A Framework Agnostic Method for Sensitivity Analysis and Parameter Specification on Agent-Based Disease Models

by

Bruce Chidley

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
December 2024

Abstract

Modeling and understanding the spread of disease has been a topic of much focus for epidemiological researchers in recent years due to the effects of the COVID-19 pandemic. High levels of global attention and an abundance of recently collected data have created an environment for epidemiological models to be highly detailed and impactful. In order to ensure the high-quality performance of these models, the parameters fueling simulations must represent the real world sufficiently. This can be achieved by basing parameter values on accurate real-world data - data that are commonly in unusable forms or simply have not been collected. Thus, public health researchers must dedicate significant resources to collect new data or to verify existing data in order to create a model that best represents true disease dynamics. This can be expensive across many parameters, and so an alternative would be to focus attention on examining the data contributing only the model parameters that most affect simulation outcomes of interest. These significant parameters can be identified via a class of statistical analysis called sensitivity analysis. This work explores how sensitivity analysis can be leveraged in the task of parameter configuration in agent-based disease models. A plausible disease model was created and replicated across four distinct frameworks, where various sensitivity analysis methods were performed on them, noting the similarities and differences of the calculated results. Further,

I propose a novel algorithm that aims to mathematically identify parameters whose values may be incorrectly initialized, providing justification for its usage and how its results may be interpreted.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the emergence of COVID-19 in 2020, there has been a surge of research related to improving the level of understanding we have in relation to diseases of varying natures. The general consensus is that we, as a country and as a global unit, were not adequately prepared to handle something of this scale for a number of reasons [17]. One point of failure was related to how we handled the pandemic from an infrastructure and governmental point of view. As a society, the possibility of such a disease arising was not taken into account as seriously as it is now [4], and we were unsure of how to grapple with the situation, being simply unprepared from a resources standpoint. Hospitals and other facilities (such as long-term care centres) were not sufficiently equipped to care for large quantities of people [4], public health officials in different regions were inconsistent with their messaging [32], and regulations and mandates arose that were retrospectively shown to be sub-optimal for a number of tasks [32]. While this was all happening in the background, there was also the problem of medical researchers scrambling to understand exactly how the disease worked fundamentally while simultaneously developing effective vaccines, which is the second main point of failure.

We wanted to understand the disease on a biological level in detail, and the goal was to come to this understanding as quickly as possible given the circumstances [17]. This is because to fully understand how a disease should be combated, a thorough knowledge of how it exists and reproduces is necessary - vaccines cannot be realistically developed without this. It would hamper the possibility of non-pharmaceutical intervention (NPI) methods such as masking and social distancing being precisely enacted. While we have the technical skills available to gain these insights over the course of a long period of time, it is over the short term that we struggled to fully understand the nature of COVID-19 [94].

The third major point of failure was related to a lack of high-quality forecasting with respect to how the disease was expected to move through a population. The more precisely we can estimate how a disease will affect a population, the more precisely intervention methods can be employed. This precise implementation of intervention methods would have resulted in a reduced burden felt by the population/region. However, the models used to inform public health decision-making were initially of questionable quality, oftentimes leading the results of such decisions to be less impactful than expected [4]. The forecasting models being inaccurate was in part due to their general level of sophistication and modernity (as well as the pressure to release public statements), but was primarily due to the lack of data available near the start of the pandemic [13]. As expected, certain pieces of data relating to the disease itself were unclear since there simply was not enough time to observe and gather this data. This caused certain model parameters to have wide value ranges (in addition to the true varying nature of certain parameters like the basic reproduction number), resulting in the models they fuel to produce a wide range of outcomes with

respect to what modelers were interested in [48].

While the first two points are certainly very important and require significant attention, this work aims to highlight the third point as a crucial and readily solvable problem. To do this, I first create four agent-based models across four different platforms (Mesa, Repast, RDDL, and NetLogo) following my work published at the International Conference on Automated Planning and Scheduling detailing disease modeling with RDDL [15], using the geography and population statistics of Kingston, Ontario. While these models were created with the intention of being as similar as possible, there are some key differences between them due to the functionalities of the platforms they were created in. Due to this reality, I compare the simulation outcomes between each model to gain a deeper understanding of how they relate to each other. Once I establish the relationship between the models, I perform sensitivity analysis on the models at a number of analysis points (called "measures" henceforth) using a wide variety of commonplace methods in order to uncover which parameters have the most influence on measure values. After the effect of parameters on model output has been concretized, I demonstrate how further parameter insights can be gained by comparing model output to "ground truth" measure values, backed by sensitivity analysis. I present an algorithm that performs this comparison, resulting in an ordered list of parameters being created, which describes the likelihood that each parameter is incorrectly initialized relative to all other parameters.

1.1 Objectives and Contributions

This work aims to provide three main contributions. First, using the generated models, the significant variance in simulation outcome observed as a result of parameter

value manipulation intends to draw attention to it as a significant problem for modelers. While sensitivity analysis methods have long been known and implemented, the frequency of their application as a tool to highlight problematic data is far underutilized in general, with incorrect approaches being taken often even when attempted [76]. It is unclear whether the reason for this is a lack of understanding of what tools exist or whether it is due to researchers deeming sensitivity analysis to be somewhat unnecessary. My hope is that this work both promotes its existence and shows its importance.

As a second contribution, it is also a goal of this work to provide a full end-to-end description of how agent-based models may be created, how the parameters comprising them may be analyzed, and how this may be used in determining which parameters (if any) have not been initialized correctly with respect to intended outcomes. It is often the case that works of this nature only cover one area at a time, with some focusing heavily on model creation and neglecting analysis and others focusing strictly on the theoretical analysis without any concrete examples. This work describes the creation of four agent-based disease models from the ground up and explores their similarities, with thorough sensitivity analysis being performed on all four. In this way, the work here serves as a "proof of concept" for this kind of modeling.

The third contribution is the novel method for identifying which parameters are likely to be incorrectly initialized based on how a model's output compares to the "ground truth" across a variety of measures based on the results of sensitivity analysis. It is the case that performing sensitivity analysis takes significant time to set up and execute (and modelers do not always *understand* exactly what sensitivity analysis truly entails), and so many modelers do not take sensitivity analysis any further

than what is directly output. This extension to sensitivity analysis takes the logical next step and uses its results to mathematically identify which parameters may have been incorrectly initialized, acting as a concrete point of reference that modelers can leverage to justify allocating resources toward collecting and refining parameter-specific data.

The main objective of this work then, is to develop a tool to automate the sensitivity analysis process on agent-based disease models, create four novel agent-based disease models that are run through this tool as examples, and to show how the results of sensitivity analysis can be extended mathematically.

1.2 Organization of Thesis

In Chapter 2, I outline the background needed to understand the nature of this problem in its entirety. I first cover disease modeling, examining compartmental models and how they are typically either modeled as mathematical or agent-based models (providing descriptions of both). Then, I discuss agent-based modeling software, highlighting the four models that are used in this work (Mesa, Repast, RDDL, and NetLogo) and providing some brief examples of their implementation. The last section in Chapter 2 is dedicated to sensitivity analysis, describing what it is and how it can be used, along with very high-level descriptions of the 11 sensitivity analysis methods that are used here. Chapter 3 outlines how each of the four models was created, focusing on geography, agent generation, and the underlying disease mechanics driving the models. This is followed by examining key differences between the models where differences exist. Chapter 4 provides a summary of how the sensitivity analysis methods were implemented in detail and how the downstream parameter analysis

algorithm works. Chapter 5 shows the results of the sensitivity analysis, along with a few example cases that show how the algorithm reacts to different parameter configurations. Chapter 6 is a discussion of limitations and future work, followed by a summary and detailed section discussing relevant related work. Additionally, there are three appendices that contain a number of tables and figures that account for different cases not directly mentioned in the main body of this work.

Chapter 2

Background

2.1 Disease Modeling

Knowing how a disease may spread in a population is essential for determining the best course of action to take in order to minimize its negative impact (or simply to understand detailed dynamics for research purposes). Disease modeling typically falls into one of two categories: researchers are either aiming to understand how it exists and multiplies within a host [89], or they are aiming to see how it moves about some population at a larger scale. Increasing the performance of models in these two categories has been a topic of research for many decades, and these types of models have been applied to real-life scenarios in a multitude of cases (OpenABM-Covid19 and Covasim being examples of population-scale models that have seen adoption from public health officials [34, 41]). While it is clear that both of these categories are invaluable for mitigating the potential negative impact of a disease, this work is concentrated on understanding and interpreting how a disease spreads within a population, and as such, I will focus the rest of our attention on this subject. This is not to take away from the importance of the intra-host approach - in fact, these kinds of models are

often crucial in gleaning insights related to fundamental disease dynamics that are used in population-scale models. For instance, knowing the speed at which a virus multiplies and dies out in a host can tell us how quickly they begin shedding the virus and for how long, which tells us the likelihood of a host spreading a virus at any given point after initial infection. This can then be captured in a population-scale disease model and is essential in tracking its behaviour. To understand these parameters and related data in more detail, I must define a few important concepts.

Used commonly in disease models, and discussed frequently on global stages as a way of quantifying the reproductive power of a disease, the basic reproduction number (R_0) represents the average number of susceptible units that one infectious host will spread the disease in question to. This value can be derived in different ways - directly observing a disease's spread through a population and observing its simulated spread in some model that has encoded its biological concepts are two common such manners [21, 22]. The basic reproduction number can also be calculated directly from other known statistics, which themselves can be broken up into various more complex parts [22]. Though it can be very involved to derive R_0 with high accuracy, a common and very simple way of explaining its value is through the equation $R_0 = \beta/\gamma$, where β represents the number of individuals that the infection is spread to over some period of time, and γ represents the rate of recovery relative to that same period of time. As expected, R_0 can vary quite significantly from disease to disease, but it can also vary within one given disease and strain depending on the environment it exists within. Typically, more urban or densely populated environments cause R_0 to be higher when compared to more rural or sparsely populated environments. This is due to the rate at which people make contact and the proximity that they make contact at [83]. Table

Disease	Basic Reproduction Number
Measles	12-18 [29]
COVID-19	1.9-6.5 [2]
Polio	5-7 [23]
Mumps	4-7 [23]
SARS	2-4 [59]
Influenza	1.2-1.4 [16]
Ebola	1.4-1.8 [93]
Smallpox	5-7 [23]

Table 2.1: Examples of diseases and their basic reproduction number ranges.

