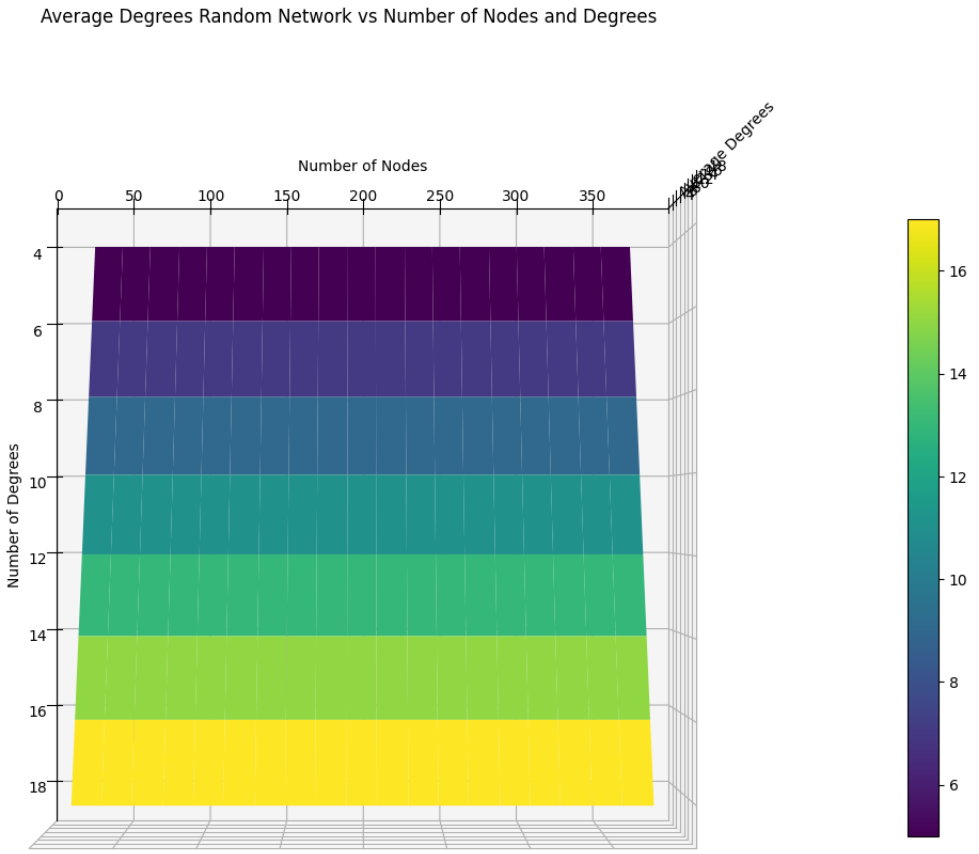

Average Degrees Random Network vs Number of Nodes and Degrees
Number of Nodes
0
50
100
150
200
250
300
350
Number of Degrees
4
6
8
10
12
14
16
18
16
14
12
10
8
6

Figure 118

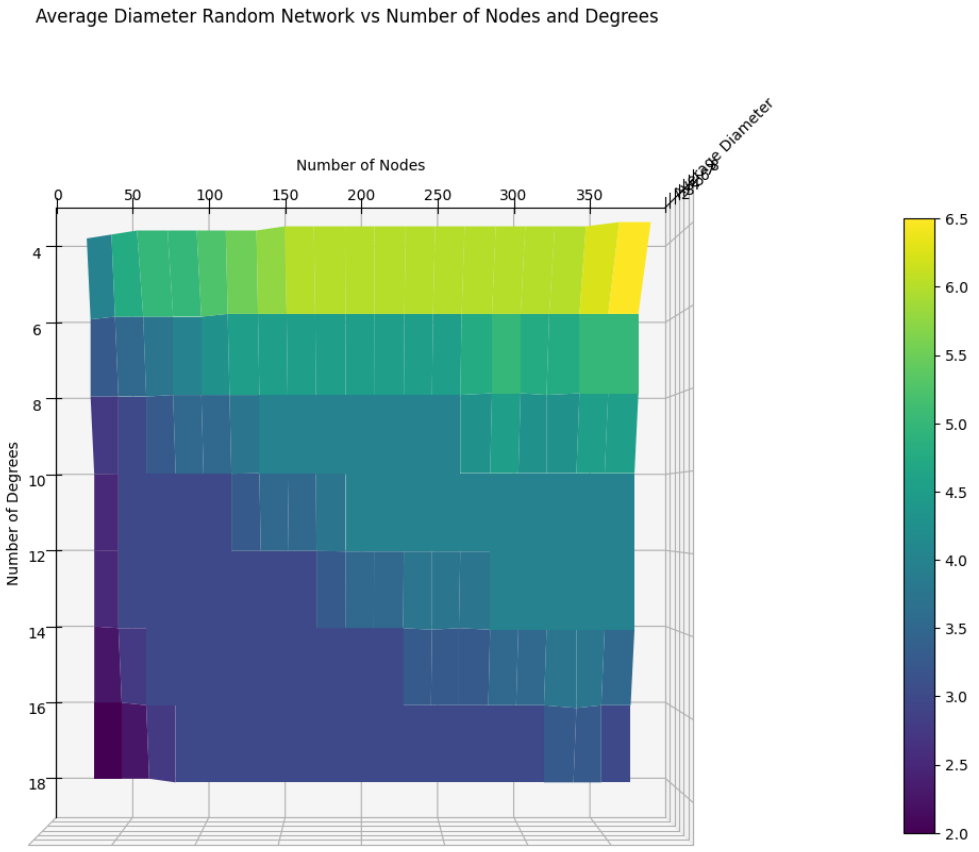

Average Diameter Random Network vs Number of Nodes and Degrees
Number of Nodes
0
50
100
150
200
250
300
350
Number of Degrees
4
6
8
10
12
14
16
18
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0

Figure 119

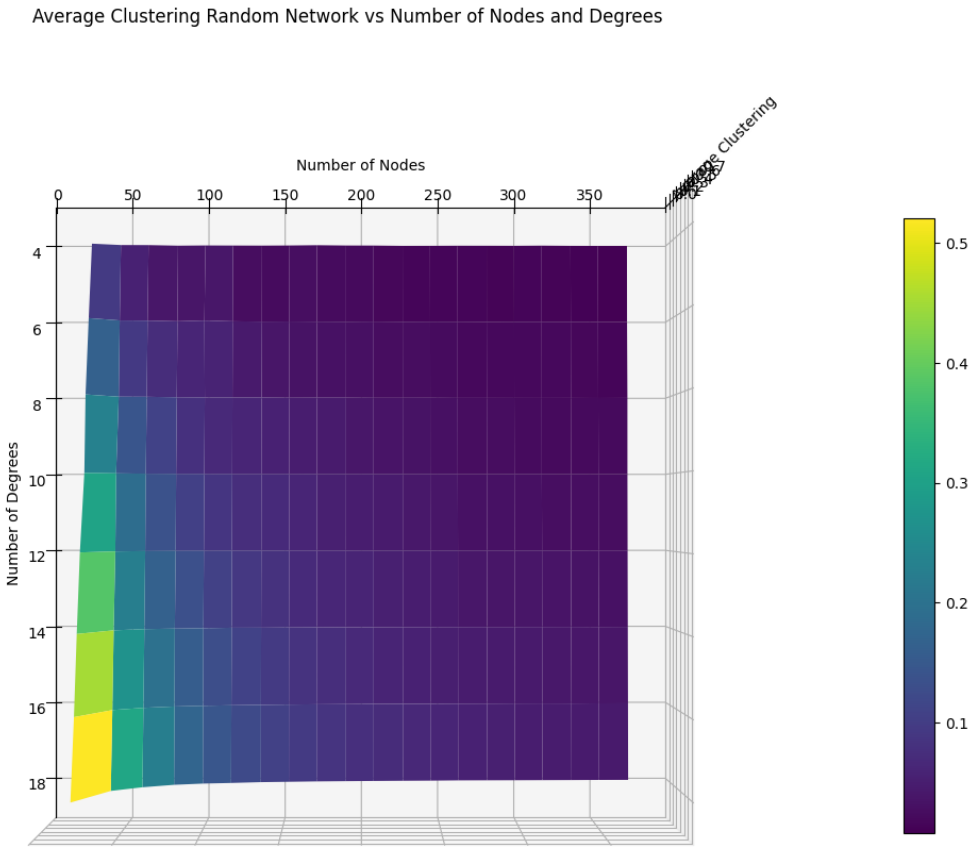

Average Clustering Random Network vs Number of Nodes and Degrees
Number of Nodes
0
50
100
150
200
250
300
350
Number of Degrees
4
6
8
10
12
14
16
18
Average Clustering
0.5
0.4
0.3
0.2
0.1