2.1 shows some common diseases and their respective reproduction number ranges for referential purposes.

While diseases do not technically refer to R_0 in their multiplication processes (i.e. this number is not written in their DNA somewhere), the basic reproduction number is a way of simplifying many complex ideas into one number. With the existence of this number, disease models can leverage it as a way of "forcing" the desired disease dynamics. While it may seem like a better alternative to have the disease spread be more closely related to other concepts involving complex transmission probabilities, it is often the case that simpler models are more desirable than complex ones [64] (on top of the fact that it may not change the disease dynamics in any noticeable way). This can be for a number of reasons, but one major factor is generalizability. If one wishes to use a model for multiple different tasks, perhaps relating to multiple different diseases, then having a more general model can make this process much easier (and, in some cases, can allow for its possibility entirely). Certain diseases operate *very* differently than others when it comes to transmission (an example being airborne diseases against transmission via direct bodily fluid ingestion), but they can

nearly all be described in terms of the basic reproductive ratio. For this reason, and for other reasons like interpretability, the basic reproductive ratio was used to facilitate disease spread in the models shown in this work.

Still, there *are* other commonly used parameters relating to disease spread that can be used in these sorts of models, often dealing with the disease's life cycle. For instance, the prevalence of a given disease in a certain region is sometimes measured through its abundance in sewage systems [39]. In these models, the lifespan of a disease in non-host environments must be taken into account. However, the models in this work do not take this approach and instead look to model the disease's existence within a host *only*. To understand how exactly this is done, I will describe the existence and usefulness of compartmental models.

2.1.1 Compartmental Models

Compartmental models refer to the division of a disease's lifespan within a host. Throughout this lifespan, how the host interacts with the world with respect to disease spread can vary quite significantly. One of the most basic and commonly used compartmental models is called the "SIR" model, where "S" stands for "susceptible", "I" stands for "infectious", and "R" stands for "recovered". In models adopting this strategy, hosts must be in one of these three compartments, moving through them linearly from susceptible to infectious to recovered. If modelers choose to have the host loop back to the susceptible class after a period of time, then it transforms into an "SIRS" model. A susceptible host implies that they do not have the disease in their system, and they can catch the disease readily. An infectious host implies that they have the disease and can spread it to other hosts via some transmission path. A

recovered host implies that they no longer have the disease in their system, and they can *not* catch it. This basic compartmental model can see a significant modification to encapsulate more complex disease mechanics and to have it more closely mimic reality. For example, compartments for death can be included, compartments for age and type of disease contracted (when multiple strains exist in conjunction) can be included, and subdivisions of these compartments can be introduced (an example being if infectivity changes over time, and modelers wish to capture that change by having multiple "I" compartments). Another common modification of this structure is the addition of the "Exposed" compartment that is placed after the susceptible compartment and before the infectious compartment, turning the model into an "SEIR" or "SEIRS" model. In this compartment, hosts are said to have contracted the disease but cannot spread it. This is also sometimes referred to as the "incubation period" for a disease. The period of time that a host remains in the exposed, infectious, and recovered compartments is disease-specific and usually comes from real observed data. The models presented in this work take the form of an "SEIRS" model (shown visually in figure 2.1), as it is very generalizable to different diseases, given that this extension can be reduced back to a simple "SIRS" model by simply letting the host remain in the exposed compartment for 0 units of time. These compartmental models are typically created in one of two ways, either falling into the category of a "mathematical model" or an "agent-based model."

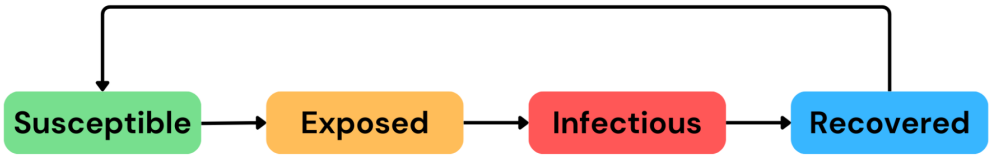

Figure 2.1: The classes that an individual moves through in an SEIRS model

2.2 Mathematical Models

Mathematical disease models are traditionally set up as a system of differential equations where each equation maps to one compartment (per the user-specified construction of the compartments). Shown below is an example mathematical model using the "SEIRS" compartments:

$$\begin{aligned}
\frac{\mathrm{d}S(t)}{\mathrm{d}t} &= \upsilon R - \beta SI \\
\frac{\mathrm{d}E(t)}{\mathrm{d}t} &= \beta SI - \gamma E \\
\frac{\mathrm{d}I(t)}{\mathrm{d}t} &= \gamma E - \delta I \\
\frac{\mathrm{d}R(t)}{\mathrm{d}t} &= \delta I - \upsilon R
\end{aligned}$$

In this example, the Greek symbols β, γ, δ, and υ represent the rate at which populations move between compartments (these values are between zero and one). Positive values in each equation represent the rate at which populations flow into a compartment, and negative values represent the rate at which populations flow out of a compartment. It is important to note that the rate at which populations

transfer between the susceptible and exposed classes depends both on the susceptible and infectious counts. In these models, populations are handled as continuous real numbers (in contrast to discrete numbers, which is how these systems operate in the real world), where the compartment values change per the mechanics just described. These types of systems are initialized with some values and are stepped through either discretely or continuously, depending on the construction and tools being used. While mathematical models can be quite simple, like in the example shown above, they can really be as complex as the modeler chooses to make them. Some mathematical models include more nuanced population statistics, taking age and other factors into account. There is also the possibility of including other hosts (such as animals) that spread disease in a different manner. These possible modifications to the basic mathematical models are near endless, increasing the complexity (and changing the quality) with the addition of each new component. However, this has the potential to take away from the greatest strength of mathematical models - their interpretability. This also runs the risk of increasing model complexity with a very minimal payoff in terms of model accuracy, which is a known problem in the sphere of modeling that can have many unexpected effects related to generalization and performance when receiving new data [64, 81]. Mathematical models are thought of as not being quite as "realistic" as agent-based models since populations are examined in aggregate, individuals in the population do not have specific behaviours, and there is no possibility for geographical components. However, it is very easy to understand exactly how and why the compartment values are the way they are throughout a simulation, which can be essential given some specific tasks.

Mathematical models in this form are generally created with the purpose of

demonstrating theory or experimenting with parameters to view theoretical outcomes that can be used for further analysis and discussion. It is uncommon that these models *alone* are sufficient for modelers in terms of accuracy and approximation of real-world phenomena, though the discussion of model significance and relevance is one that has been present for many years in philosophical branches of research. The interested reader is pointed to [61, 64, 86], and [27]. In most cases, when researchers are attempting to convince public health officials to adopt certain strategies or to illustrate more effectively the possible outcomes of certain systems, they will use a combination of mathematical models and agent-based models, which tend to be more "convincing" in a few ways, and elicit dynamics that have the possibility to more closely mimic certain systems [20].

2.3 Agent-Based Models

Agent-based models require more overhead in terms of programming and computational power and are less interpretable by nature (being more "black-box"), but in turn, can much more closely represent most systems by way of representing objects in the system as "agents" that act individually according to certain programmed characteristics. In many cases (and in the case of most disease models such as the ones being presented in this work), there is a geographical aspect to these agent-based models, where the agents will move about some space, affecting the world and/or other agents. It is the combination of agents acting on their own and the unique geographical setting that agents move about in that really sets these kinds of models apart from mathematical models in terms of output - agents interacting with objects

may result in certain unexpected outcomes that are challenging to capture in mathematical models, and the geography they exist in may vary simulation outcome quite significantly.

While the construction of agent-based models may be quite different from that of mathematical models, seeing as how the former uses complex simulations with no inherent structure and the latter strictly uses a system of differential equations, they have some important similarities. The most obvious would be the parameters that fuel specific important parts of the simulations. In the case of disease models, parameters such as the basic reproduction number or the length of time spent in certain SEIR compartments may be presented in the exact same numerical form - it is only how they interact with the model that differs. For this reason, mathematical models and agent-based models can be reasonably compared when they are attempting to model the same thing. In this case, the mathematical model acts as the "theoretical" that the agent-based model can be compared against when performing certain analysis [45]. However, one should note that significant differences can arise in agent-based models that lead the "theoretical" mathematical model to appear quite different in output when compared to the agent-based model (as we will see in the case of our model). This may lead researchers down the wrong path in their analysis, and the models should be compared with caution. Another important similarity is in the types of analysis that can be performed on both of these models. While there are certain techniques that are only possible on mathematical models due to their structure, many techniques, such as sensitivity and uncertainty analysis, can be performed on these models in much the same way. In these cases, the models can be similarly compared with one another to see how the model transition from mathematical to

agent-based affects things.

Agent-based models are usually "more trusted" by those who are to make decisions based on their output when compared to mathematical models due to the perceived (and in many cases, actual) increase in realism and accuracy that they bring [20]. While these may be the best way of influencing certain people, it is not always the case that there are the resources available to create such models - they usually take significantly more time in all aspects due to their complexity. So, they are not the most viable of options in time-sensitive situations or where having a detailed agent-based model would result in minimal gains in terms of model quality when compared to the mathematical model counterpart. For example, in the case of COVID-19, most models that were created were mathematical compartmental nature [67]. This is partly because tracking COVID-19 and predicting how it would spread in the immediate future was crucial - there simply was not enough "breathing room" to sit down and carefully generate an agent-based model. It is also because these mathematical models lend more naturally to statistical analysis, which can be valuable given the circumstances.

2.3.1 Agent-Based Modeling Software

Deciding how these agent-based models should be constructed is not a simple task - and even when one has a general idea of how they want things to be, deciding how *exactly* one wishes to encode it can be challenging. There are many agent-based modeling software or tools that exist that all specialize in slightly different things that are better suited for different tasks.