Figure 120

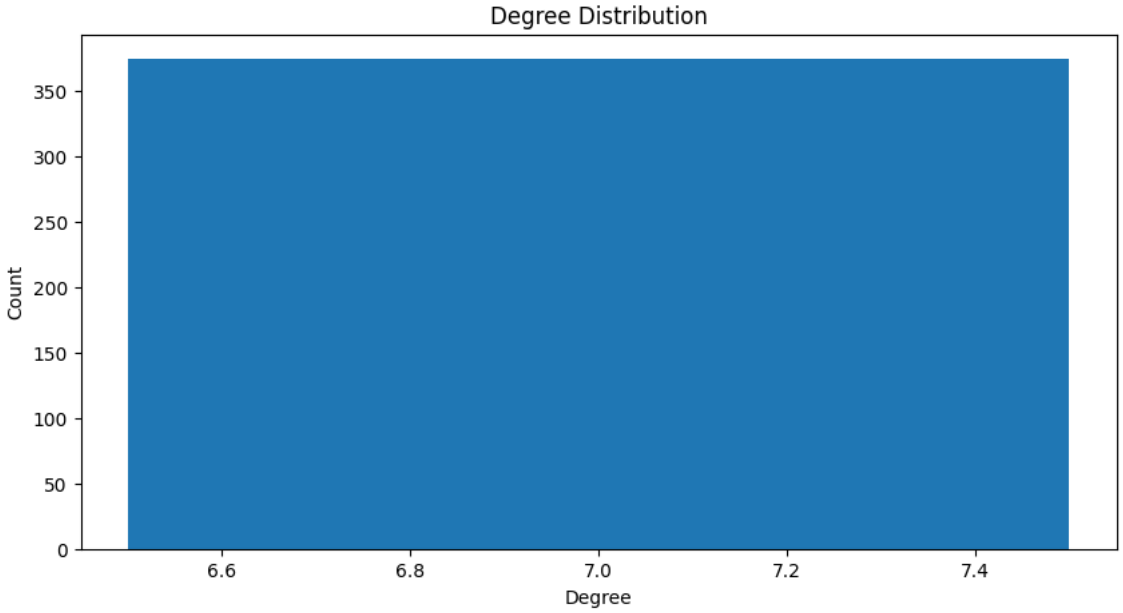


Figure 121

B.6.5 Sociopatterns Network

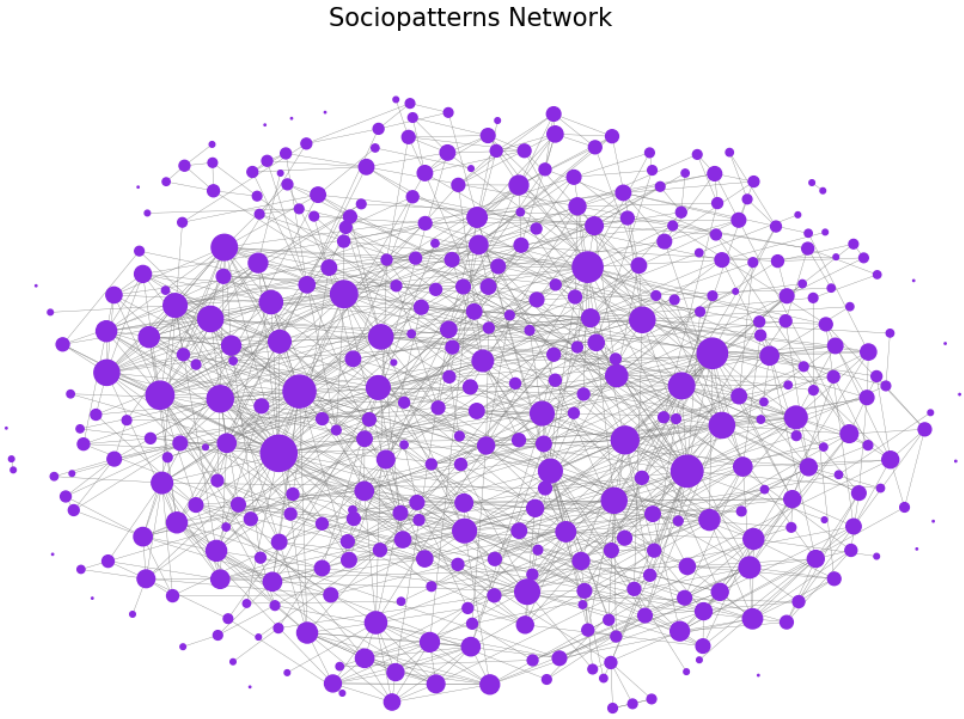


Figure 122

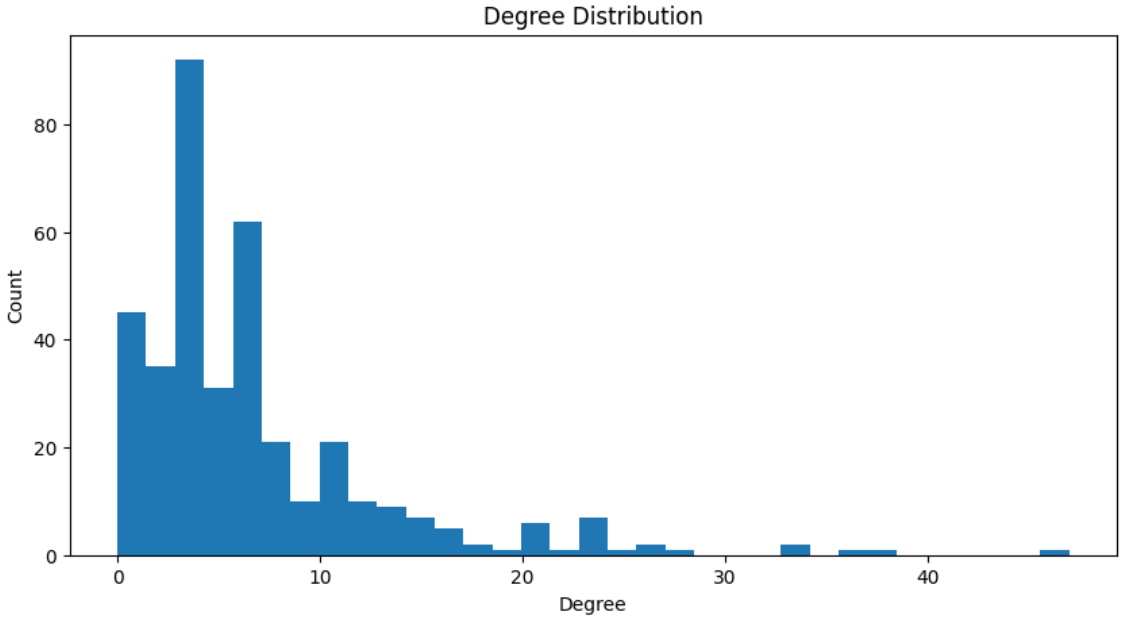


Figure 123

B.7 Network Comparisons

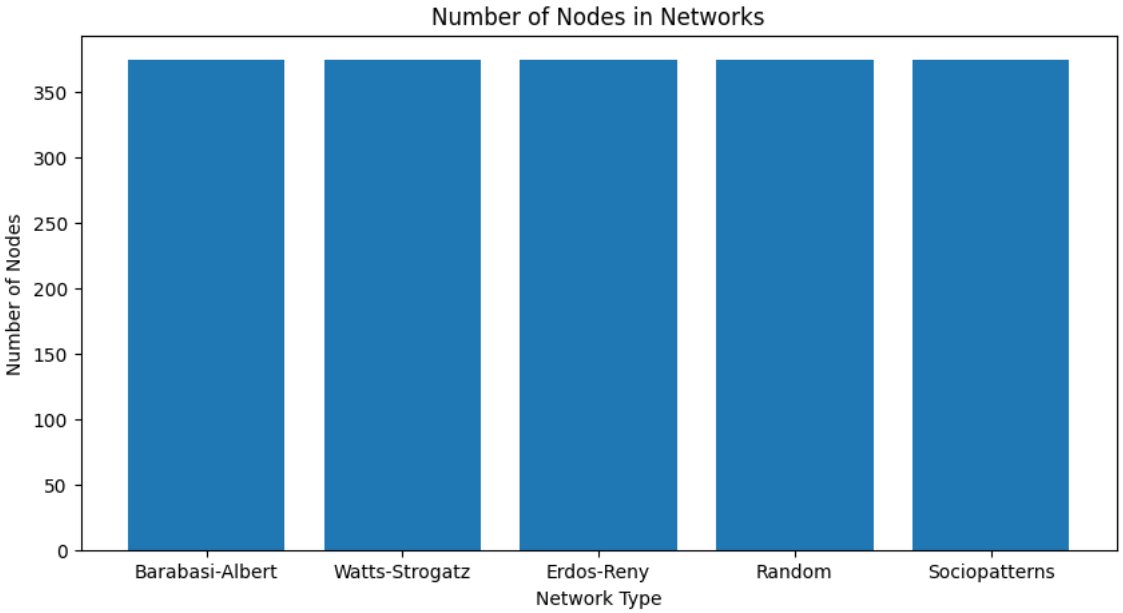


Figure 124

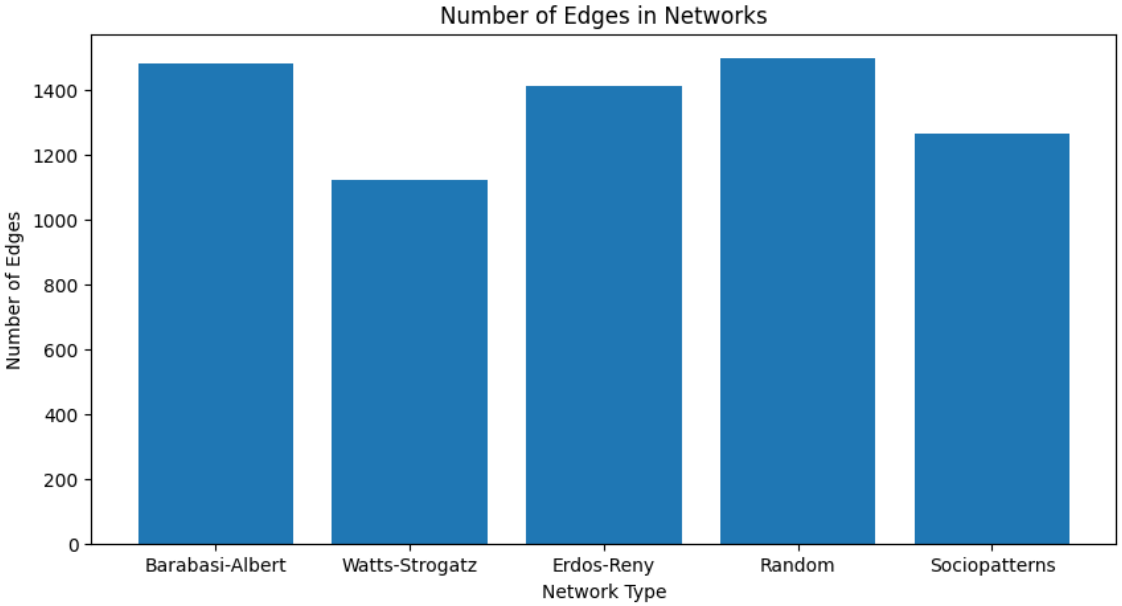


Figure 125

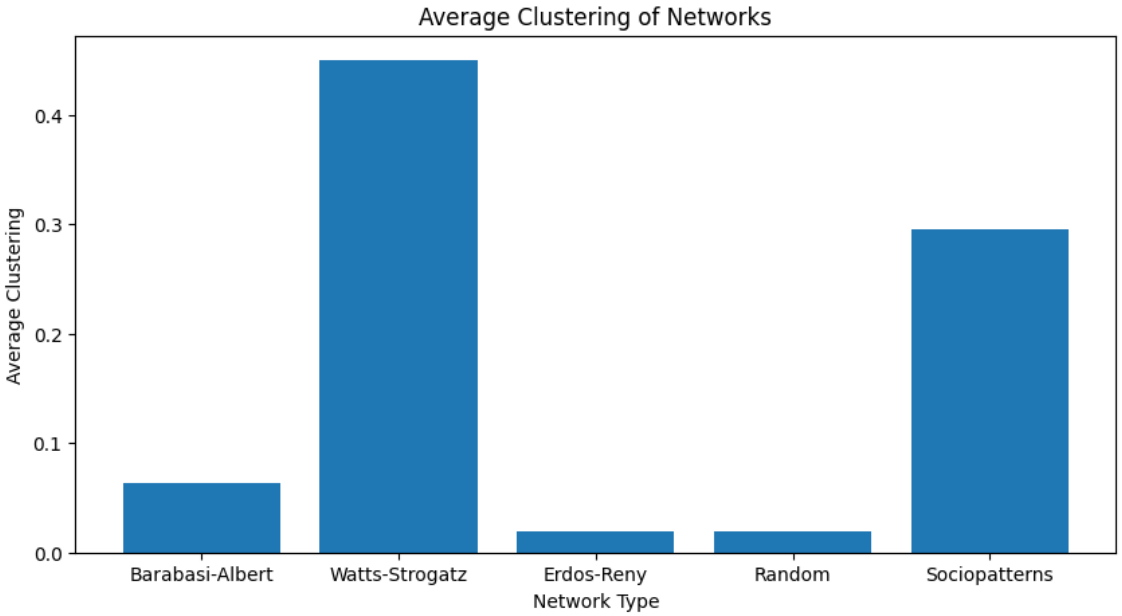


Figure 126

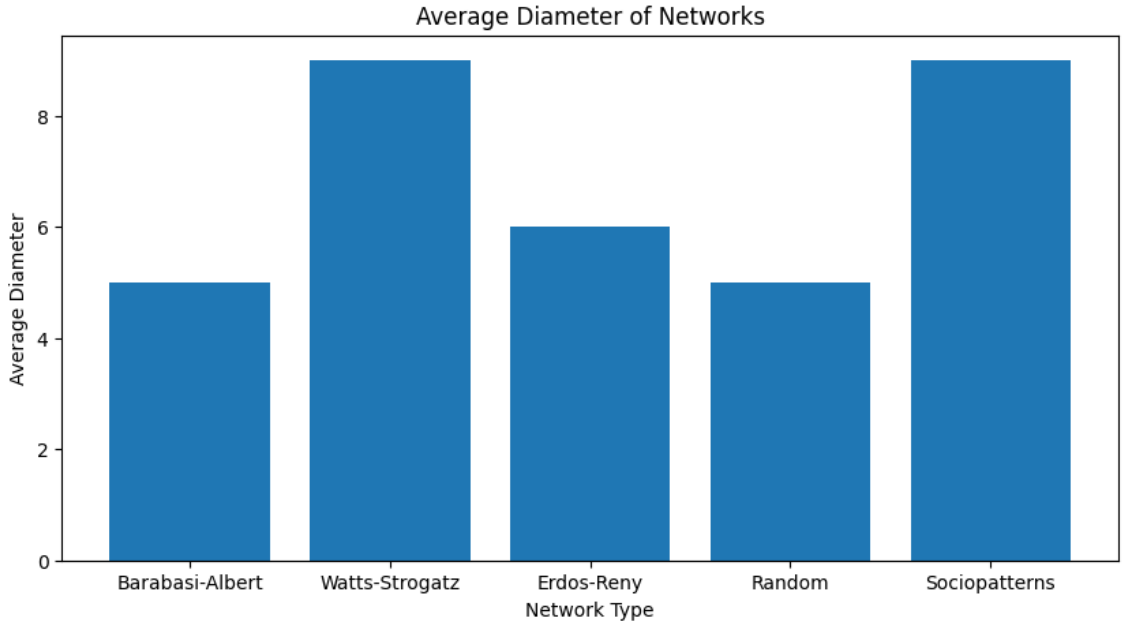


Figure 127

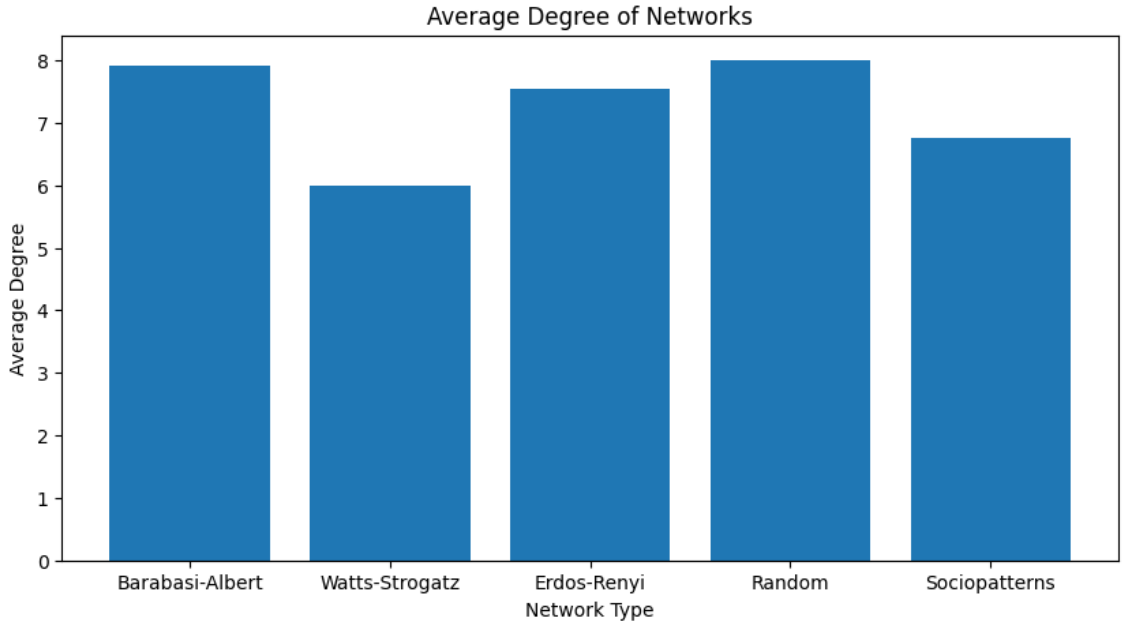


Figure 128

B.8 Infected Networks

To view animated network infection spread videos for different network models and parameters, refer to the project drive.