These software/tools can be broken up into a few categories depending on how

they handle certain topics. For instance, there are certain software like NetLogo and StarLogo that use their own platform and have their own programming language. The benefit of these is that they are typically very user-friendly, have nice built-in geographical representations, and lend well to experimentation when one is not an expert in traditional programming [71, 88]. However, these tend to have less depth in terms of customizability, making their ability to scale to more complex systems relatively weak. There are also others that are built into certain existing programming languages like Python, Java, or C, such as Mesa, Repast, or MASON [19, 49, 51]. These are comparatively less user-friendly and do not have the same quality-of-life features but have much deeper capabilities when it comes to actually specifying how the model operates. This is primarily due to the libraries/packages that are offered through the programming languages coupled with the simplicity of the tools' construction, offering nearly endless possibilities. For this work, Mesa, Repast, RDDL, and NetLogo were chosen, and basic descriptions of them, paired with the reasoning for choosing them, are shown in the following sections. These modeling frameworks are elaborated on in appendix D, providing slightly more technical descriptions.

2.3.2 Mesa

Of the many Python-based agent-based modeling tools available, Mesa is very simple compared to some existing tools when it comes to getting a model running, and it uses straightforward data types. Comparisons between the four tools used in *this* thesis can be seen in Appendix D This makes downstream analysis much easier from a development standpoint, simplifies usage of adjacent Python libraries, and makes debugging a much easier process. However, this simple approach to agent-based modeling comes

at the cost of speed and scalability. Other agent-based modeling software such as Repast and MASON boast being able to handle very large and complex models due to their distributed computing frameworks (which Mesa does not natively support), among other optimizations. However, these kinds of software typically pigeonhole the user into using their specific data types and unique constructions, making it less intuitive and harder to customize or implement with downstream code. This simplicity is why Mesa was chosen as one of the four models that were used in this work.

As a whole, Mesa has not seen as much uptake and discussion as some other agent-based modeling tools that aim to fill a similar niche, but that could be in part due to its novelty, size, and notoriety in comparison to some other options. However, for our purposes, Mesa performs very well and lends nicely to the sensitivity analysis methods used here. This is thanks to its simplicity - basic Mesa models simply output values in natural Python data-types, whereas others write to files by necessity, making some tasks more difficult. For example, in cases where multiple environments are running simulations at the same time and performing downstream analysis, writing to and reading from files can become complicated when multiple environments use the same directories (overwriting and reading from incorrect files may occur). This sort of problem is common and requires further labour from a programmer's perspective. Since disease modeling is often done by those with limited programming experience, this is something to take into consideration when selecting agent-based modeling tools to use.

2.3.3 Repast

Repast is one of the most well-established agent-based modeling tools, being available to use in Java, C++, C#, and Python. For this research, I will be using the Python implementation, Repast4Py. Repast4Py leans to the "complex" side of agent-based modeling tools, requiring a moderate amount of background knowledge of general agent-based modeling practices and techniques. It also requires that the user studies their specific data types and model construction constraints in a relatively deep level of detail prior to implementation to avoid unexpected interactions or errors. This level of complexity is partly due to their distributed computing methods that aim to greatly increase the scalability of these models by leveraging available CPU cores in ways that other popular options do not. The general spatial structure is that given some 2-dimensional space (of which there are varying constructions within Repast4Py), a user-defined number of CPU cores will be assigned some region in the space such that they can all be stitched together to re-construct the original space. With this structure, as agents pass between sub-regions that are covered by two different cores, they must be deleted from one core and re-created on another core. While this adds to complexity, it has the potential to vastly reduce simulation runtime at large scales.

Repast4Py (and other Repast implementations) are at their best when dealing with highly complex spatial models with large agent counts. While the disease models generated in this paper do not necessarily exploit this strength as much as other models may (given that it is only operating in a relatively small region with a moderate agent count), the domain of disease modeling is one that could greatly benefit from it. For instance, it is sometimes useful to examine disease mechanics on country-wide or global scales (i.e. agent counts of millions to billions). While the details would be

much coarser in these settings, agents would still have their respective behaviours, and the computational power needed to run simulations of this nature would be much larger. As a result, optimizations from certain tools like Repast have become essential.

2.3.4 Relational Dynamic Influence Diagram Language

Relational Dynamic Influence Diagram Language (RDDL) comes from the field of Automated Planning, and aims to solve what are called "probabilistic planning" problems. This is done via agent-based simulations, meaning that RDDL can be used as a tool to create disease models. When these RDDL models are created, simulations are run with the functionality of having an automated entity (called the "planner") take actions that can affect agents or the environment in a variety of ways. The planner takes these actions with the aim of maximizing some user-defined reward function. The planner, through a series of decisions made with the help of complex heuristics [26, 77, 84], generates a "policy" at time step 0 that defines how and when it will take actions for upcoming simulations. It is important to note that the actions it decides to take are *not* made during a model's execution but are made before a simulation begins based on the initial configuration of the world. The models created in probabilistic planning problems are almost always stochastic in nature - if they were deterministic, then more traditional methods from the field of Automated Planning could be leveraged.

There are countless possible applications of such a language, and the research spans multiple domains. The majority of these models are created on a relatively small scale. This is not to say, however, that the models are not complex - it is actually quite the opposite, with many of these models being created specifically to

test the capabilities of certain planners (as there are many different planners) under different conditions. "Small scale," in this sense, simply refers to the number of agents and the general size of the world that is being simulated. Given the complexity of the heuristics used to generate policies with these planners, scaling up the size of the model causes the time needed to generate such policies to increase very significantly and can cause the planners to fail in some cases. As such, disease models that potentially operate on the scale of hundreds of thousands or millions are not yet feasible. These models would need to be scaled down quite significantly (to the hundreds in some cases) or localized to some specific small setting, such as a classroom, which has seen experimentation [31]. In this work, I will be focusing on applying RDDL to the problem of sensitivity analysis because of the possibility for future impactful work that can be done with probabilistic planning in relation to disease modeling.

2.3.5 NetLogo

Unlike the other agent-based modeling tools discussed, NetLogo utilizes its own platform for model creation, which can be downloaded online. The "selling point" of NetLogo is that it offers direct visualization of simulations with a simple and intuitive user interface. Figure 2.2 shows a screenshot of the NetLogo interface using the "Wolf Sheep Predation" model, which is commonly used to introduce new modelers to NetLogo due to its simplicity and effectiveness. It also serves as a baseline for comparing further ecological models to [40].

Due to the fact that the display is so straightforward and that users can interface with the model quite easily via constructions like buttons and sliders that directly

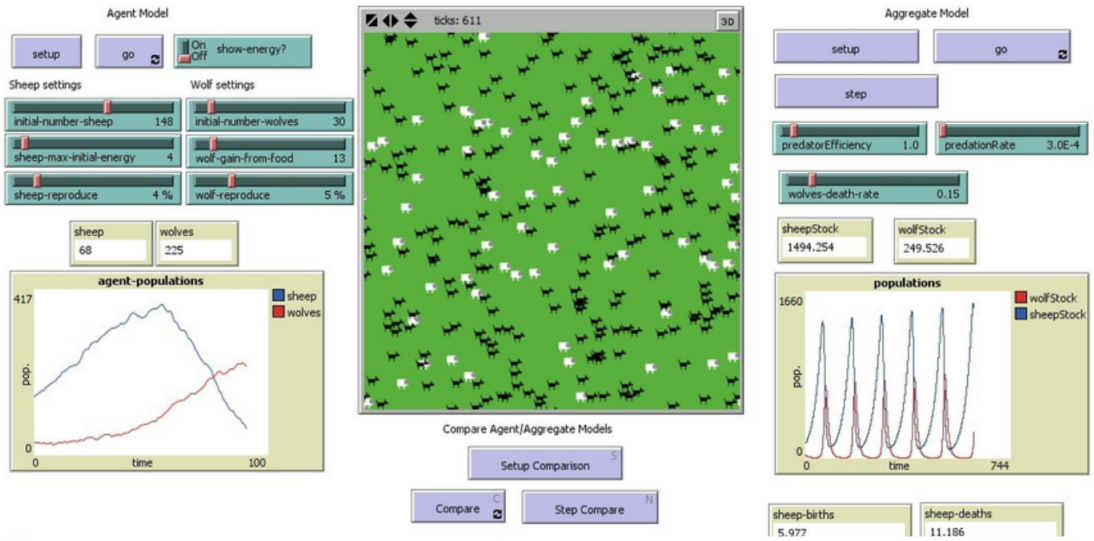

Figure 2.2: Example of the Wolf Sheep Predation model in Netlogo - a common example used to instruct new modelers.

change parameter values, NetLogo is a very popular choice among modelers with limited programming experience or for those who are looking to learn about agent-based modeling as a whole. However, while NetLogo is very strong on the visualization and interface fronts, there are many downsides related to its computational power and extension to other environments.

In NetLogo, users must learn and use the built-in NetLogo language. Unfortunately, this language does not have the depth associated with more popular coding languages like Python or Java, and as such, users may find themselves unable to generate certain model behaviours that they wish to. This adds a layer of difficulty that modelers must overcome, and even when being used to its fullest, the language simply cannot compete with more widely adopted languages such as Python, Java, or C in terms of model behaviour.

The next major drawback of NetLogo is its computational power. NetLogo was

not created to handle very complex simulations, but rather was created with the idea of visualizing and interacting with models that were simple enough such that they could be meaningfully visualized on the small window assigned this task. As such, there is a significant slowdown as model complexity increases, making it a poor choice in many circumstances. In the context of disease modeling, models are as complex as the situation calls for, and NetLogo may still be a viable option if the task is to model disease spread in some small regions. As a result of its primarily educational nature, NetLogo is often not considered for benchmark comparisons [47].

For the task at hand in this paper, NetLogo is serviceable and produces results in a somewhat reasonable timeframe. This introduces the possibility that modelers may aim to perform sensitivity analysis on results generated from a NetLogo model due to its simplicity, which warranted its inclusion here.

2.4 Sensitivity Analysis

All of the models and processes above can have their outputs interpreted in a wide variety of ways depending on the requirements for a given task. Sensitivity analysis is a common technique used to mathematically identify how each parameter comprising a model affects different measures at different points in time (some lesser-used methods apply to the whole time frame of a simulation, but the majority operate at a single point in time. Because of this, modelers often aggregate a collection of values into one single value depending on the sort of information they wish to discover). Some common points of interest for modelers would be how parameters affect equilibrium values (should they be reached), maximum and minimum values, and when these types of values occur. Sensitivity analysis is a deterministic approach comprised of

many, sometimes complex, steps that can take different forms depending on the kind of sensitivity analysis in question.