B.8.1 Infected Network SIR

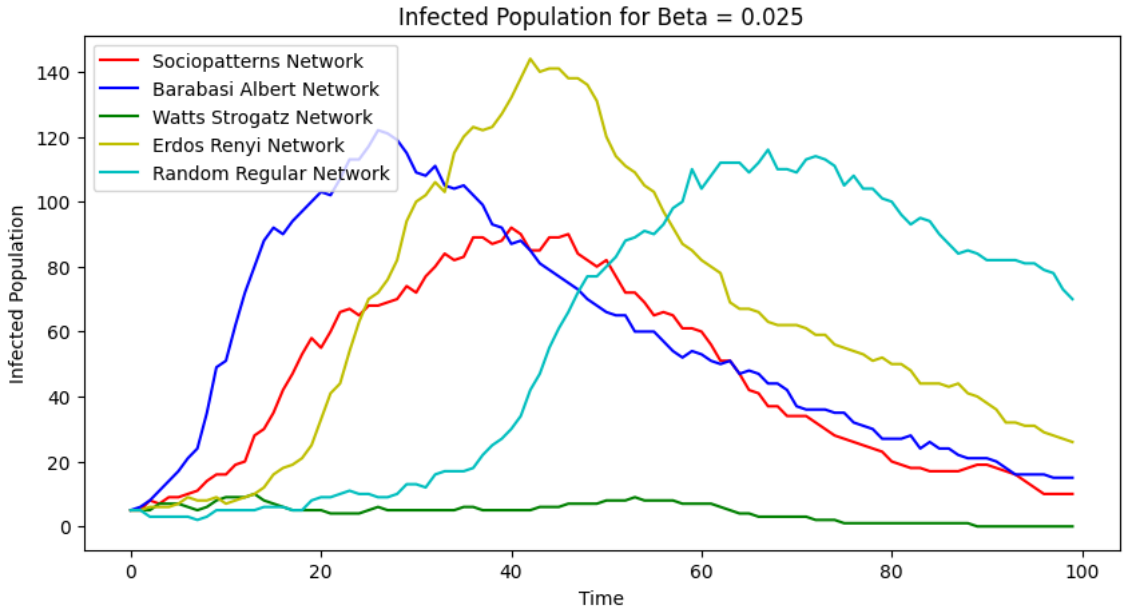


Figure 129

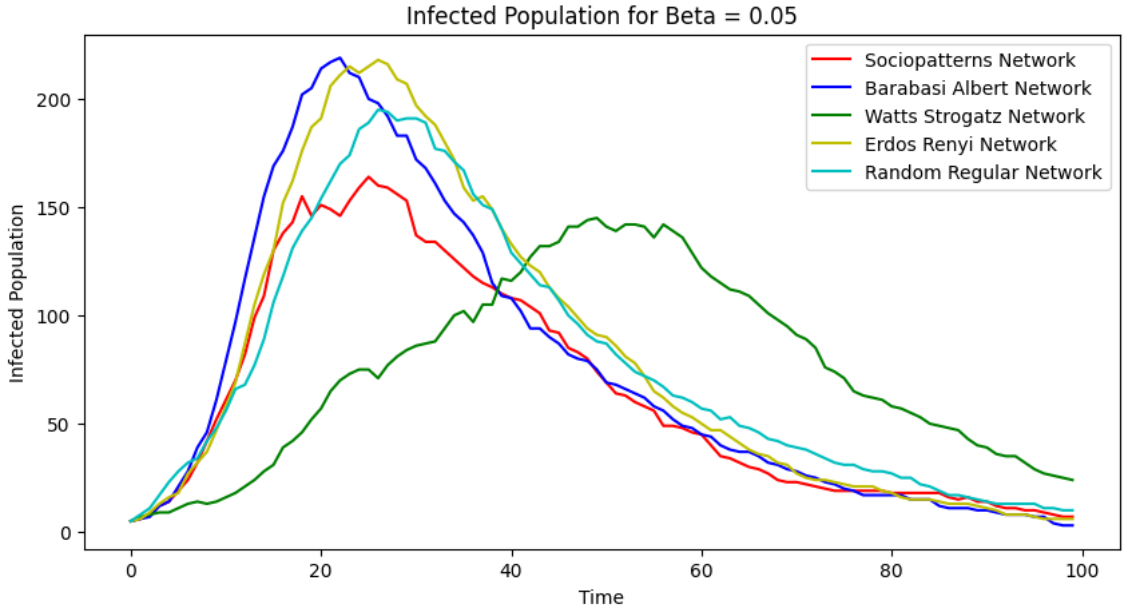


Figure 130

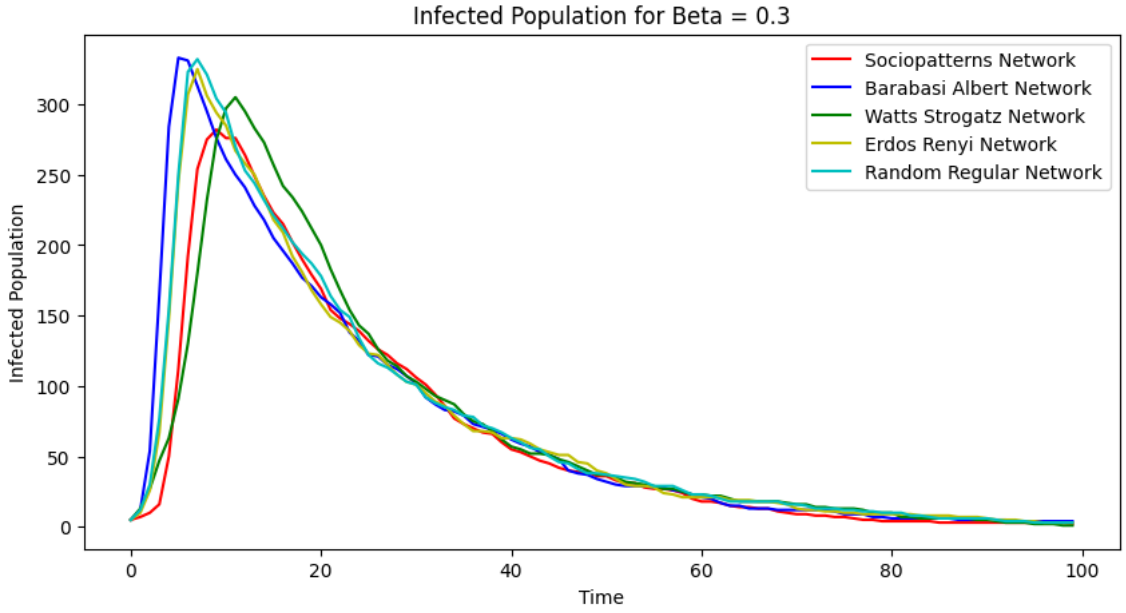


Figure 131

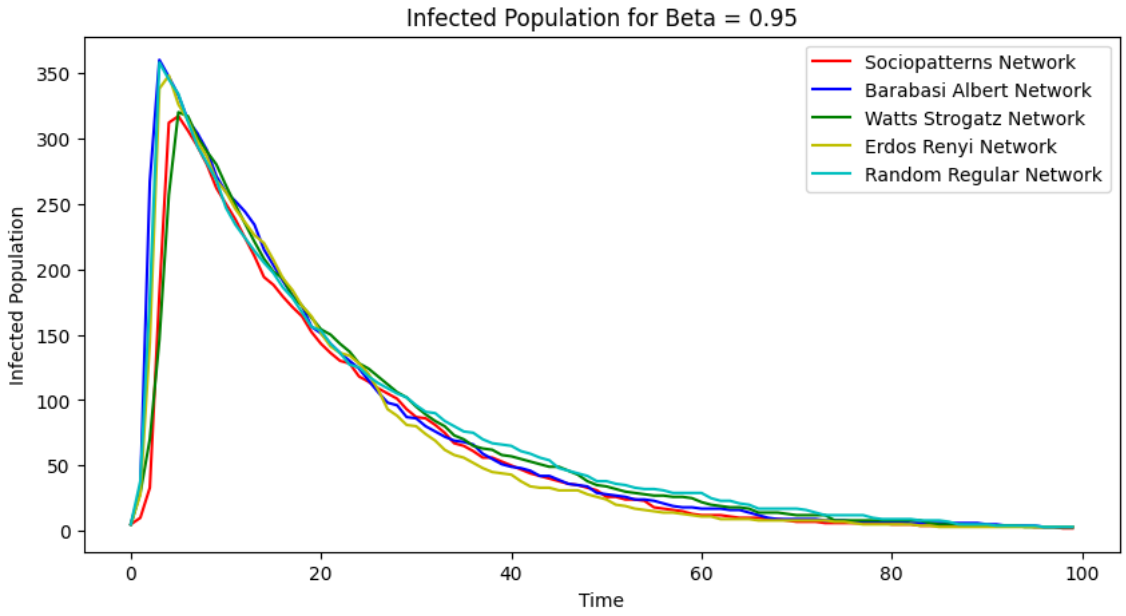


Figure 132

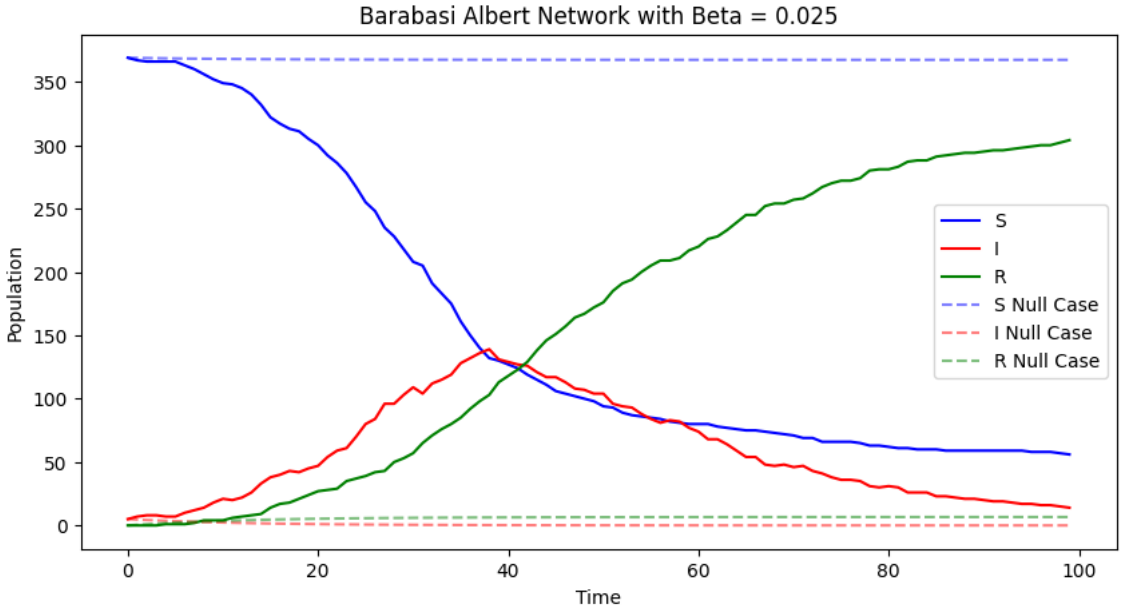


Figure 133

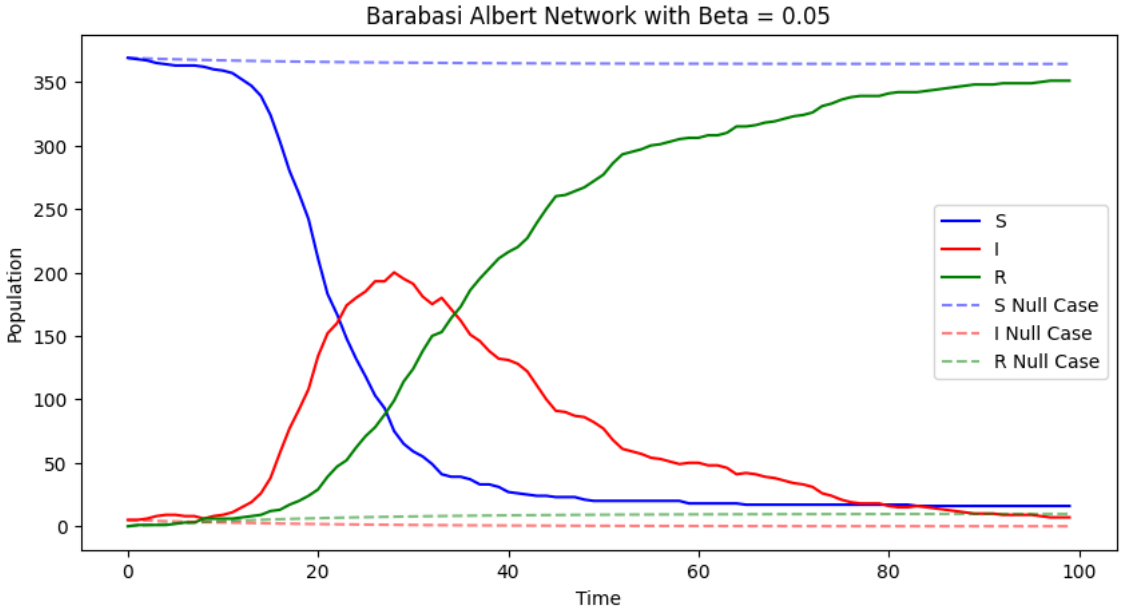


Figure 134

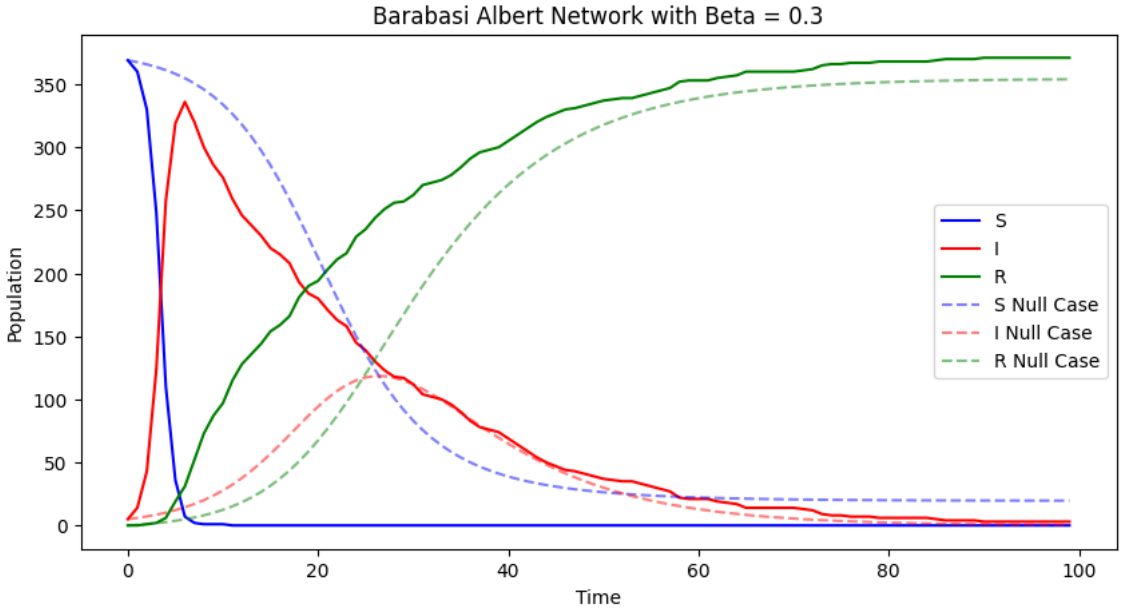


Figure 135

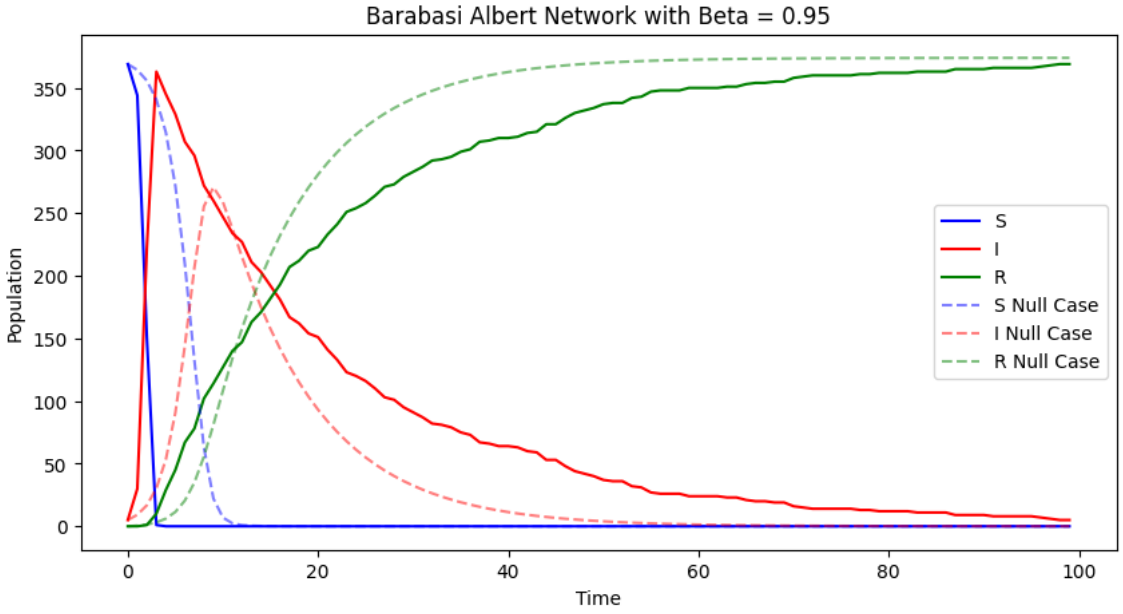


Figure 136

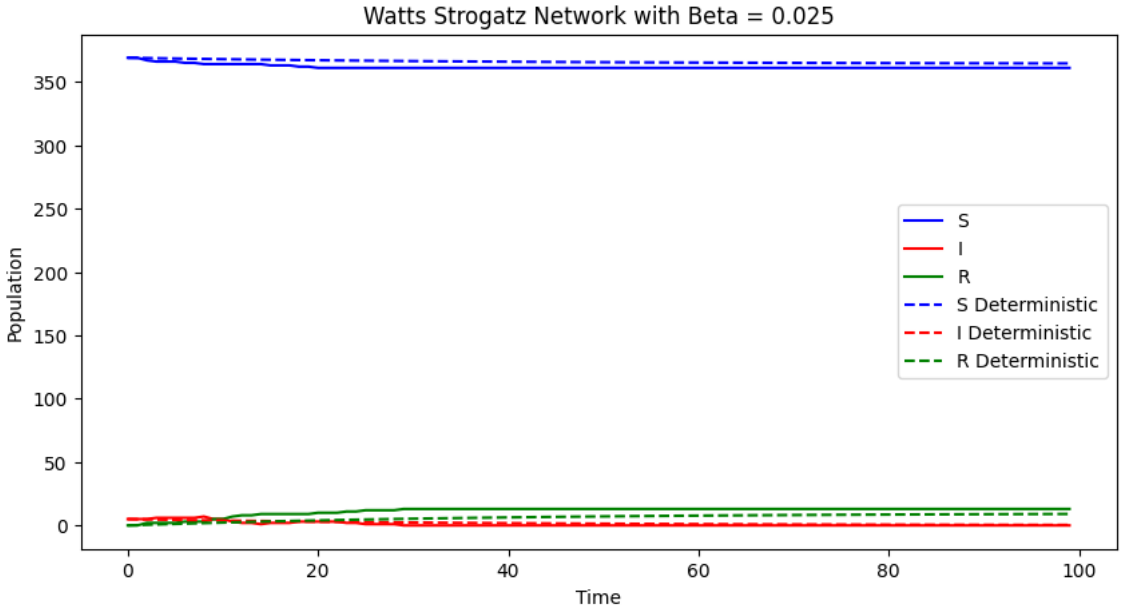


Figure 137

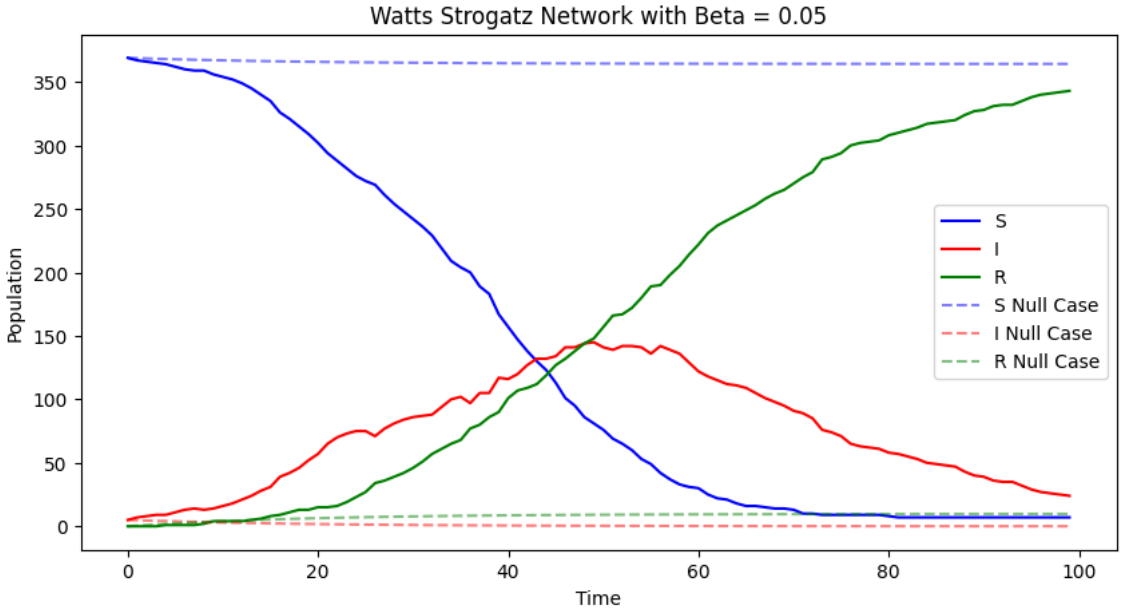


Figure 138