Oftentimes, sensitivity analysis methods will provide the user with some type of "sensitivity indices", which show a parameter's impact on model outcome from slightly different angles. These sensitivity indices range from first-order, second-order, etc., to total-order, representing an individual parameter's contribution to model output, the contribution to model output of parameters taking into account interactions with one other parameter, and so on, with the total-order sensitivity index describing the contribution of parameters taking into account all interactions. These higher-order sensitivity indices serve to convey certain non-linear effects that can arise from parameter interaction.

A variety of methods falling under the sensitivity analysis umbrella were explored, following some of the most common techniques used, encapsulating all methods that are facilitated by the SALib package in Python [33, 37]. It is important to note that not all of these methods may be strictly classified as "sensitivity analysis". However, since they all are very closely related to the topic and can be used in conjunction with each other to gain a full picture of the model in question, these methods will all be classified as "sensitivity analysis methods" for simplicity.

Sobol' Sensitivity Analysis

Sobol' analysis is a type of global sensitivity analysis, meaning that all parameters are varied simultaneously when running simulations, and is one of the most common forms of sensitivity analysis [36]. Sobol' analysis was originally created with mathematical functions in mind but is easily extended to agent-based models by letting $f(x)$ be

equal to the output of a given model at some point, and all parts contributing to the value of $f(x)$ be equal to the specified model parameters [74]. This is possible due to the minimal assumptions needed and the black-box nature of the function in mind that Sobol' analysis operates on. With this in mind, this process can be carried out in quite a straightforward manner for modern agent-based models, and all sensitivity indices can be retrieved from first-order to total-order. Sobol' analysis is arguably the most common form of sensitivity analysis and can be applied in nearly all situations, providing a relatively high level of insight into parameter performance.

Method of Morris

The Method of Morris, first presented by Morris in [54] and later refined in [11], operates by splitting the range of each parameter into k equal regions where k is some value specified by the user. Generally, a higher k value will result in more precise analysis at the cost of more required data. The Method of Morris loosely works by selecting some parameter at a given value, calculating model output with this configuration, perturbing the model by varying other parameters such that they remain in the current region of analysis with respect to k, and calculating model output at each step, generating some distribution. Performing this process over all k regions gives a good picture of how each parameter affects the output across its range, and the results can be aggregated into single values. The mean and standard deviation of this distribution then represent the overall influence of the parameter and the influence of the parameter with parameter interaction (non-linear influence), respectively.

The Morris Method, along with Sobol' analysis, is one of the more cited and

used methods for sensitivity analysis due to its interpretable and relatively simple structure - that being one revolving around "elementary effects" of parameters per the process outlined in the paper initially conceiving it. The Morris Method is a distribution-based method of sensitivity analysis, and as such, it is most fruitful when the parameters under scrutiny affect model output differently at different regions of their total range.

PAWN Sensitivity Analysis

PAWN is another density-based method of sensitivity analysis with the aim of having a simple implementation due to its focus being directed at approximating the cumulative density function (CDF) of a model rather than its probability density function (PDF), which is said to be more complicated and computationally intensive to derive [62]. First, the base CDF for the model is approximated, and then all conditional CDFs desired are approximated. After this, the "distance" between CDFs can be calculated using a certain metric, which provides values between zero and one. The greater the value, the more impact a certain parameter is said to have on the output. The converse of this also holds, where smaller values imply less parameter contribution to the output distribution. As with other density-based methods of sensitivity analysis, PAWN should be used when it is suspected that the parameters may influence output differently at different points of their range. Additionally, PAWN is very computationally efficient and typically allows for the execution of fewer simulations to achieve the desired results.

Fourier Amplitude Sensitivity Test

While the inner workings of FAST are quite mathematically intensive, the essence is that one can perform a Fourier transformation on the output of a model with respect to the parameters in question, where each parameter has a distinct interference-free frequency - if one observes high wave amplitudes at the frequency associated with a given parameter, one can say that it affects model output strongly [72]. FAST is another method that provides traditional sensitivity indices, and as such, it should be used in conjunction with other methods reporting on these metrics.

Random Balance Designs Fourier Amplitude Sensitivity Test

Random Balance Designs Fourier Amplitude Sensitivity Test (RBD-FAST) works by combining the FAST and RBD frameworks for sensitivity analysis [87]. The RBD method works very similarly to the FAST method, but rather than giving all parameters interference-free distinct frequencies, they are all assigned the same frequency. They are then randomly perturbed, and their interaction effects can be observed by examining properties of amplitudes and parameter distance after the output values are also re-ordered in increasing order with respect to a given factor. This process is then carried out for all other parameters to obtain the desired sensitivity indices.

RBD-FAST combines the two by splitting parameters into some number of groups per the RBD method and then performing the traditional FAST method on these groups, which are varied in order to gain higher-order interaction effects. RBD-FAST is a modification of the FAST method and should be used alongside it for the best results.

Delta Moment-Independent Measure

The Delta Moment-Independent Measure describes the contribution of a parameter to the output's total distribution rather than simply its variance, which is the case for most methods, including Sobol' analysis [8]. "Moment-Independent" here refers to the fact that it is not just considering a single moment (usually variance) but rather looks at the whole distribution. Traditionally, with this measure, a single value, "delta", is reported with values ranging from zero to one, where a value of zero means a parameter contributes nothing to output distribution and one means it contributes entirely to output distribution. While the Delta Moment-Independent Measure strictly refers to the delta value only, it is often presented with other sensitivity measures; namely, variance-based measures such as first and total-order sensitivity indices in order to gain a richer understanding of parameter behaviour [33]. Being based on parameter distribution, the Delta method aims to provide a greater understanding of parameter influence when their impact is more complex in nature.

Derivative-Based Global Sensitivity Measure

The Derivative-Based Global Sensitivity Measure (DGSM) is based on a slight modification to a value calculated in the Morris method, called the "Morris importance measure" [80]. The Morris importance measure first assumes that a function $f(X)$ (where X represents all parameters in a model) is differentiable with respect to all parameters. Then, partial derivatives with respect to each parameter are calculated and integrated over the parameter space. DGSM works by integrating over the *square* of the partial derivatives instead (called v_i). With this, a simple relationship can be

established between the total-order sensitivity indices and v_i. In particular, the total-order sensitivity index for one parameter must be less than or equal to some expression hinging on v_i, meaning that a small v_i implies low parameter importance. However, the authors note that the converse is not necessarily true and that parameters should not have their importance ranked based on v_i. While this value can be calculated directly in the case of some mathematical models, for agent-based models where there are no defined functions, these partial derivative values can be approximated by running a number of simulations with varying values.

Fractional Factorial

Fractional Factorial is a sampling method rather than a thorough sensitivity analysis method, and offers a framework for comparing and selecting parameters of interest in a model [73]. Fractional factorial sampling is a more popular and simplified version of full factorial sampling, which examines all possible individual and joint parameter configurations at all of their possible respective defined values (must be discrete levels rather than a continuous range). Fractional factorial sampling is based on the modeler's assumption that higher-order effects are negligible and that parameters can generally be tested in isolation. This vastly cuts down computation time when there are a large number of parameters and can lead to similarly accurate results if the aforementioned assumption is correct.

This sampling method can be exchanged with other common ones at the modeler's discretion. This is generally not the case, as it is, by definition, less accurate than some of the more common sampling methods that are now essentially baked into most sensitivity analysis methods, such as Latin Hypercube Sampling [82]. This is because

it is generally not the case that modelers are so constrained that they can only afford to run a very small number of simulations - modern simulations are typically not so complex that this is required.

High-Dimensional Model Representation

High-Dimensional Model Representation (HDMR) calculates new sensitivities using equations based on the same fundamental equations used in other methods like the Sobol' and Fast methods [46]. As such, variances of the components are estimated on their own and conditionally with respect to other components. What differs is how these sensitivity indices are defined. The sensitivity indices calculated represent the total, structural, and correlational contribution of a single parameter or of a subset of parameters. These are quite similar to traditional indices, but claim to tell a more descriptive story of how the parameters come together to shape a model. Also, HDMR performs the data collection and calculation necessary for the calculation of its sensitivity indices in a different way. Rather than creating some large number of samples, running simulations, and observing how the output changes as a result via some algorithm (which is done in Sobol' and Fast for example), HDMR operates by attempting to create a function $f(x)$ that represents the system in question (known as a "meta-modeling" approach in contrast to a "classical" approach) and running simulations in order to best approximate its unconditional and conditional variances through variance decomposition. This is done using a specific sampling method, random sampling high-dimensional model representation (RS-HDMR), in conjunction with an algorithm based on statistical F-tests. The main idea with HDMR is that this new $f(x)$ will be simpler in nature, and with the assistance of their sampling

method and F-test algorithm, the derivation of the variances will be faster than other methods while producing results that are essentially identical. HDMR also boasts of being able to handle cases where the input parameters are correlated or uncorrelated, which is said to be a problem that arises for certain other methods that only calculate one such measure.

Regional Sensitivity Analysis

Regional Sensitivity Analysis (RSA), while still being a form of sensitivity analysis, is more of a framework for imposing some other sensitivity analysis on [63]. RSA looks at the total range of values for each parameter and observes how model output changes within certain regions of the overall parameter range. In other words, parameter value ranges are split into some number of bins, where the influence of the parameter on the outcome of a simulation in a value range represented by a bin can be obtained via downstream sensitivity analysis.

Initially, RSA was paired with CDF estimation, where the CDF of outputs and conditional CDFs based on parameter values were estimated. From here, given certain output values, one could derive the range of input parameters that elicited certain outputs. RSA should be used if different regions of the *output* are suspected to be affected in different ways by the parameters in question.

Discrepancy Sensitivity Indices

Discrepancy Sensitivity Analysis was introduced as a means of creating an interpretable form of sensitivity analysis, given that the current environment of modeling does not see thorough sensitivity analysis being carried out (or even any sensitivity

analysis *at all* being carried out) [66]. The authors base their calculation of sensitivity indices on how well some arbitrary function seems to be able to map one given parameter to the output under parameter variations. The authors explain this visually using scatterplots. If there appears to be a relationship between the input and output, their relatively straightforward mathematical approaches based on output value distances from expected values should capture this information (in other words, if the points on a scatterplot are close to the function of best fit, then the parameter can be said to be relatively important). Overall, this method is not quite as robust and descriptive as some more common methods, but it aims to appeal to newer modelers who are not familiar with sensitivity analysis due to its simple visual explanation method.