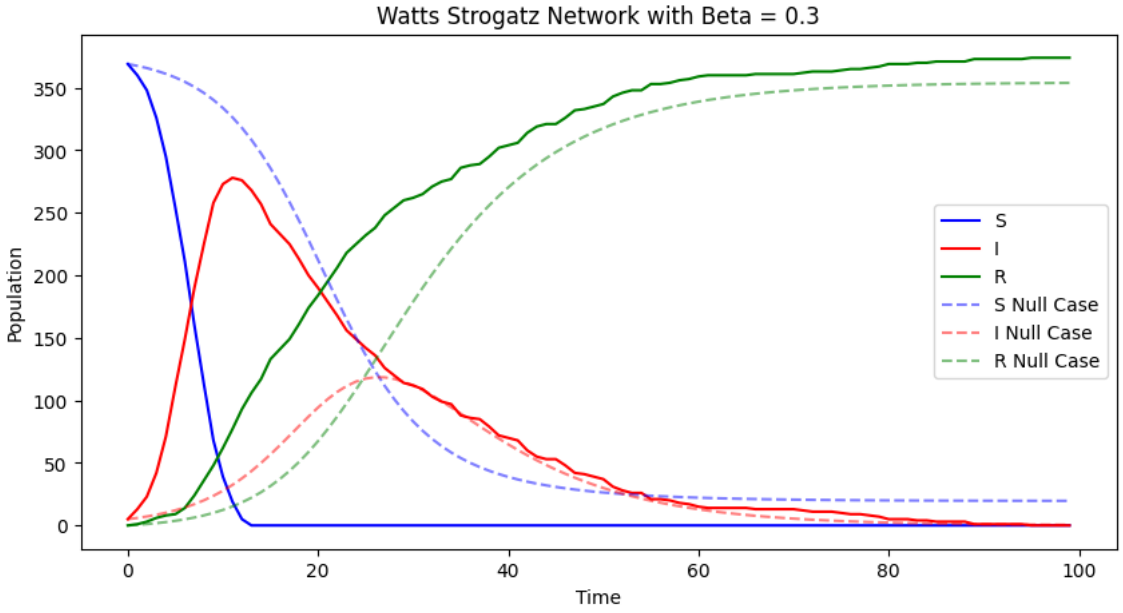


Figure 139

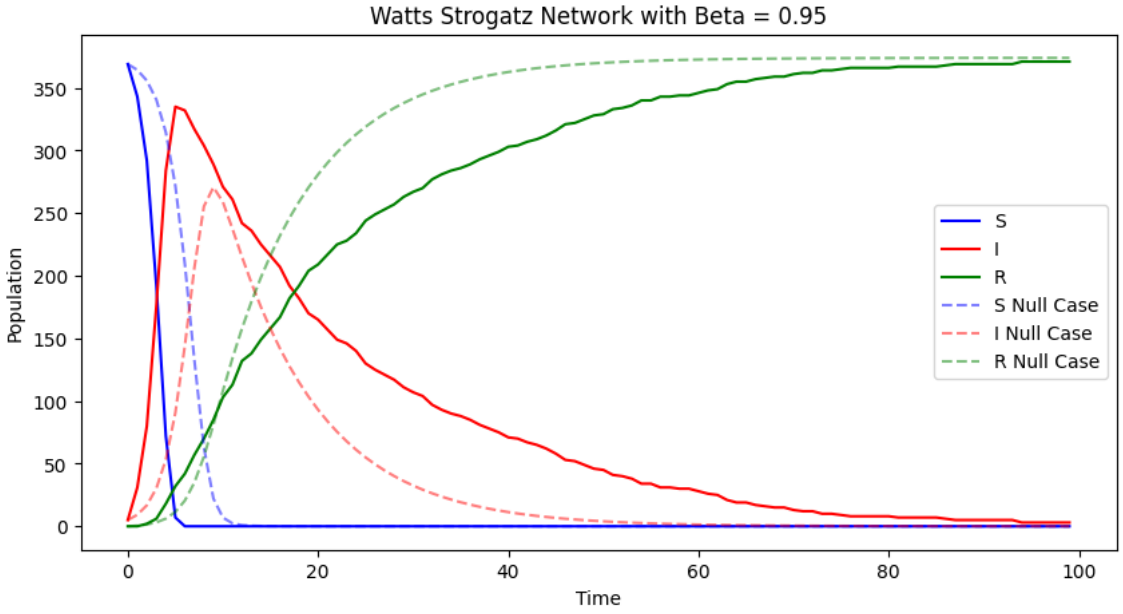


Figure 140

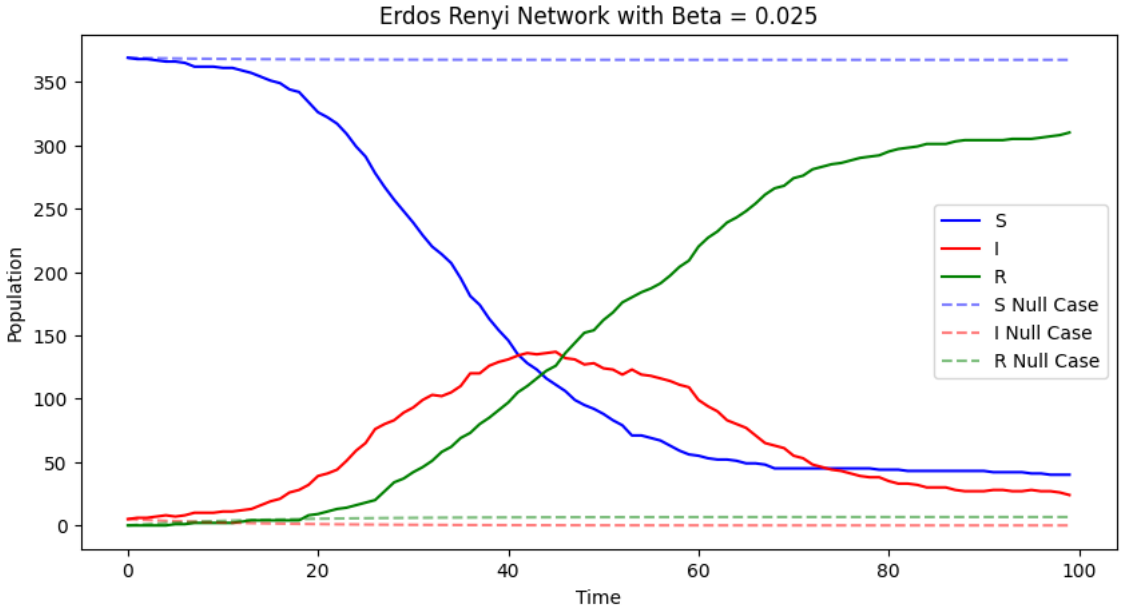


Figure 141

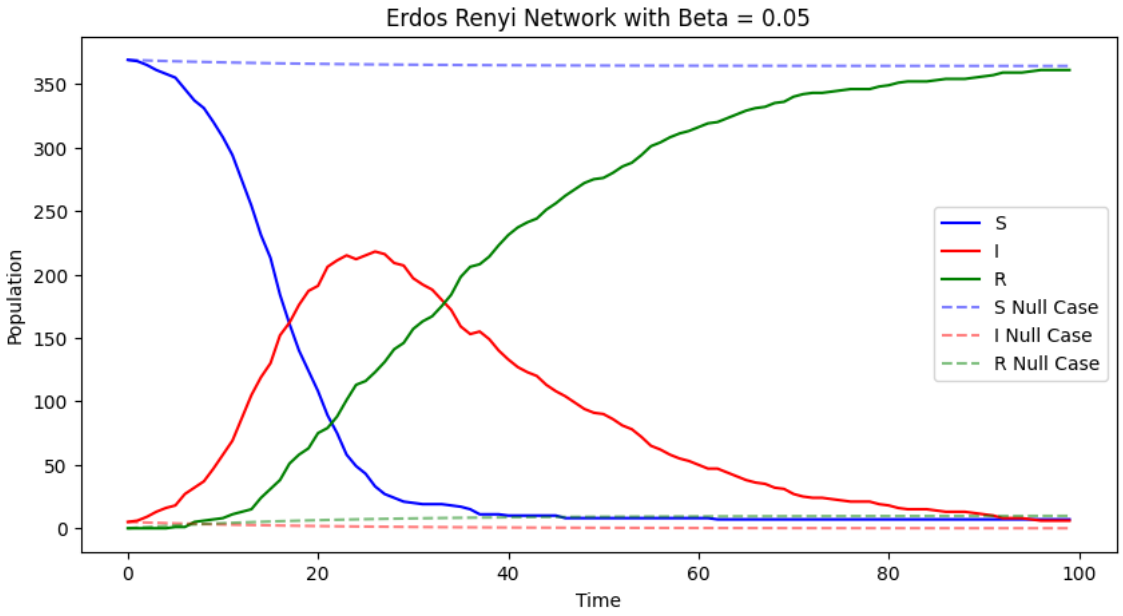


Figure 142

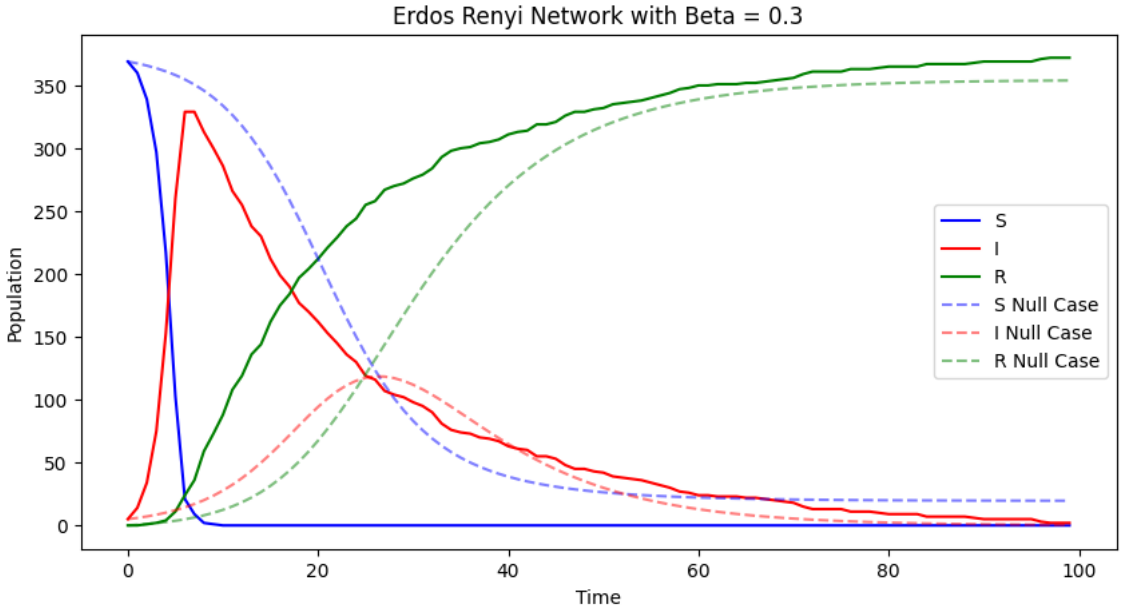


Figure 143

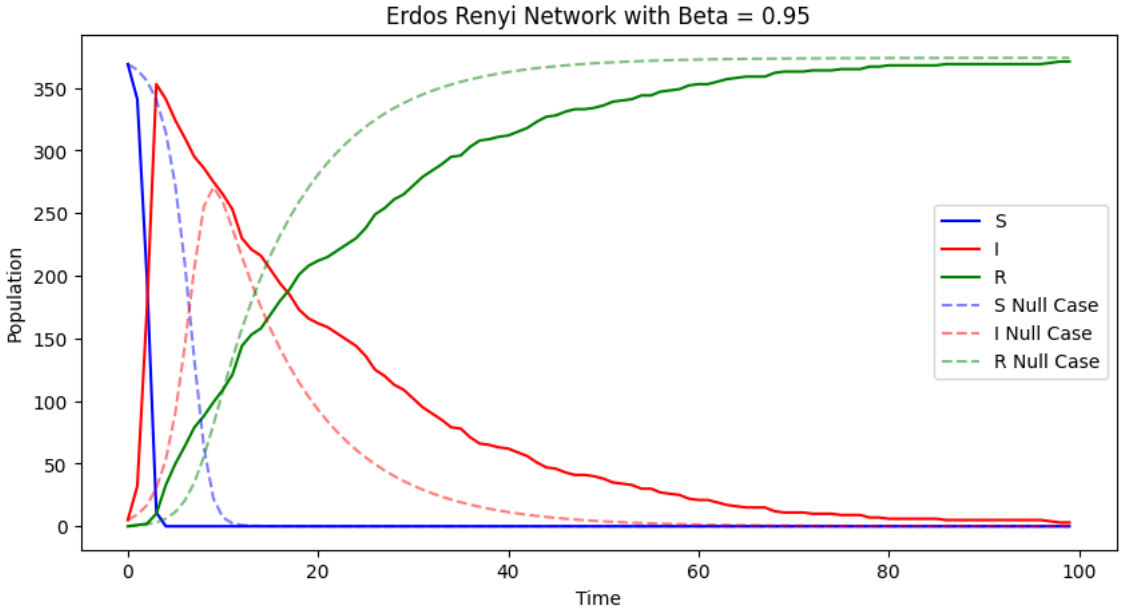


Figure 144

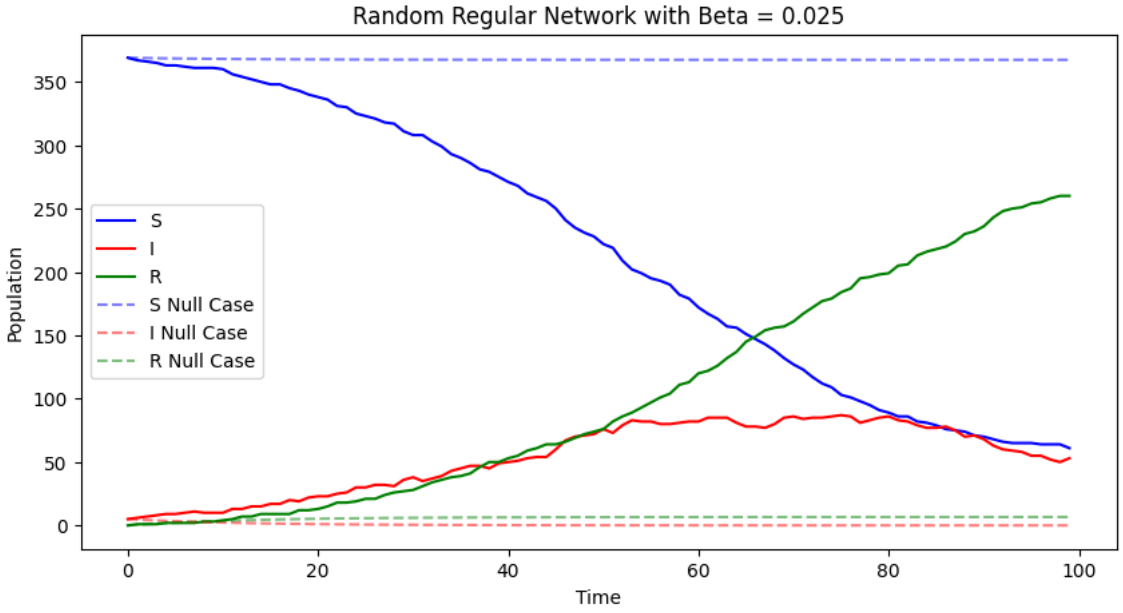


Figure 145

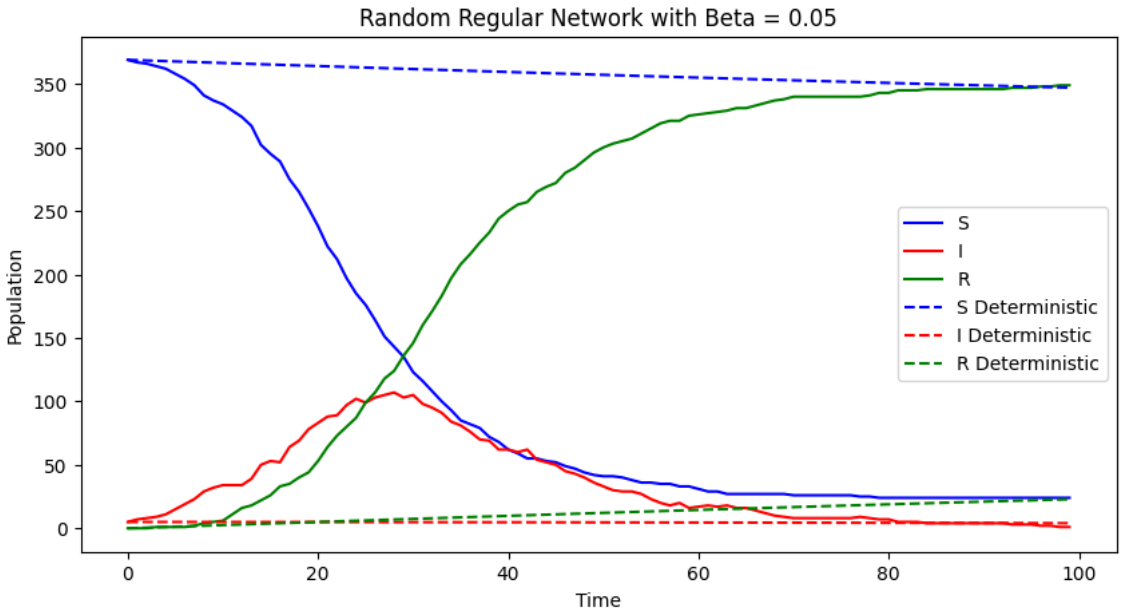


Figure 146

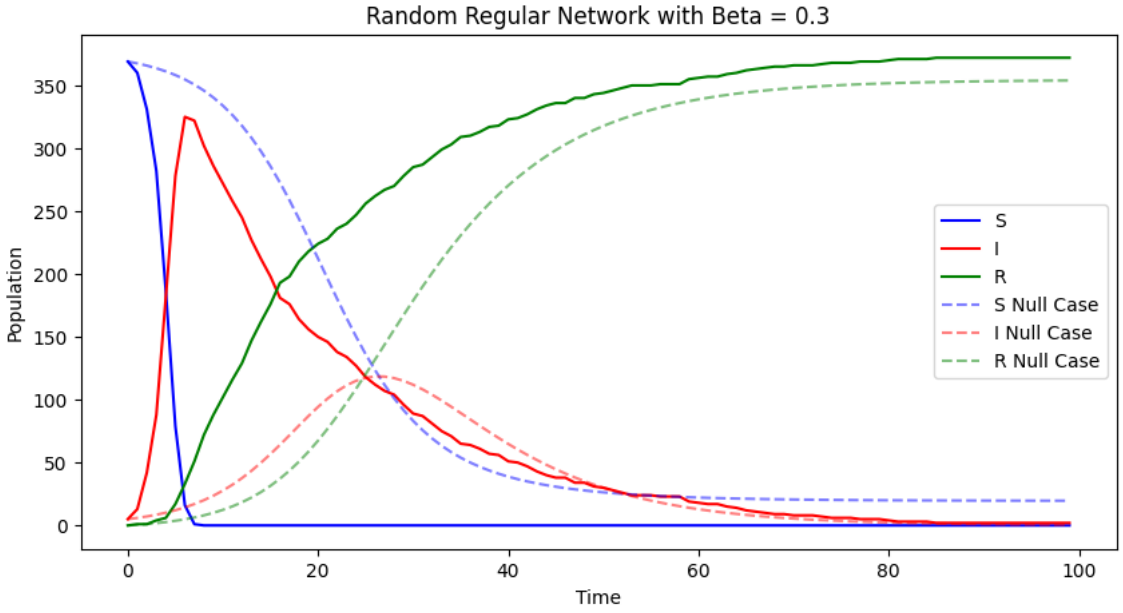


Figure 147

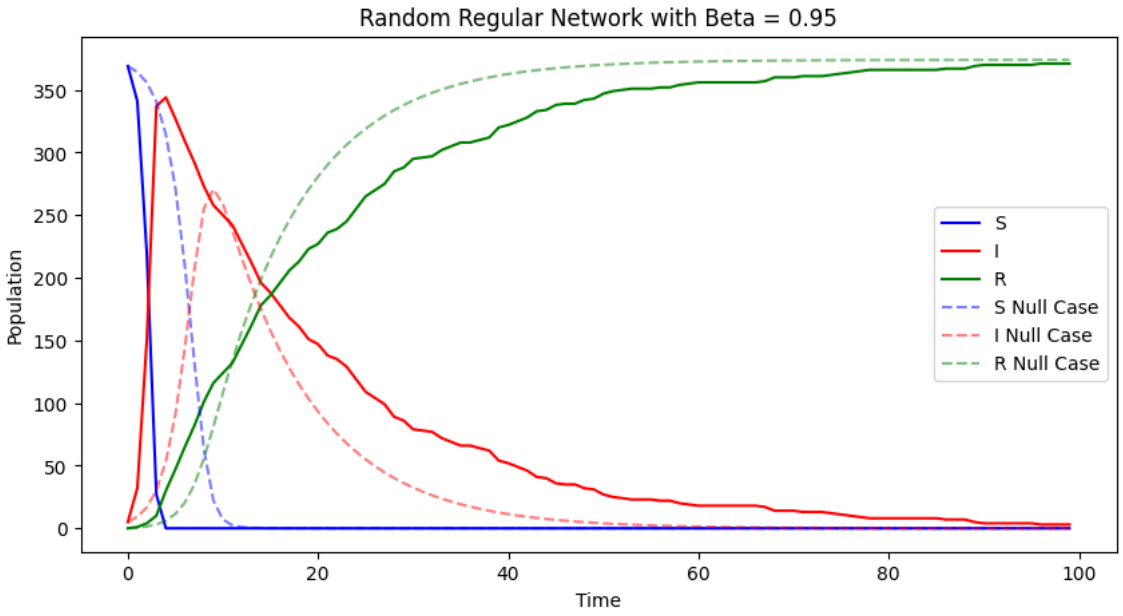


Figure 148

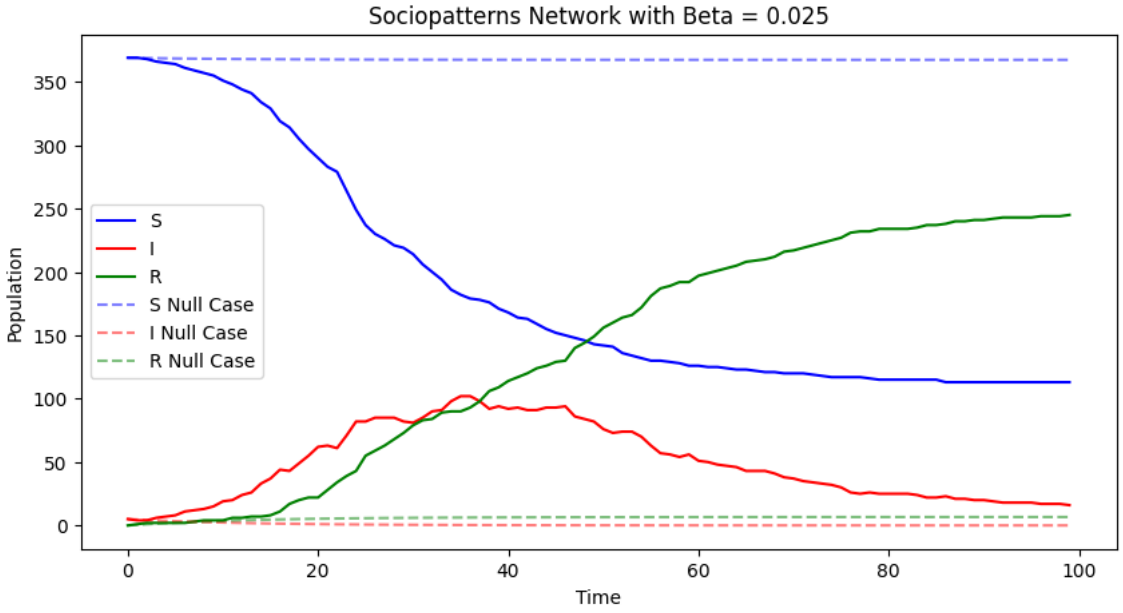


Figure 149