2.5 Related Work

2.5.1 Mathematical Disease Models

Often serving as the basis for more complex agent-based models, mathematical models seek to explain the spread of disease from a compartmental standpoint. The states a person can be in (SEIR) relative to the disease and the rates of change of each of these states in a given population over time are modeled and can be readily tweaked to account for a variety of situations. Individuals themselves are not tracked in these models, but rather, the population as a whole is estimated on each time step. Significant work was done early into the pandemic in Sweden, furthering our understanding of the disease behaviour in a population with real, up-to-date data [9, 38, 79]. While these models performed well for certain tasks, the simplicity and generalizations made by the models produced results that were often not in line with observed data. Still,

this works acts as a base for the theory supporting the construction of the ODE models and agent-based model underlying functionality seen in this work.

2.5.2 Agent-Based Modeling Tools

Agent-based modeling, being a general strategy for modeling any system, has many possible applications spanning a wide variety of domains. As such, special tools and software have been developed to help modelers best represent the task they are working with. CloudSim [10] is a Java-based tool that assists users in modeling cloud computing architecture by way of having modern cloud computing systems built into its functionality such that modelers can explore the systems in great detail without having to actually interact with it directly. There are also agent-based modeling tools for specific fields in biology, like BSim [28], another Java based tool which exists to assist in the modeling of bacteria populations. While this deals in the realm of microbiology, there are also tools that specialize in modeling entire ecosystems, like Framsticks [44], a tool using its own custom language, which focuses on evolution and artificial creatures, and the Java-based BREVE [43] which fills a similar niche, but at a more general level. These biology-based tools I have just mentioned (BSim, Framsticks, and BREVE) also have spatial elements directly built into them, with BSim and Framsticks being 2-dimensional, and BREVE being 3-dimensional. This spatial element is one that many modeling tools incorporate, and is commonly used to categorize modeling tools. Another example of a spatial modeling tool, modeling 2-dimensional space, is UrbanSim [91], which uses a custom programming language and aims to model urban development for the sake of the city and environmental planning. While these models all have their place, the purpose of this work was to

use models that are highly general, such that the largest number of modelers possible could benefit from it.

This brings us to the modeling tools that are more general in nature, where modelers can use its functionalities to suit any task they have in mind. However, many of these general models still have special functionalities relating to some things - typically, in computational efficiency or extensibility. For example, MASON [49] is a commonly used tool based in Java that is general in its domain, but is particularly suited to handling very large-scale, multi-agent models. Another tool that boasts computational efficiency is Repast [49] (used in this work), which spans multiple programming languages, and is one of the most popular tools due to its wide language coverage and generalizability. Python is among the least covered in terms of support from modeling tools, possibly due to the lack of low-level specificity in comparison to other languages that allow for better optimization; however, tools like Mesa [51] (used in this work) do exist. We choose Repast and Mesa here due to the fact that they can be easily incorporated into Python and allow for simple disease-related models to be created.

In addition to modeling tools that are specifically for advanced researchers, tools do exist with a more educational spin or with the aim of being accessible to researchers who do not have the deep programming background that others do. NetLogo [88] (used in this work) is an example of these types of accessible models, with its software coming with a built-in, highly interactive user interface that allows for high-level analysis and parameter adjustment without touching any actual code. Some beginner software also exists for students with limited to no programming experience like StarLogo [71], where modelers can create simply models using “puzzle pieces” that

they can drag around and give properties to. SimSketch [7] is another of these tools, with the functionality of having user's drawings "come to life" in the models that they create. While these have their uses, the agent behaviours that modelers can elicit are not at the level of complexity when compared to NetLogo, and so they were not used for this work.

2.5.3 Geographical Disease Models

There are also a slew of tools and models that exist for the purpose of disease models, which have seen increased attention in recent years due to the COVID-19 pandemic being of high importance to disease researchers. Early into the pandemic, many older agent-based models were adapted to model COVID-19's spread and resistance to intervention methods [24]. While these sorts of models were arguably the most advanced at the time, they often had limitations with regard to population behaviour and the implementation of intervention methods [1]. Building on these ideas, popular models such as OpenABM-Covid19 and Covasim emerged, which have been used by public health officials worldwide to aid in decision support, taking detailed population behaviour into account and tracking how a variety of intervention methods affect the spread of COVID-19 [34, 41]. These models are highly complex, and were a source of inspiration for the models created in this work. They are seen as state-of-the-art COVID-19 models and are being consistently improved upon. GSAM is another powerful model with very high scalability, allowing billions of agents to act distinctly at the cost of specificity with regard to their environments, in contrast to Repast, which attempts to balance both of these concepts. This type of model is good for understanding broad disease dynamics but does not perform well when trying to

understand fine details such as the effects of regional population distributions [60]. There are also models like BioWar, where nefarious agents are introduced who act to intentionally spread disease, giving way to some interesting dynamics [14]. In addition to these, there are a plethora of other models that exist, which specialize in different tasks [3, 50, 78, 90] - many of them being open source and regularly updated.

There also exists a NetLogo disease model based in the town of Schull, Ireland [35] that acted as a base for the NetLogo model created in this work. It operates in a similar way, with agents having daily routines including day jobs, schools and stores that they move between. While the actual disease mechanics are slightly different (transmission is a fixed probability when agents occupy the same location, which is a simplification but allows for greater scaling properties), and the agents have slightly more complex behaviours, the model that they created is the most similar to the ones presented in this work.

2.5.4 Sensitivity Analysis Applications

Sensitivity analysis is a field that has seen much research in recent decades, with techniques based on concepts ranging from linear regression [42] to machine learning [25], whose applications span any domain due to their generality in nature [69]. While each method has its pros and cons based on assumptions they make and how they measure variable importance, I will focus this discussion on how sensitivity analysis has been used in relation to the problem this work aims to highlight - the task of identifying incorrectly initialized parameters.

The concept of using sensitivity analysis to inform public health officials on where to focus their attention is not new - Tarantola et al. [85] first introduced this as

a possibility in 2002. Haghnegahdar et al. [30] illustrate this via an end-to-end study on Canadian hydrological processes, where they create a model and perform sensitivity analysis on it, using the results to highlight where attention should be focused. However, they do *not* provide any steps into identifying whether those parameters are, in fact, incorrectly initialized or if they are already correct. There has also been work done into identifying the *quality* of sensitivity analysis results, with Saltelli et. al. [75] introducing "Sensitivity Auditing", which provides a set of guidelines that modelers and public health officials can follow to create the best possible analysis given what is available to them.

With respect to work that takes sensitivity analysis further, there has been research into concepts like linear regression, with Kleijnen [42] presenting the idea of tweaking variable values in order to obtain some desired output being presented. However, this only works in the case where the relationship between variables and output is linear, which is almost never the case for more complex models, hence the advanced work that has been done in the field of sensitivity analysis since (and in some cases, prior to) this. Other work examines the quality of fit with respect to specific variables through the use of logical systems. Claessen and Hughes [18] presented the tool called QuickCheck for Haskell in 2000, which serves to identify whether or not a user-created model is logically equivalent to some ground truth via the automatic generation of a series of test cases that must be successfully completed. This is useful in settings where the ground truth a user is trying to mimic can be readily converted into a logical system, and the kinds of test cases that fail can sometimes be used to highlight parts of the model being tested that are incorrectly specified. While this deals in the domain of logic rather than continuous value comparison and sensitivity

analysis, this work has been influential in the realm of model comparison, and this work can almost be thought of as a logical abstraction and simplification of the work presented in this thesis.

As far as directly identifying incorrectly initialized parameters, there does not appear to be any significant work done toward a tangibly implementable solution. There is, however, concrete work done into identifying problematic regions that should be focused on or parameter values that provide high-quality solutions, but nothing that *identifies* the location of these parameter regions themselves. For example, the concept of "robustness" has seen much discussion in the field of modeling and sensitivity analysis [69], in which the quality of certain parameter configurations are quantified. A robust configuration (in a simplified sense) implies that even with slight parameter perturbations the output, per what a modeler wishes to examine, does not change very much and that with the introduction of new data, the model's general results should still hold. McPhail et al. [53] provide a system of quantifying robustness in a digestible format rather than the complex and multi-levelled alternatives. Specifically, their work allows for the direct comparison of the robustness of parameter configurations, which is a step in the direction of directly identifying robust solutions.

There has also been some recent work in developing techniques to improve existing sensitivity methods or to examine them from different angles (separate from simply inventing new sensitivity analysis methods). Razavi and Gupta [68], for example, present a step after sensitivity analysis where they aim to consolidate the results of many methods into one value. This type of work has the possibility to improve the quality of sensitivity analysis across the board, regardless of the method used, and could possibly be something to consider in the task of identifying problematic

parameters. Another approach taken was by Mathur and VanderWeele [52], who discuss "confounding bias" (when external factors influence how a set of parameters affect a set of measures) and how that has the potential to change sensitivity analysis results. As the confounding bias becomes stronger, the results get increasingly "hazy". They introduce techniques to gauge how strong the confounding bias may be, and they also work towards being able to "untangle" this bias with varying degrees of certainty.

Chapter 3

Model Creation

In this chapter, I will describe the creation of the four disease models that are based in Kingston, Ontario. It is important to note that while these disease models do function similarly to many existing disease models, and the complexity of such models may be greater or lesser in comparison (per the modeler's discretion), the models developed here are done so for the sake of demonstrative purposes. They act as an end-to-end case study illustrating the development of geographical disease models using real data from the city of Kingston, Ontario, followed by detailed parameter analysis. By describing the exact construction of the four models used, it is the hope that the downstream analysis performed will become more commonplace with modelers having seen concrete, realistic, and detailed examples of agent-based disease models that the analysis can be performed on.

3.1 Geography Mapping

For each of the four models, the geography and agent initialization are generally the same - when there are differences in a specific model, that will be made clear in their respective sections. All of these disease models simulate Kingston, Ontario, using *only*

buildings. Other features like roads, parks, and public transportation routes, while publicly available, are *not* included here for the sake of model simplification and to capture only the areas where the majority of diseases see the most transmission (inside buildings).