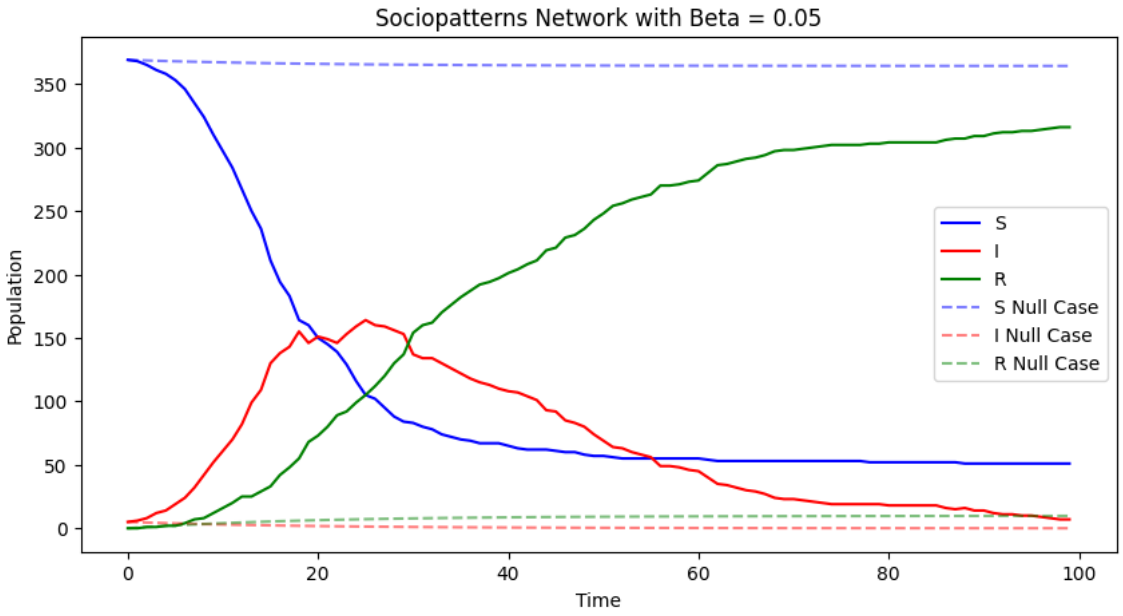


Figure 150

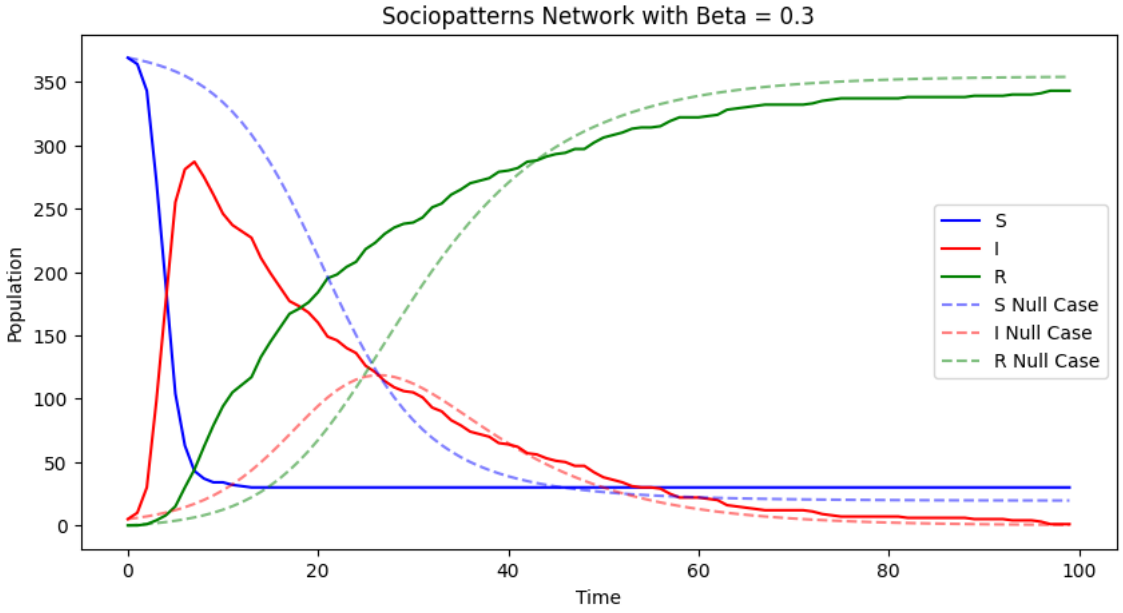


Figure 151

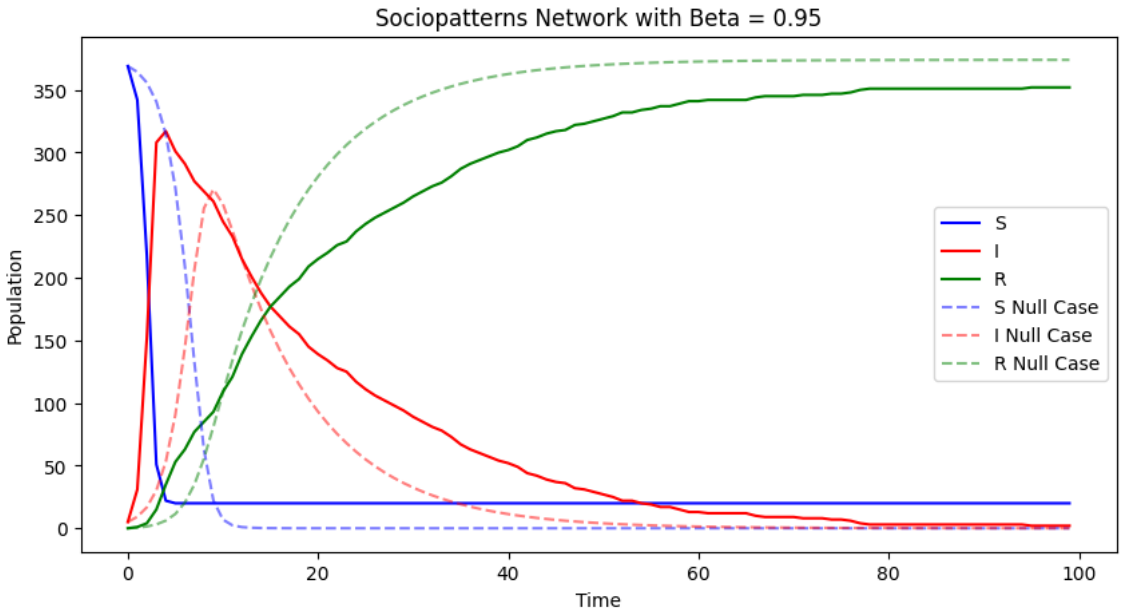


Figure 152

B.8.2 Mean

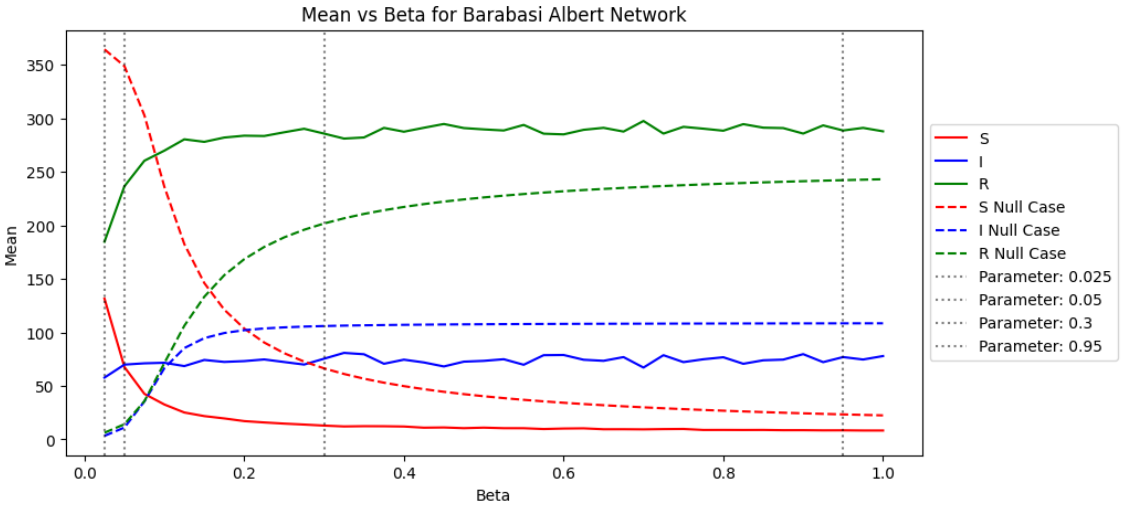


Figure 153

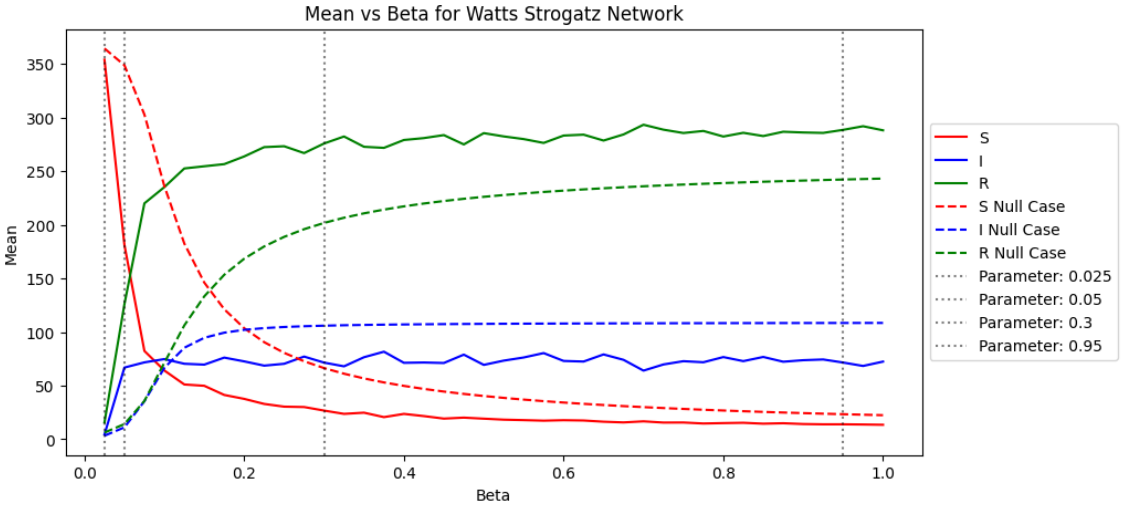


Figure 154

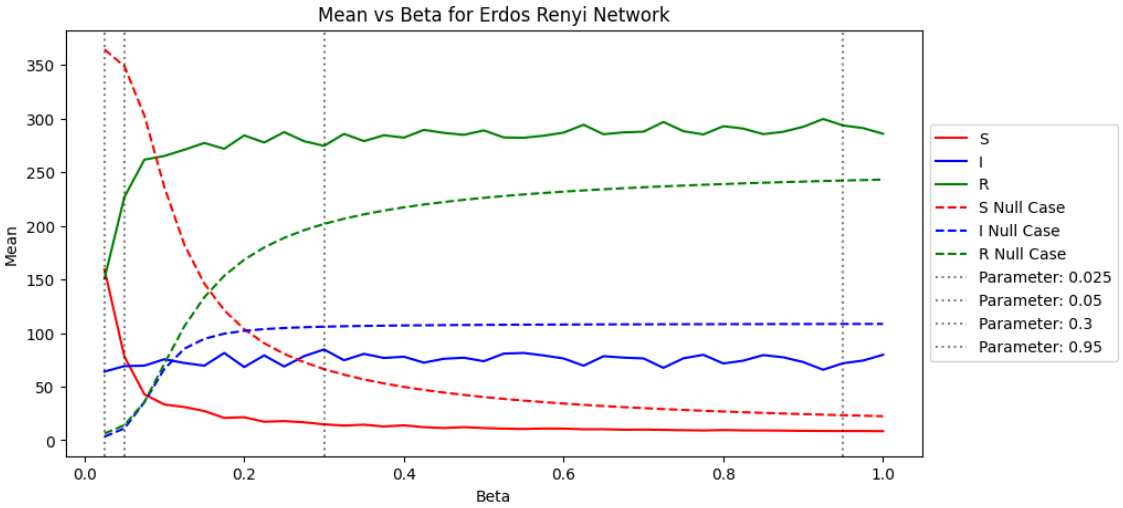


Figure 155

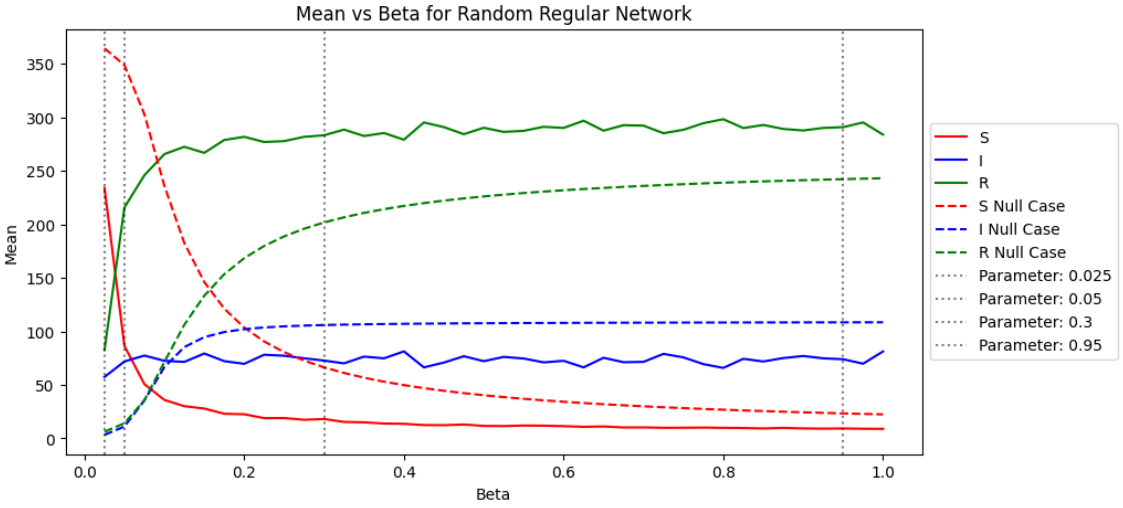


Figure 156

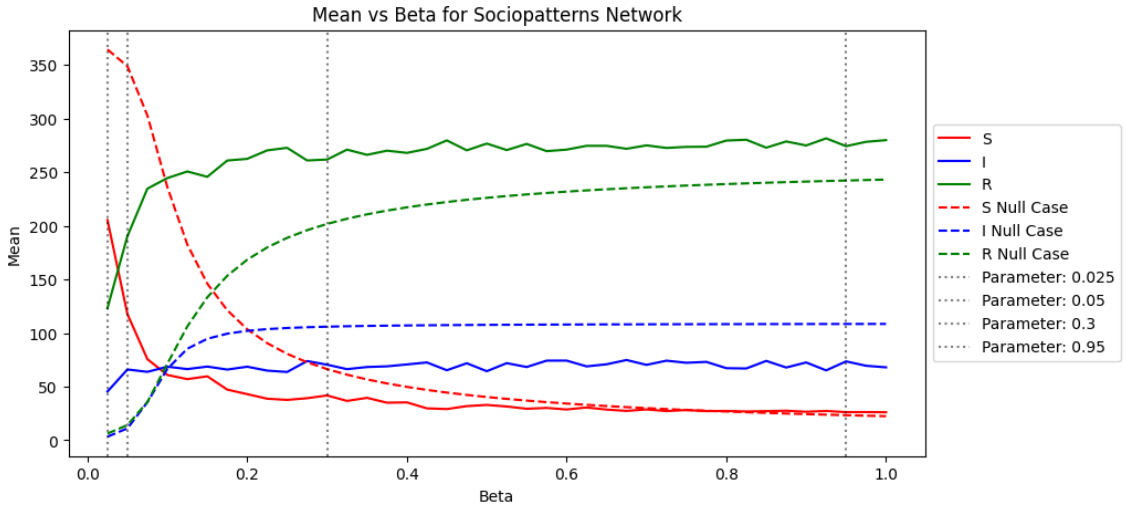


Figure 157

B.8.3 Standard Deviation

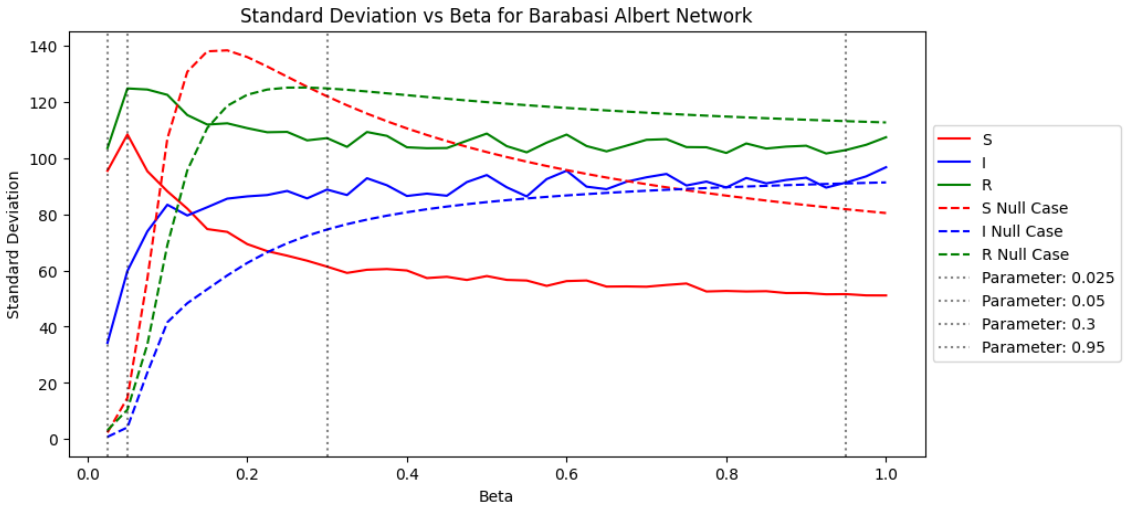


Figure 158

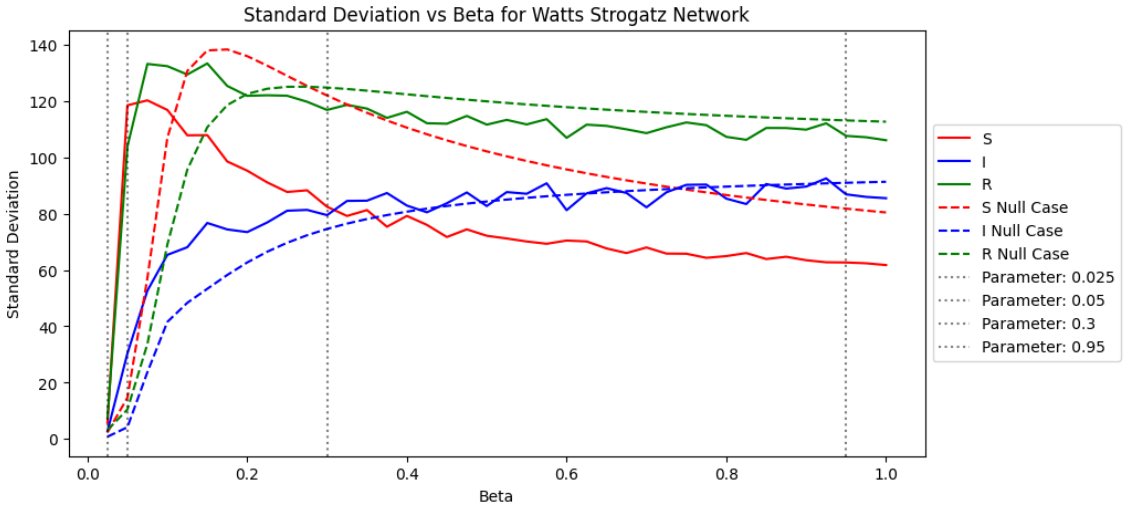


Figure 159