Python was chosen as the base language for this work due to the extensive libraries offered with regard to geographical representations, visualizations, and sensitivity analysis. It was also decided that creating a tool allowing modelers to perform sensitivity analysis with ease would be best done via Python libraries. The first step in creating the geographical space was to input Kingston's geography into Python. This was done via the OSMnx package [6], which provides the coordinates and brief descriptions of all registered buildings (in addition to other unused features) in various cities provided by OpenStreetMap [5]. The buildings provided are broken into four categories as follows:

- Residential: "House", "Apartment", "Dormitory", or "Residential", encapsulating the leftover residential buildings
- Work: "Office", "Commercial", "Industrial", "Retail", or "Warehouse"
- Commercial: "Commercial" or "Retail"
- Education: "School", "College", or "University"

It is important to note that not all of these distinctions were *applied* in the models created. Specifically, all of the subdivisions in the "work" category were treated as one, marking locations where agents may work during the day. The same idea applies to the "commercial" category, where the "commercial" and "retail" subdivisions were

combined into one, marking stores that agents may visit. Finally, in the "residential" category, the "house" and "residential" subdivisions were combined due to the lack of specificity associated with the "residential" buildings (per their nature of being "leftovers").

For each model type, when generating a complete instance of the model, the user has the ability to input population specifications (up to a suggested maximum of 100,000, mimicking the population of Kingston). Specifically, the user has the ability to define the population of each of the three post-secondary schools in Kingston and of Kingston in its entirety, along with being able to define the exact number of residences/dormitories associated with the post-secondary schools. Depending on the values selected, a subset of the existing buildings will be used. This is done in order to avoid buildings becoming too sparsely or densely populated when varying agent counts are analyzed. The building subsets were chosen by utilizing pre-defined building capacities and removing random buildings of certain classes based on their overall contribution to housing space in the geography. For instance, "houses" and "residential" buildings are set to have a specific capacity ranging from 1 to 6. The housing capacities were chosen in line with Kingston census statistics detailing household sizes [12]. These buildings were removed such that enough houses remained to contain the specified population of 6 people per house. A similar process was done with apartments, but the capacity values for these buildings were chosen at a cap of 20 rather than 6. As previously mentioned, the number of dormitories that are used is defined by the user and their capacities are calculated by referencing the residence populations for each post-secondary institution in Kingston (Queen's University, St. Lawrence College, and the Royal Military College of Canada) [55, 56, 70].

3.2 Agent Generation

Before I can describe how agents are placed into the geography, I will first define exactly what features agents have associated with them. Since disease and behaviour dynamics are strongly linked to age, it makes sense to assign agents some sort of age. While there are many options, such as having continuous ages similar to how we think of age traditionally, I chose to discretize age into 10-year bins until the age of 70, at which point agents are said to be in the 70+ bin. One obvious effect of discretization is that the model and certain agent characteristics become easier to interpret (for example, vaccination rates being stratified by age group is generally easier to understand at a glance than having some complex equation governing it). Another effect is that age discretization, in some ways, has the potential to better represent how certain dynamics play out (in addition to data often being reported in 10-year bins). An example of this would be in masking and vaccination rates for children still in school. In educational environments, children may be subjected to these kinds of things at rates much different than the general population. Due to reasons such as this, modelers and data collectors tend to stratify by age group out of necessity. Thus, agents in this model vaccinate based on their age group, according to data from the Government of Canada [58]. So, upon creation of a given model, agents will immediately decide whether or not to vaccinate based on the probability associated with their age group, and they will remain in this state for the entirety of the simulation. The same process occurs for masking, although this is *not* tied to age, but rather is a parameter that is varied in the later sensitivity analysis.

Agents must also be assigned three to four separate buildings. First, all agents are assigned a home. For agents not said to be attending a post-secondary institution,

homes are chosen randomly from the subset of available accommodations, assuming the accommodation has not reached capacity. Agents who are attending a post-secondary institution are placed in a slightly different way. Clearly, these agents must be the only ones occupying dormitories, and so they are placed in these buildings first (i.e. before all other agents) until the buildings reach capacity. It is also the case that in Kingston (and other cities for the most part), residential buildings surrounding post-secondary institutions tend to fill up with students as opposed to non-students in a region. For this reason, after students are placed in dormitories, I assign each additional student to the building closest to the campus where they will be attending school, which is not at capacity. This is done *before* the assignment of agents that are not post-secondary students. By doing this, more realistic student housing areas are created, lending to more representative disease dynamics.

Agents are also assigned a "job" that they will attend during the day on weekdays depending on their age bracket. All agents between the ages of 0 and 19 are assigned to a school as their "job". The school that they are assigned to is chosen to be the one that has the shortest Euclidian distance from their home. *Some* agents between the ages of 20 and 29 are assigned a workplace, whereas others are assigned a post-secondary school. Then, all agents between the ages of 30 and 69 are assigned a traditional workplace. Workplaces are chosen at random for agents (i.e. not based on some other feature such as proximity to home) since Kingston is relatively small in size, and it is expected that agents may commute to work across the city. Students attending post-secondary institutions select a building on their home campus to be their "job" location. For agents in the 70+ age bracket, they are assumed to be retired and go to a commercial building during the day as their "job". This commercial

building was chosen to be the closest Euclidian distance to their home.

On the weekend, agents are made to go to commercial buildings during the day. They have two commercial buildings associated with them - one they go to on Saturday and one they go to on Sunday. These buildings are chosen to be the two closest in Euclidian distance to their homes. This process is applied to all agents identically, independent of age bracket. Now that I have discussed the assignment of agents and the properties that allow them to be placed and move about some geographical space, I can explain how disease functionality is injected into the models.

3.3 Disease Mechanics

In addition to the geographical attributes held by each agent, agents must also hold attributes relating to disease spread. While the detailed description of how disease spread occurs will be explained later in this section, agents must be assigned important attributes related to model creation. In the models, agents must always belong to one of the four SEIR classes. However, on model initialization, all agents will belong to either the susceptible or infectious class with an approximately 90-10 split, respectively (this initial configuration will not affect long-term equilibrium values). Agents infect each other probabilistically as the simulation plays out, causing movement between the SEIR classes. When an agent moves into a new class (excluding the susceptible class), the length of time that they will remain in that class is dependent on the class that they are in. On each time step, this value will decrease by one until it reaches 0, at which point the agent will move into the next SEIR class. Agents in the exposed class remain there for an average of 4.5 days (9 time steps) with a variance of 1 day; agents in the infectious class remain there for an average of 8 days

(16 time steps) with a variance of 2 days, and they remain in the recovered class for 7 days (14 time steps). The time spent in the exposed and infectious classes is in line with data observed from the initial main breakout of COVID-19 [57, 65, 92], and the recovered time was chosen arbitrarily so that an equilibrium could be reached. In reality, individuals remain in the "recovered" class for much longer than 14 days, often spanning months or years. However, due to the number of agents in each model, longer recovery times essentially always cause the disease to die out entirely rather than reaching an endemic equilibrium. Since equilibrium numbers are of high importance to modelers and the aim was to use these values in the downstream analysis, the recovered period had to be initialized in this way (against the alternative of *drastically* increasing the agent count, which would cause the sensitivity analysis process to take *significantly* longer).

Now that I have defined how agents move about the SEIR classes, I can discuss how disease is actually defined to spread in the four models. One common way that agent-based disease models are created is by having agents move about some space inhabited by other agents, spreading disease when they come near each other at some probability (i.e. having agents "bump" into each other and spread disease) [35]. In our models, I spread disease by looking at all agents that occupy the same building on a given time step and infecting susceptible agents based on the basic reproduction number, R_0. More specifically, I define the probability of one infectious agent spreading disease to one susceptible agent at a given location as follows:

$$\frac{(R_0 \text{ / (total time that the infectious agent will remain in the infectious class))}}{\text{(number of susceptible agents at the location)}}$$

By using the basic reproduction number in this way, one can "force" infectious agents to spread disease to an average number of susceptible agents equal to the value of the basic reproduction number. The logic behind this is as follows: if the probability of infecting another agent was static, then as the number of susceptible agents changed, the average number of agents that an infectious agent infects would also change. For instance, if there is 1 infectious agent in a room with 10 susceptible agents, having the infectious agent infect all 10 of the susceptible agents at some static probability would result in fewer susceptible agents contracting the disease than if that same infectious agent was in a room with 100 agents (by a factor of 10 if we are using some static probability). We adjust for this by dividing by the length of time the infectious agent will remain in its class and again by the number of susceptible agents at the location in question in order to have one infectious agent spreading disease to R_0 susceptible agents as often as possible (dependent on the susceptible population and their distribution among the locations). The basic reproduction number will be one of the key variables I vary in the upcoming sensitivity analysis.

Agents can also be masked or vaccinated at different rates, with the chance of masking being varied in this work for the purpose of sensitivity analysis. While the exact behaviours of masks and vaccines can vary and be quite complicated in nature, I simplify this by multiplying the above probability by one or more values between 0 and 1. These two factors are also variables that I will vary for sensitivity analysis. More specifically, if a susceptible agent is masked or vaccinated, I multiply by the masking factor and the vaccination factor once. If the infectious agent in question is also masked, then I multiply by the masking factor again. Importantly, I do not multiply the probability by the vaccination factor if the infectious agent is vaccinated

- if they are infectious, I assume that they spread disease normally.

The final element that plays into the spread of disease in the presented models is self-isolation. This is the last variable I will vary for sensitivity analysis and exists as a static value applied to all agents. When agents enter the infectious class, they have a chance of self-isolating equal to the value of this variable. If an agent is set to self-isolate, they will remain at their home location (i.e. they will *not* travel to any other location during the day or on weekends). However, they can still spread disease to other agents who share the same home.

It is also important to note that agent movement *between* locations is not captured - they can be thought of as "teleporting" from location to location. This kind of movement is the general concept for all four models, though its technical implementation, amongst other things, is slightly different between them and is outlined along with all other major differences in the following section.

3.4 Model Functionality

The above sections described agent generation and disease mechanics that are consistent throughout *all four* models. However, some models have subtle differences when compared to each other that are worth noting. These differences were sometimes necessary due to model restrictions, but other times intentional with the aim of creating models that perform the same end task with different approaches (i.e. creating some variety for the sake of realism). This section will also serve to provide more description of how the models work "under the hood".