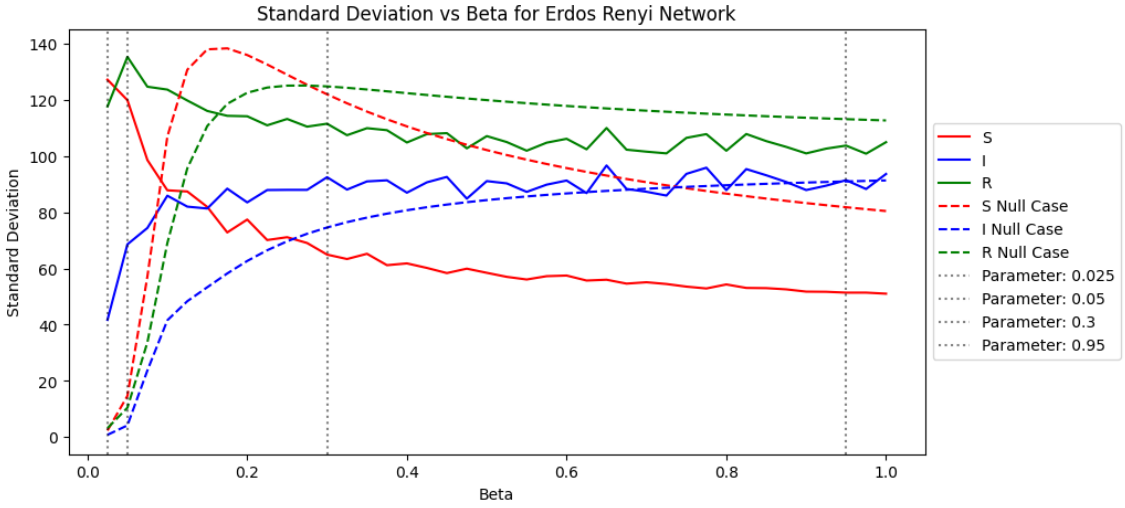


Figure 160

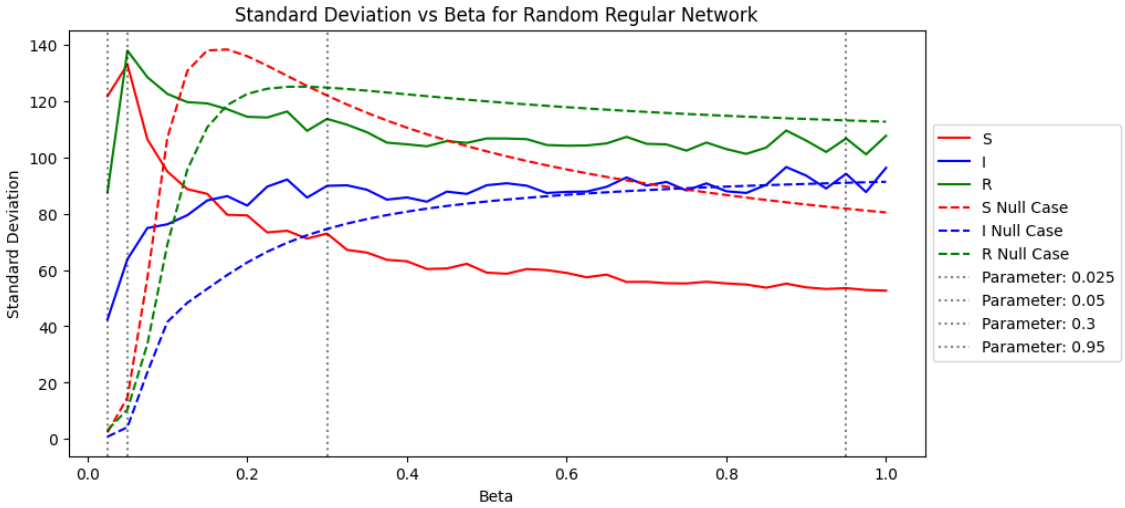


Figure 161

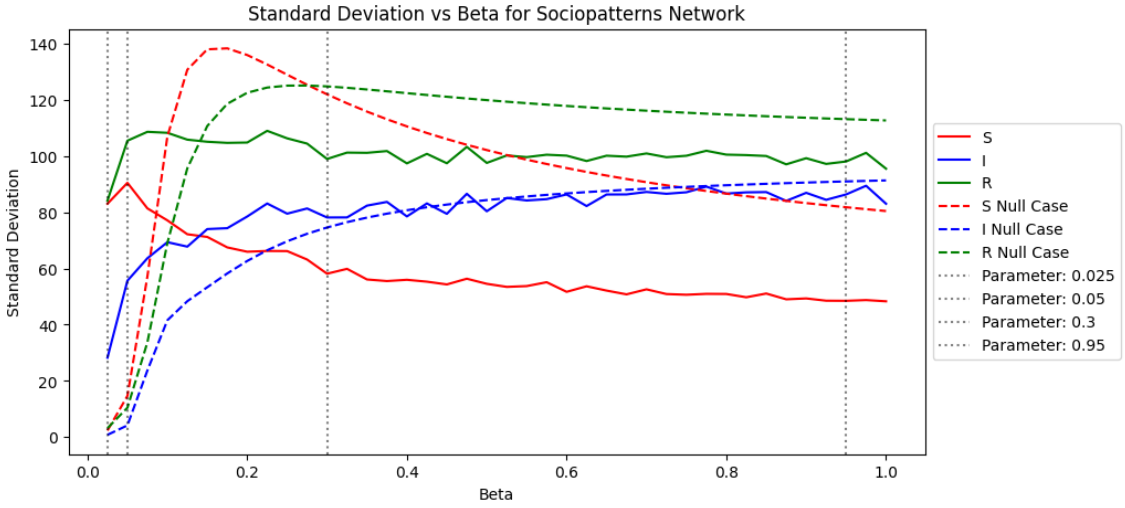


Figure 162

B.8.4 Covariance

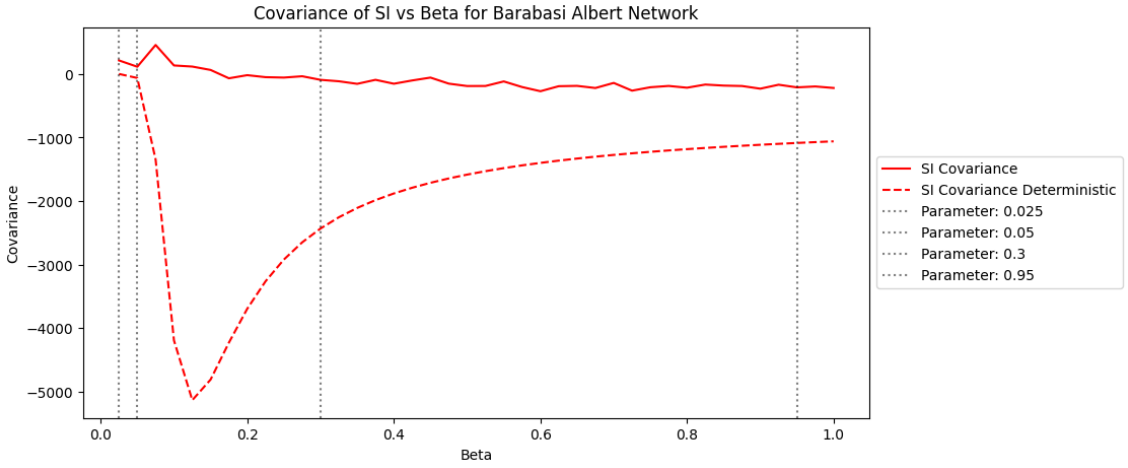


Figure 163

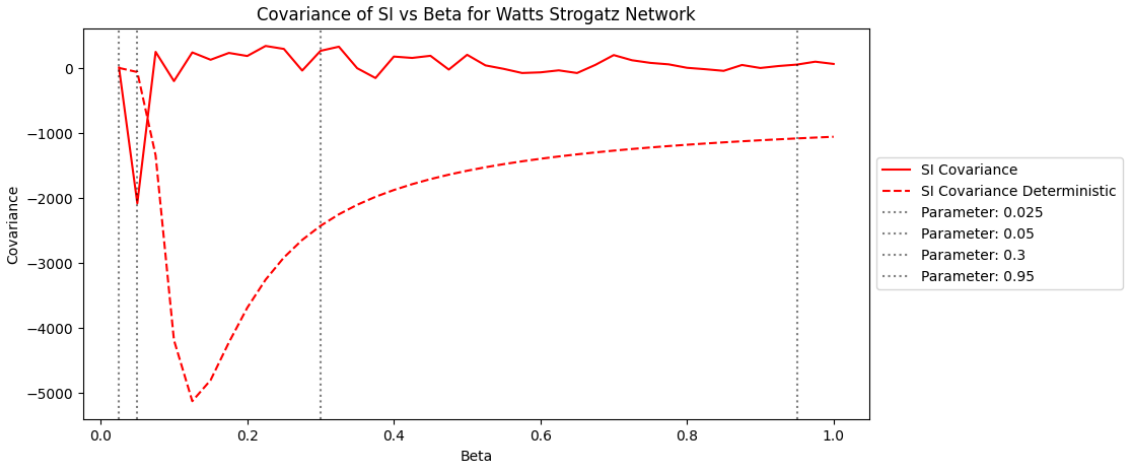


Figure 164

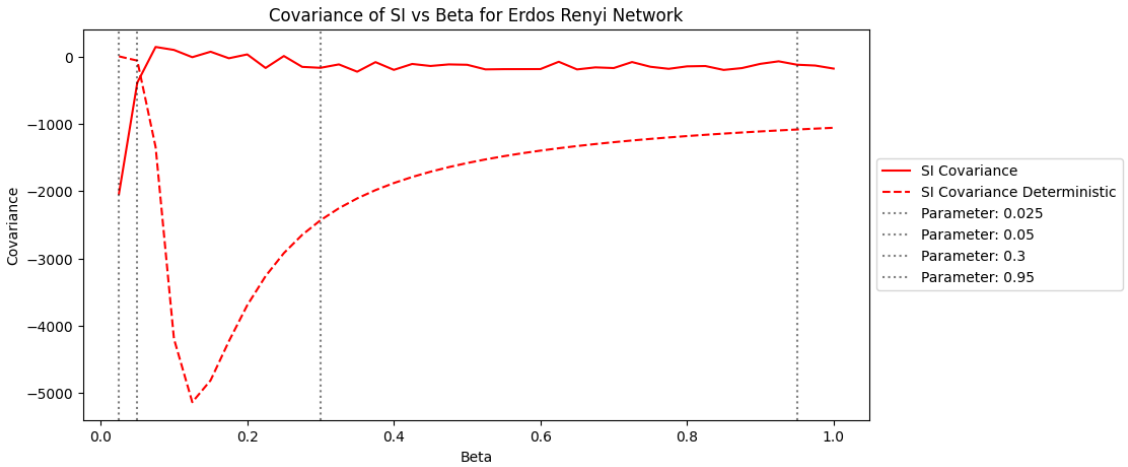


Figure 165

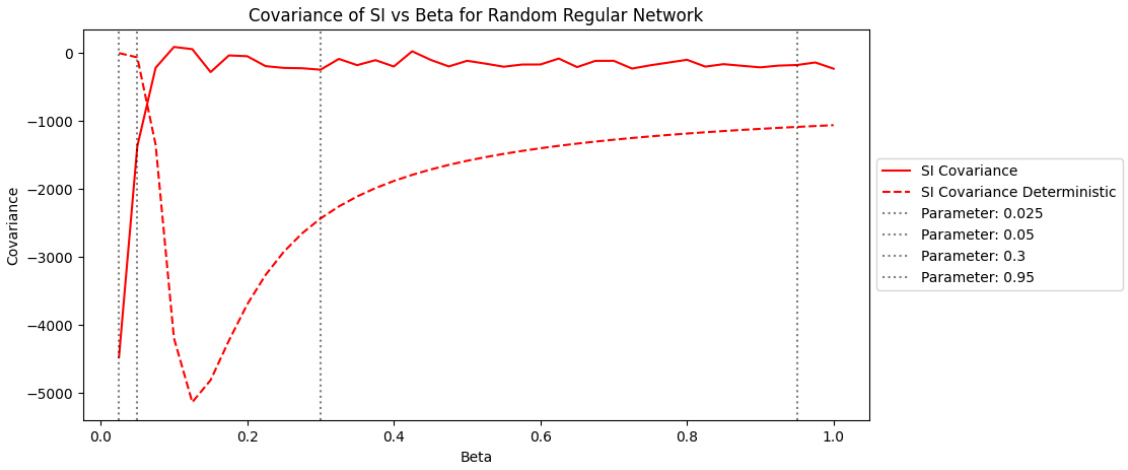


Figure 166

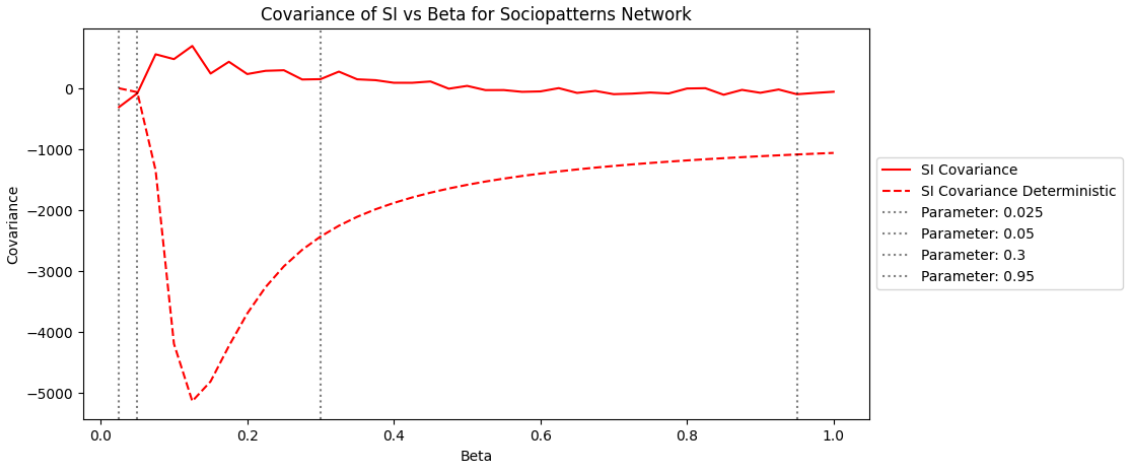


Figure 167

B.9 Vaccination Strategies

To view animated network infection spread videos for different network models and parameters, refer to the project drive.

B.9.1 Null Vaccination Strategy

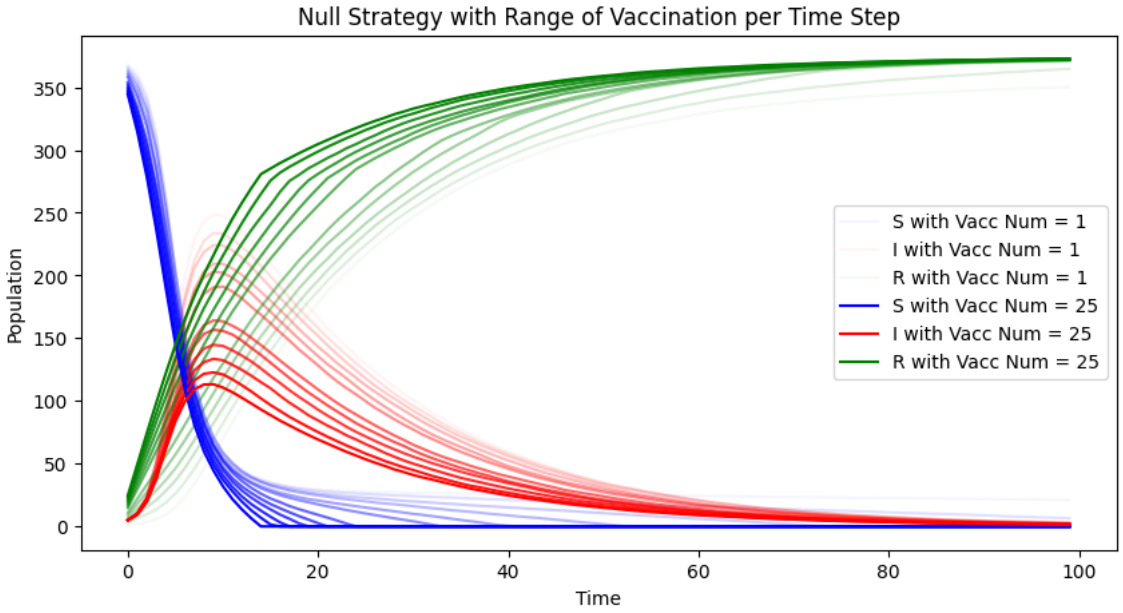


Figure 168

B.9.2 Blind Highest Degree Vaccination Strategy

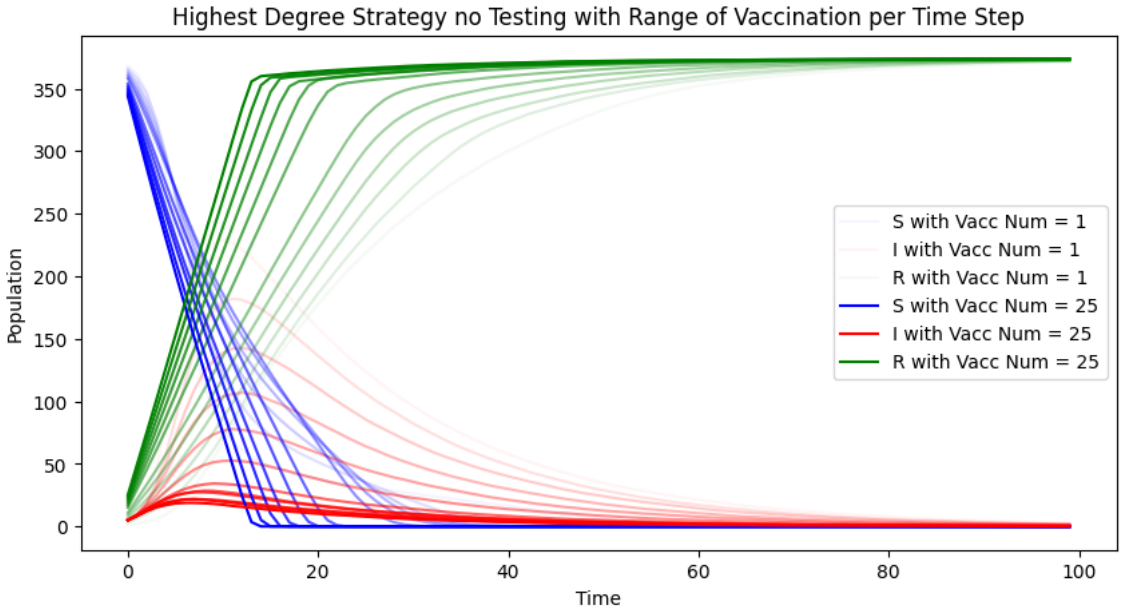


Figure 169

B.9.3 Dynamic Vaccination Strategy

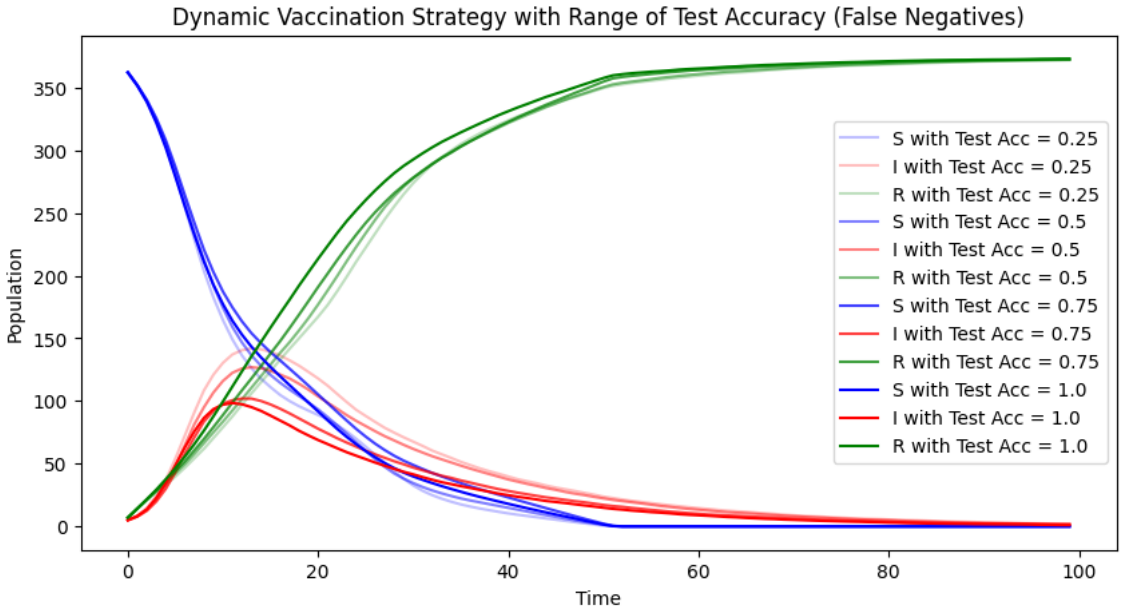


Figure 170

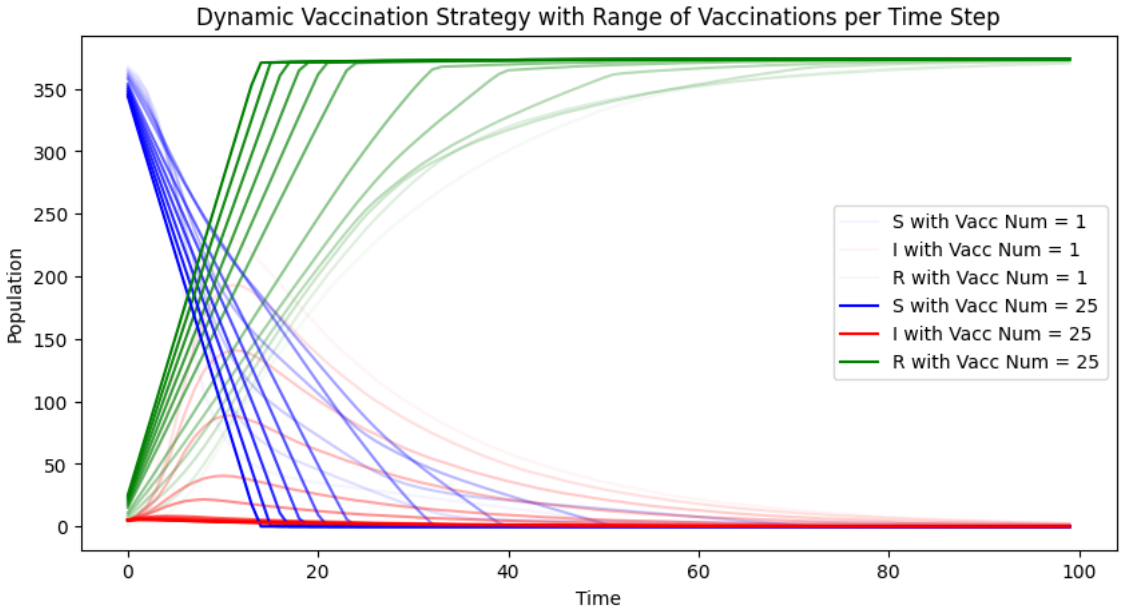


Figure 171

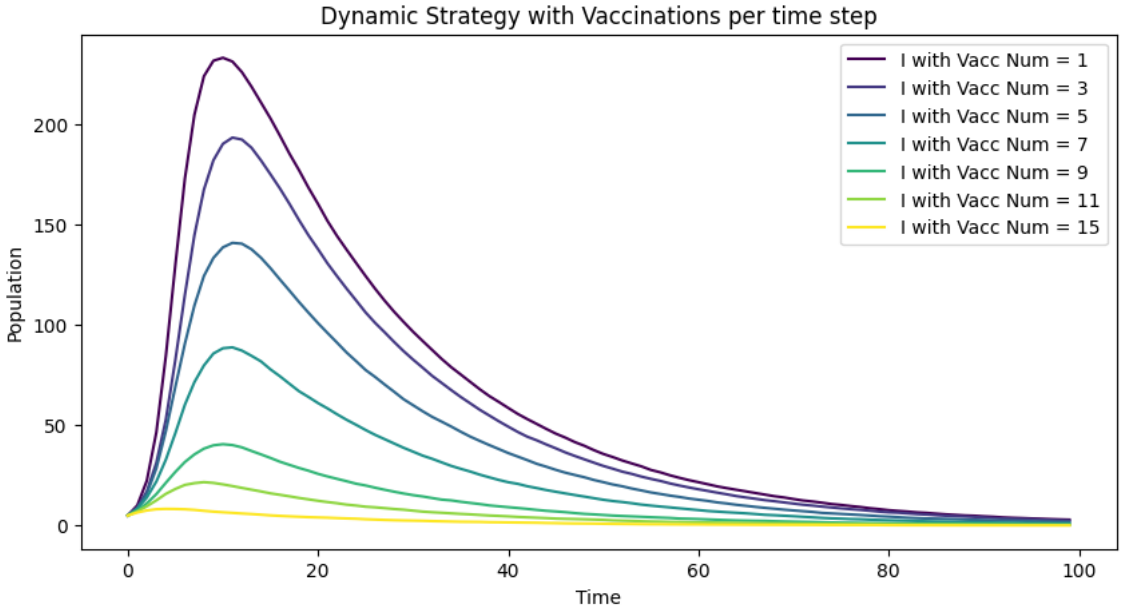


Figure 172

B.9.4 Vaccination Strategy Test Accuracy Comparisons

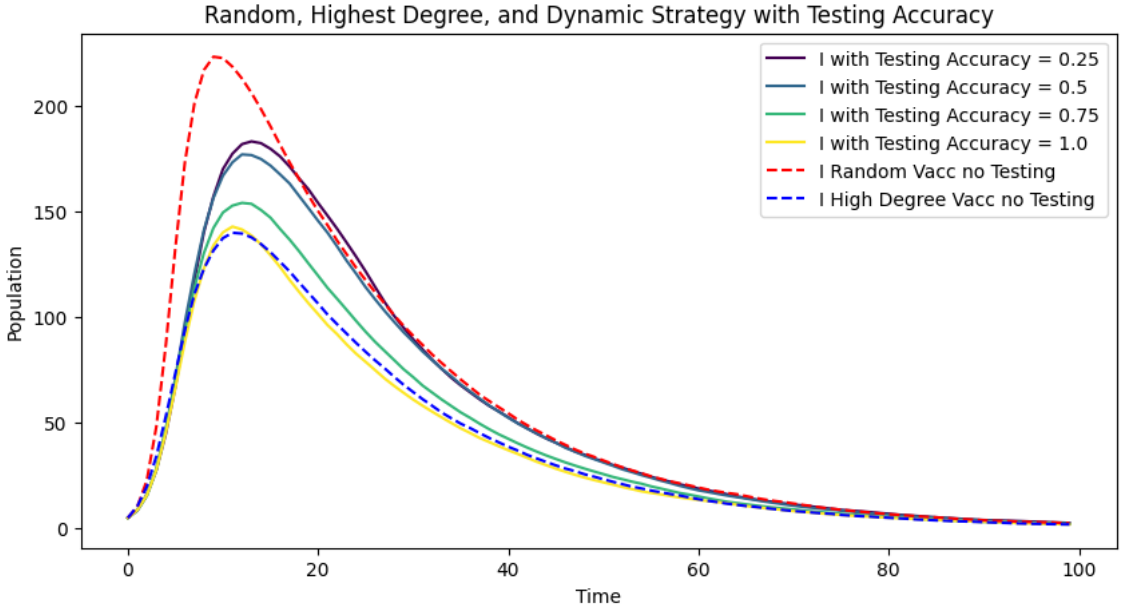


Figure 173

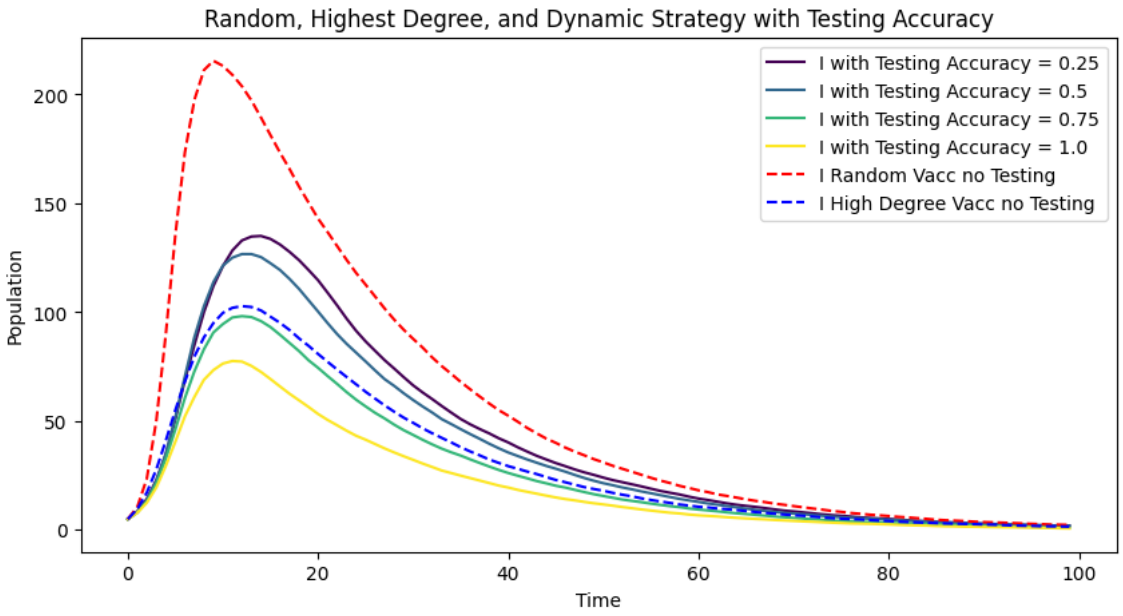


Figure 174

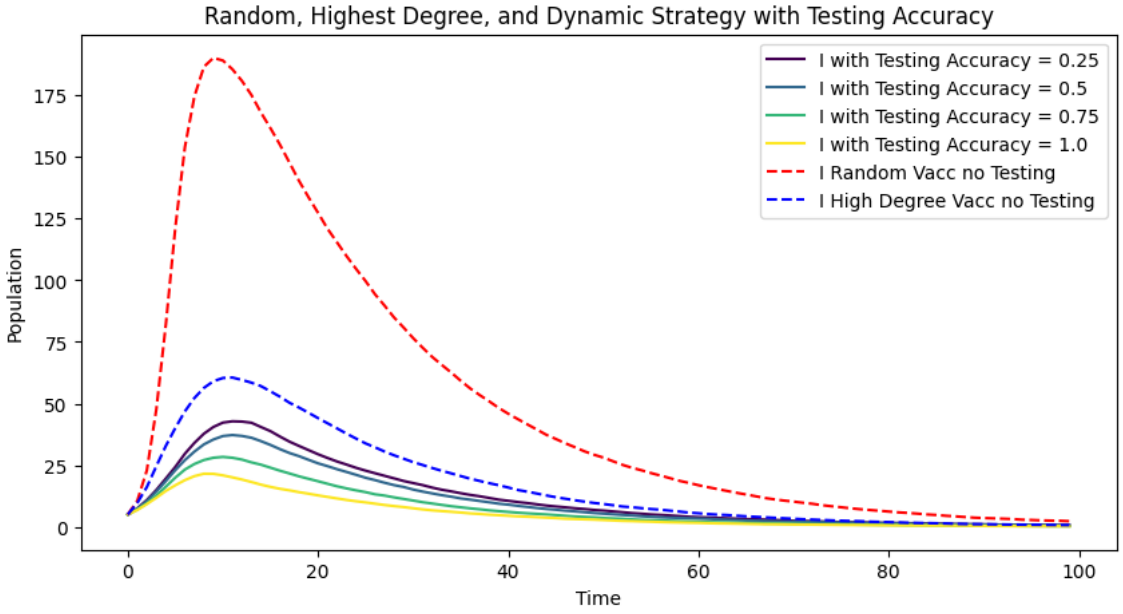


Figure 175