3.4.1 Mesa

Geography

Mesa is arguably the most "simple" of the four modeling tools used here, and following this, the geography encoding is also quite simple. Since I do not consider agent movement between locations, having a virtual geographical space using building coordinates to re-construct Kingston in the final Mesa model is not necessary here - I can define locations in one of many arbitrary ways that work (per user requirements) so long as the structure supports the intended disease and agent mechanics. Since the buildings/locations associated with each agent are defined *before* model creation (which *does* depend on building coordinates), these coordinates do not need to be captured in the model in theory.

Thus, due to the relative simplicity of Mesa and the freedom of choice here, locations were defined to be "agent" objects just as the "people agents" were. The idea with this is that rather than looping over all "people agents" and having them move between locations and spread disease individually, all "location agents" could be looped over instead. With this, disease could be spread all at once by looping over the "people agents" at a given location *depending on the day of the week and time of day* and then "moving" all "people agents" en masse to another location. As such, each location was defined to have an associated list of agents that, at any point in the simulation, may be at the location. So, a home will have, as a list attribute, all of the agents living there at "night" just as job and store locations will have all of the agents that go there during the "day". When all of these lists are in place, the locations that should be "active" at any given point (i.e. home locations at night and work/school/store locations during the day, depending on the day of the week) can

simply be iterated through, and disease can be spread between the "people agents" contained in these location lists.

However, this functionality requires some careful implementation when it comes to stores. Since "people agents" have two different stores associated with them (one for Saturday and one for Sunday), simply iterating through all "people agents" at each store location on a given time step would result in agents being in "two places at once", or double counted. To counteract this, I can store "location agent" IDs associated with each "person agent" as one of their attributes and run a simple check to see if the current "location agent" being iterated over has the same agent ID as what is shown on the side of the "person agent". If there is a match, then I can begin the disease spreading process including them. So if a given store is being iterated through on a Saturday, all "people agents" are looked at who belong to this "location agent" and only the agents who have the same "location agent" ID in their "store 1" attribute as the one that is currently being iterated over are incorporated in the disease spread calculation.

Apart from this way of handling locations, the Mesa model operates quite strictly, adhering to the process outlined in the previous sections. The values of interest (SEIR values) are updated on each step, where they can be directly accessed from outside of the model and stored in any desired format.

3.4.2 Repast

Geography

Repast's key advantage, when compared to many other agent-based modeling tools, is its ability to divide the computational load among available and specified CPU cores.

This is done automatically, but *only if* there is a properly defined geographical space in the model. More specifically, once a geographical space is defined with its boundary lines, Repast will divide the region into sub-regions of equal size to the number of CPU cores that are to be used, specified at runtime. Each CPU is then said to be "responsible" for everything that happens within their assigned sub-region. When agents move between sub-regions, they are essentially deleted from one CPU and re-created on another CPU. The exact process involves a few more internal steps not handled by the user and can be found explained in detail in the Repast documentation [19].

In order to facilitate this for our model, a continuous geographical space is defined with dimensions based on the coordinates from the locations used. In order to ensure that no space is wasted, the x coordinates of the continuous space range from the smallest x coordinate observed in the list of locations to the largest x coordinate in the list of locations. The same process is applied to the range of y coordinates. One minor complication of this is that the geographical space can only be defined for values greater than or equal to zero. To combat this, I simply multiply the x coordinates by -1, since all x coordinates are negative in Kingston. I also multiply out the decimals (i.e. I multiply all coordinate values by 100,000) for ease of further numerical manipulation, though this step is not required.

After the geographical space is defined, I can simply have agents move between locations by using the built-in "move" command in Repast, ensuring that they are moved to the proper location per the day of week and time of day. When they are moved, they also need to be "restored" if they end up changing CPU cores. There are also "location agents" that have a list of "people agents" currently at that location

that updates on each time step. When disease is to be spread, each "location agent" is stepped through, spreading disease to only the "people agents" currently at that location.

Data Output

Repast4py most commonly outputs data via the built-in "logging" feature, where the specified data is output to a CSV file and saved locally at the end of a simulation. While this makes the process of outputting data organized in a sense, it also complicates matters when multiple instances of a simulation are being run, and multiple output CSV files need to be read back in for further calculations, as we will see when conducting sensitivity analysis.

3.4.3 RDDL

Geography

Like in the Mesa model, a geographical space being created here does not make sense for this RDDL model. Agents simply have IDs associated with their home, job, and store locations, whereas another value indicating an agent's current location flips between them when they "move" locations.

Disease Spread

Disease spread is facilitated using the same general mechanics as in the other three models, but since RDDL does not have the same level of manipulation from a developer's standpoint, this was done in a less efficient manner. In this model, when each agent is stepped through, all of the agents occupying a location must be counted

before disease spread can actually happen. This is in contrast to the other models, where a running count and list of agents are held with each location. This increases the time complexity quite substantially and causes other aspects of RDDL (such as the introduction of actions) to scale more poorly than it might otherwise.

Data Output

Calling the RDDL file directly from Python gives us access to the current "state" of the simulation, listing all agents and their associated statuses. However, this is not done in a way that makes it easy to retrieve attribute counts and other details - the primary goal of RDDL is to create action policies rather than simply tracking how agents evolve over time. In the "state" variable retrieved from the RDDL environment on each step, all agent fluents defined in the Conditional Probabilistic Function (or "CPF") block of the RDDL domain are recorded as key-value pairs where the key is equal to the name of the fluent plus two underscores followed by the agent ID, and the value is equal to the value of the fluent. All of the desired values can be extracted from this for downstream analysis.

Planner Actions

The main idea with RDDL is to use a planner to generate actions with the intention of maximizing some user-define reward function. However, no actions need to be taken for our base model in RDDL. As such, the domain does not have any planner actions defined, and the reward function is set to a constant value of 0 that will never change as the simulation progresses. This, in reality, is an unrealistic scenario, but this model was included to show that this sensitivity analysis process could be applied

to a variety of different models even if their typical usage is quite different.

3.4.4 NetLogo

Geography

Geography in NetLogo works differently from many other agent-based modeling tools. Rather than having some space that can be defined per the user requirements, NetLogo space operates through discrete regions called "patches" that come together to form a discrete space. These patches are treated as objects and function as a space for agents, called "turtles" in NetLogo, to exist on. The key limitation here (assuming NetLogo 2D is being used, as is the case here) is that no two patches can occupy the same location at the same time. This seems obvious but creates a limitation when one is trying to map something that is not two-dimensional. In the case of our disease models, there are many buildings that occupy the same coordinates - residential buildings on top of commercial or other residential buildings being the most common occurrence. To combat this, I created a sort of hierarchy of importance for the four kinds of buildings - residential buildings were deemed the most important, followed by educational buildings (which rarely, if ever, have overlapping coordinates), commercial buildings, and then workplaces. From here, the patches could be defined normally, using the maximum and minimum coordinate values observed as boundary points for the 2D NetLogo space.

Data Output

In NetLogo, it is often the case that data is never truly output to the user for downstream usage but is simply displayed on a simple graph directly in the NetLogo interface. However, it is possible to output data to files if necessary. It is also possible to call NetLogo simulations from other software (which is the case for this model, as it is called via a Python script), where global variables in NetLogo can be chosen as ones that will have their values recorded each time step. In Python, dictionaries containing the names of variables and a list of the corresponding values recorded at each time step are output. This is a fairly simple process and allows for straightforward data manipulation post-simulation.

3.5 Model Execution and Data Collection

Sensitivity analysis on the scale being operated on requires significant computing power - running the 11 forms of sensitivity analysis for tens of thousands of trials, each across several models, would be very demanding for a single machine. Thus, the simulations were all run on a high-performance CPU cluster provided by the Queen's University School of Computing, facilitated by the MuLab [1]. With this, running all simulations across all models in parallel can be completed in a matter of days compared to the months it would take to run them sequentially.

In running these simulations, there are a few points at which the information calculated and gathered may be useful to modelers in downstream analysis. So, at these points of interest, the relevant data was saved to CSV or pickle files. Specifically, after running the model for some defined number of trials for a specific method of

[1] https://mulab.ai

sensitivity analysis, all values feeding into measure calculations at each time step for every trial were saved in a CSV file, and the actual measure values for each trial were also saved in a CSV file. As for the sensitivity analysis itself, all parameter configurations were saved in a pickle file, and the analysis results were saved in a pickle file for each measure.

Chapter 4

Sensitivity Analysis

In this chapter, I detail the sensitivity analysis, outlining exactly how it was performed after preliminary model exploration. The process of sensitivity analysis is not always straightforward, and the results directly depend on the approach taken. By examining a wide variety of sensitivity analysis methods and exploring how they were applied, this chapter serves as an example of how this sort of analysis can be performed given any sort of disease model. As a novel contribution, I also outline an algorithm to be employed *after* traditional sensitivity analysis that aims to solve the problem of insufficiently accurate parameter values (that being cases where parameter values may be presented as a wide range or when they are otherwise incorrect). This straightforward algorithm serves to enhance the existing parameter analysis methods used by modelers, and addresses the issue surrounding a lack of data that define particular parameter values.

4.1 Methods Used

The methods used were outlined in section 2 of this thesis, but for ease of reading, they will be listed here again. The methods and their abbreviations where relevant

include:

- Sobol' Sensitivity Analysis
- Method of Morris
- PAWN Sensitivity Analysis (PAWN)
- Fourier Amplitude Sensitivity Test (FAST)
- Random Balance Designs Fourier Amplitude Sensitivity Test (RBD-FAST)
- Delta Moment-Independent Measure
- Derivative-based Global Sensitivity Measure (DGSM)
- Fractional Factorial (FF)
- High-Dimensional Model Representation (HDMR)
- Regional Sensitivity Analysis
- Discrepancy Sensitivity Indices

4.2 Comparison Points

Of the many possibilities, one common area of research in disease modeling is to gain a thorough understanding of how the number of infectious individuals may progress and behave in a disease outbreak. If one chooses to examine this measure as it changes over the course of simulations, the next issue is in deciding what *part* of the infectious population to examine. Depending on the goal in mind, there are many ways this can be approached. For the sake of simplicity and demonstration, I will

select three points of analysis for the number of infectious agents in simulations: the equilibrium number of infectious agents, the peak number of infectious agents, and the time it takes to reach the peak number of infectious agents. By performing sensitivity analysis on these measures, we will be able to gain a rudimentary understanding of how the parameters under examination affect the number of infectious agents overall with respect to our model.