B.9.5 Vaccination Strategy Vaccination Num Comparisons

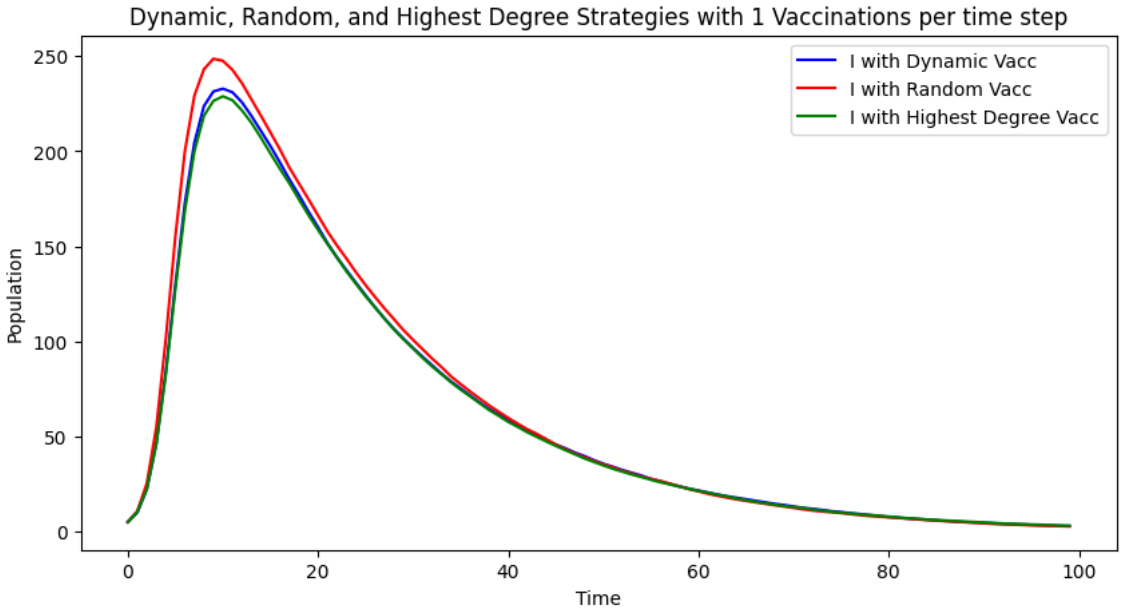


Figure 176

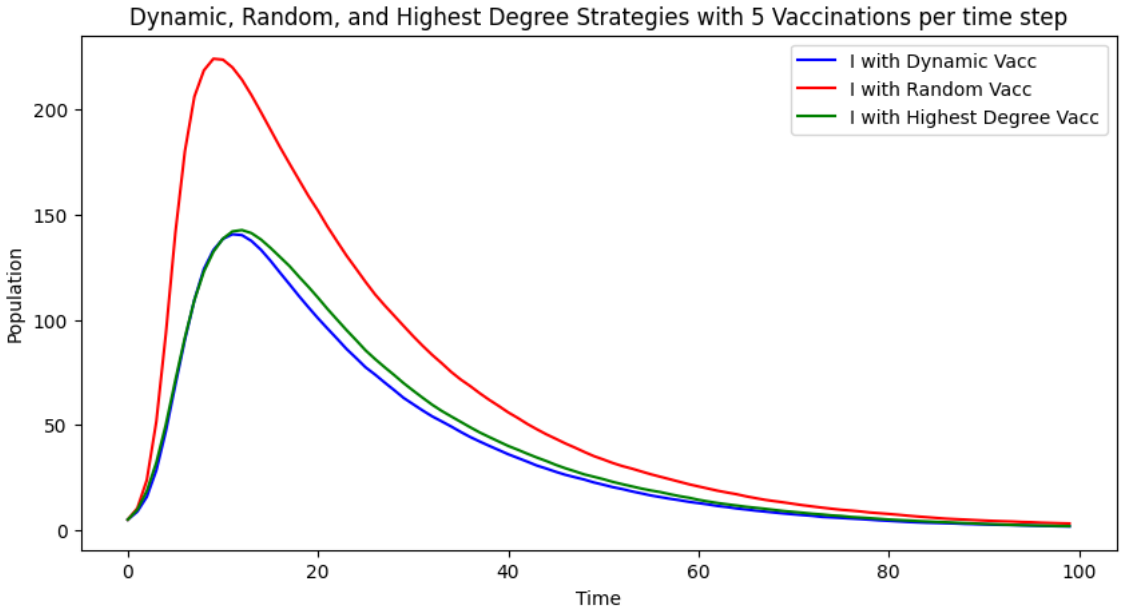


Figure 177

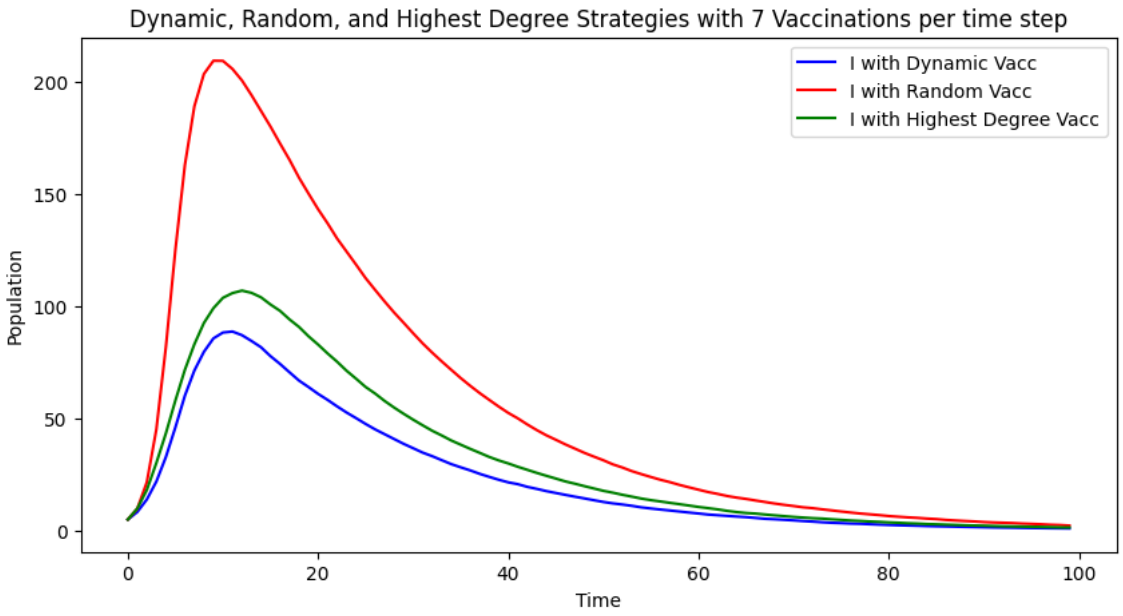


Figure 178

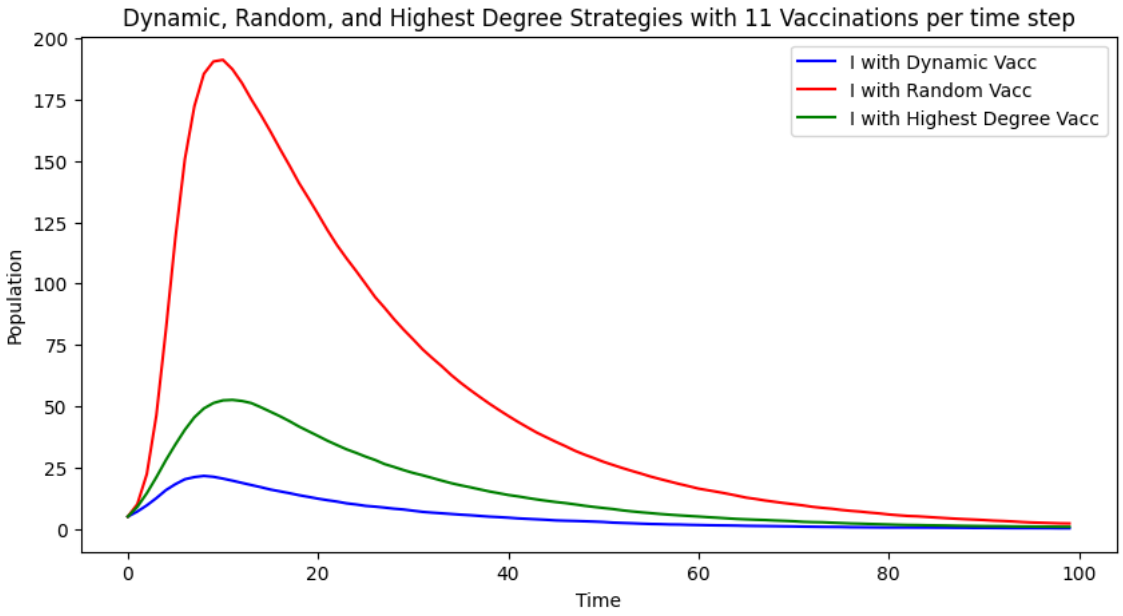


Figure 179

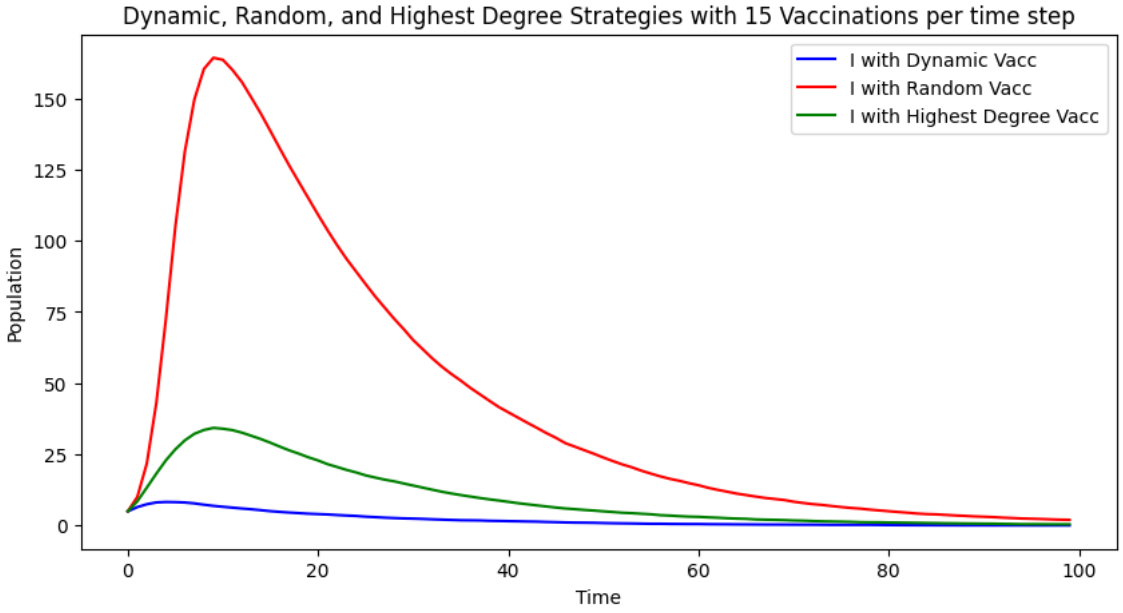


Figure 180

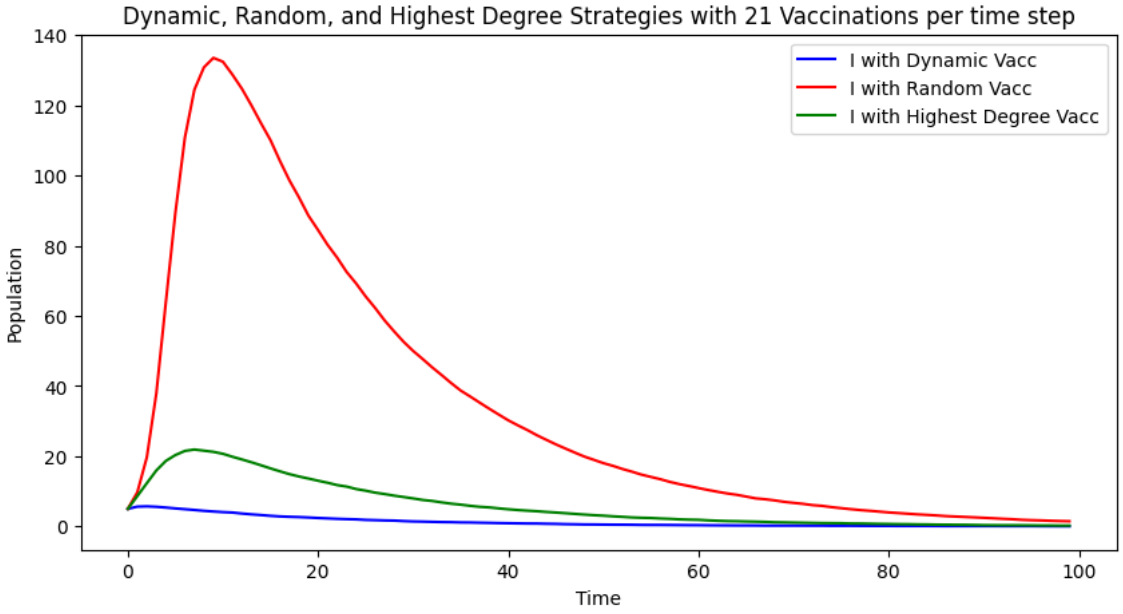


Figure 181

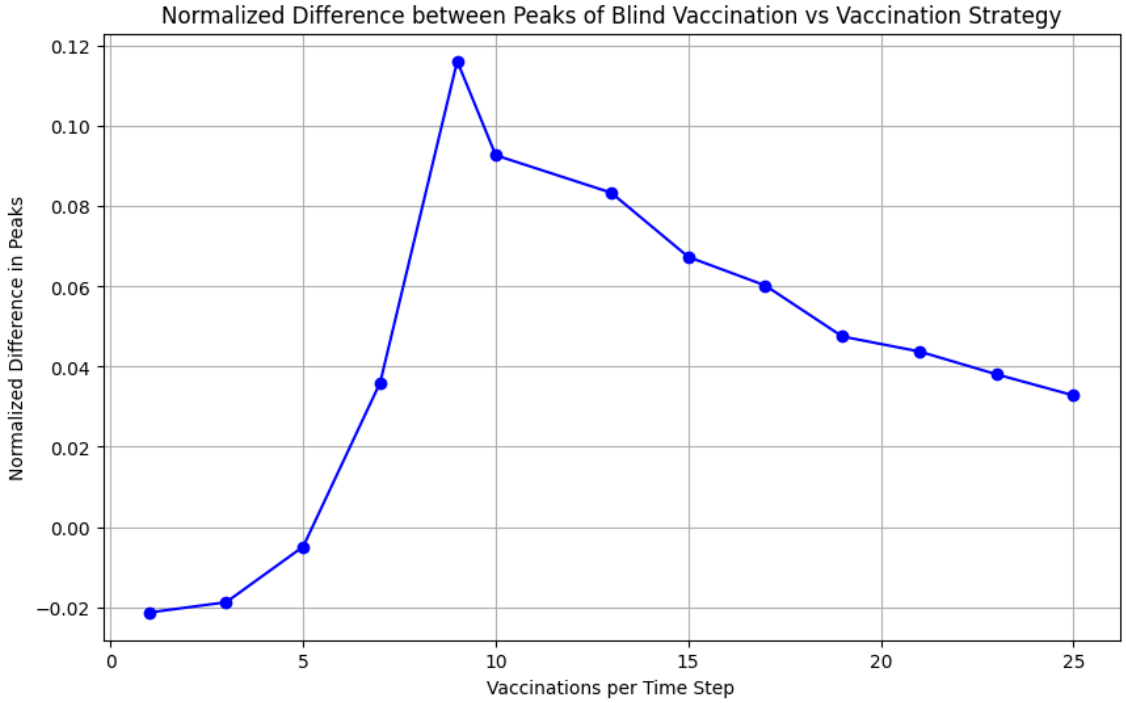

Normalized Difference between Peaks of Blind Vaccination vs Vaccination Strategy
Normalized Difference in Peaks
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
0
5
10
15
20
25
Vaccinations per Time Step

Figure 182