I also consider other measures in addition to the number of infectious agents over time. To supplement the above analysis, I perform sensitivity analysis on the equilibrium values for the number of susceptible, exposed, and recovered agents. While it may be beneficial to place the other three disease classes under the same level of scrutiny as the infectious class here, the purpose of this analysis is primarily demonstrative - this work acts as a template and proof of concept for the sort of analysis that can be (and should be) performed on a model, and how those results could be interpreted. The final two measures I look at are the number of masked and vaccinated agents in a given simulation. While the number of masked agents is obviously a direct consequence of the parameter controlling the rate of masking, this measure can be used to provide an example of the parameter verification process. The vaccination count should not change *at all* as all parameters are varied and act as an example of a measure that is entirely independent of parameter space.

4.3 Initial Model Exploration

In order to gain an understanding of how these models compare to each other by way of simulation output, I ran the Mesa, Repast, and RDDL models 10,000 times with the same parameter configuration and noted some basic statistics and behaviours

that arose as a result. In the case of the NetLogo model, I ran it 1000 times only due to computational limitations. Running one simulation for 500 time steps takes anywhere from 165 seconds to 300 seconds, varying at a seemingly random rate. This means that running 10,000 trials would take anywhere from 19.1 days to 34.7 days, which was deemed unreasonable. With all models, I track the counts of agents in each SEIR class and I also look at the mean and standard deviation values observed at each of the above measures. This preliminary exploration helps to understand how these models operate relative to each other, and it also helps to gain insight into the volatility of certain measures. It can be important to understand this measure volatility, as a measure that is very non-volatile under a specific parameter configuration may be especially useful as a measure when attempting to determine correct parameter values - if the ground truth reports some value for this non-volatile measure, attempting to attain this value from simulations may uncover more accurate parameter configurations (assuming that the measure in question is sufficiently sensitive to changes in parameter values, which is at the modeler's discretion). Figure 4.1 shows the SEIR graphs after running 10,000 trials on the Mesa model with the following parameter configuration (where the parameters listed are the ones that will be varied in sensitivity analysis): the isolation rate of agents was set to 0.3, the basic reproductive ratio was set to 6, the chance of an agent masking was set to 0.7, the infection factor (factor by which I multiply the chance of infection at any given moment) from masking was set to 0.8, and the infection factor from vaccinating was set to 0.4. The same graphs for the Repast, RDDL, and NetLogo models can be seen in appendix B in Figures B.1, B.2, and B.3 respectively, and are all similar but with some minor differences - specifically, the Mesa and Repast models are essentially

identical, while the NetLogo and RDDL model have some important differences that will be discussed after analysis of table 4.1.

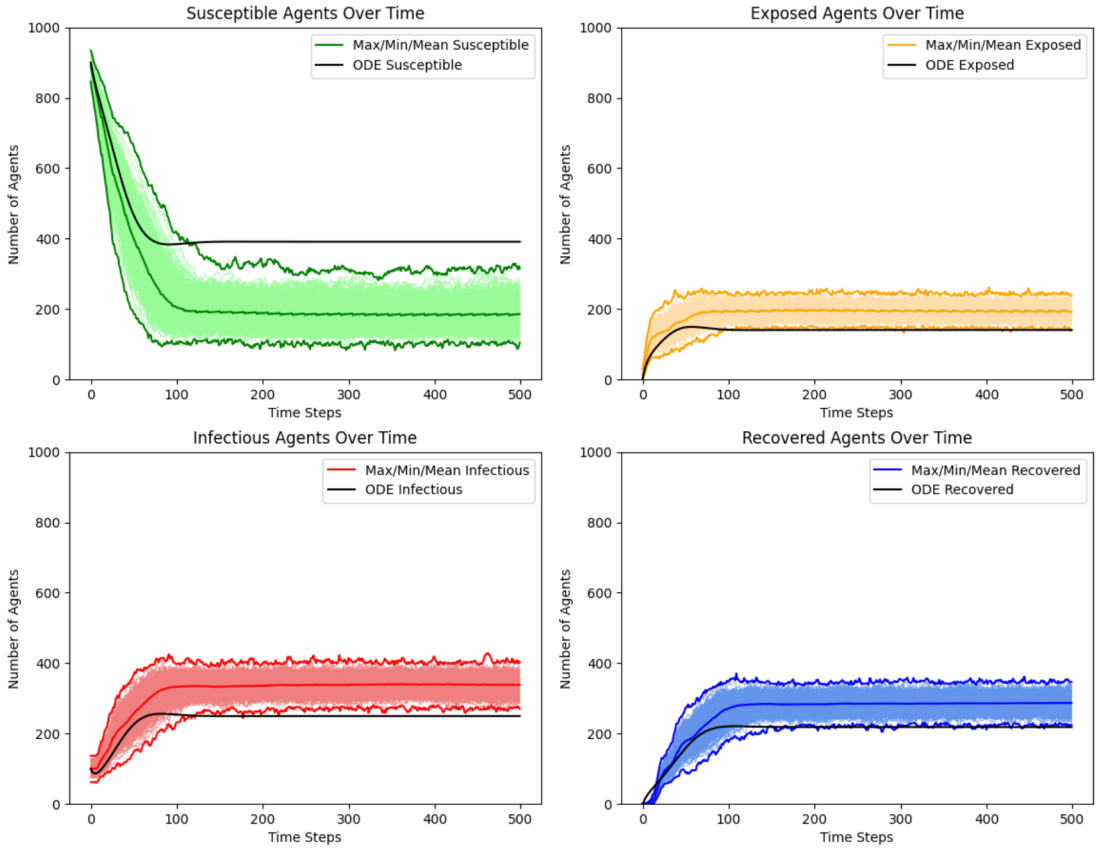

Figure 4.1: SEIR graphs for the base parameter configuration on the Mesa model

Each graph here shows the maximum, minimum, and mean values observed at each time step over all 10,000 trials in a darker colour, with the outputs for 1000 trials shown in a lighter colour. On each graph is a black curve, which represents the transposition of this agent-based model to a mathematical model. We can clearly see that its output is quite different when compared to the agent-based model output, illustrating how mathematical models cannot always accurately represent agent-based models. While behaviours such as masking and vaccinating can be somewhat

achieved in the mathematical model via scaling the chance of infection according to the "average person's" masked/vaccinated status, there are some elements that simply cannot be captured. The most obvious difference is that the agent-based model incorporates geography, leading certain agents to interact with each other at rates that cannot be captured in the mathematical model without significant work, if possible in general cases at all. Another element that cannot be simply captured in the mathematical model is the concept of self-isolation. In our agent-based model, agents who self-isolate will remain in their homes and only infect the other members of their household. This parameter significantly impacts model output (as well as will be seen from sensitivity analysis results), and, like the geographical element, this nuanced behaviour can be captured simply in mathematical models. The exact ODE model superimposed on the graphs is shown below:

$$\begin{aligned}
\frac{\mathrm{d}S(t)}{\mathrm{d}t} &= \upsilon R - \beta\frac{SI}{N} \\
\frac{\mathrm{d}E(t)}{\mathrm{d}t} &= \beta\frac{SI}{N} - \gamma E \\
\frac{\mathrm{d}I(t)}{\mathrm{d}t} &= \gamma E - \delta I \\
\frac{\mathrm{d}R(t)}{\mathrm{d}t} &= \delta I - \upsilon R
\end{aligned}$$

Here, υ is 1/14, γ is 1/9, and δ is 1/16, matching the average speed at which an agent moves between the classes defined in the agent-based counterpart. In capturing the rate of transmission, the SI terms just need to be divided by N since we are looking at the population in aggregate. The key term here that is attempting to mimic the agent-based model is β. In using the formula $R_0 = \beta/\gamma$, β can rearranged

for, such that $\beta = R_0 * \gamma$. Since γ is equal to 1/14 here, β can be set such that $\beta = R_0/14$. Determining the value of R_0, in this case, required a bit of work - it is not simply 6 (like in the agent-based model) since I am attempting to capture masking and vaccinating as well. To capture this, the disease dynamics and how they operate in the agent-based model must be referred to. When one agent is spreading disease to another, either none, one, or two of them are wearing masks, and either none or one of them is vaccinated. Thus, the value of R_0 here can be multiplied by the average infection factor for each agent brought on by masking and vaccinating, assuming these possible cases. The equation here is as follows:

```
R_0 = 6 * (((chance of masking * mask infection factor ) +
        ((1 - chance of masking) * 1)) ^ 2) *
        ((chance of vaccinating * vaccine infection factor) +
        ((1 - chance of vaccinating) * 1))
```

Plugging in the values used results in an R_0 value of around 2.237, which can be divided by 14 to get a β value of around 0.160. Using this value produces the curves shown in Figure 4.1. Table 4.1 shows the mean and standard deviation for each measure across all four models.

We see that the Mesa and Repast models produce very similar results, with only slight differences between the two that may simply be attributed to the number of trials that were run. However, when looking at the RDDL model, we notice that the values reported are not simply the same as Mesa and Repast but scaled by a factor of 10. This could be for a number of reasons, with agent spread in geography being a primary possibility. Since the number of agents is so low, how they exist within the world may be quite different when compared to the Mesa and Repast

Measure	Mesa	Repast	RDDL	NetLogo
Susceptible Equilibrium	μ =184.6, σ =1.4	μ =188.7, σ =1.9	μ =40.5, σ =1.4	μ =330.8, σ =5.5
Exposed Equilibrium	μ =194.3, σ =1.4	μ =193.1, σ =1.9	μ =15.5, σ =2.5	μ =157.3, σ =7.6
Infectious Equilibrium	μ =339.1, σ =0.6	μ =338.8, σ =1.2	μ =25.8, σ =1.7	μ =273.0, σ =3.1
Recovered Equilibrium	μ =285.9, σ =0.7	μ =283.3, σ =1.6	μ =22.1, σ =0.4	μ =240.0, σ =0.7
Infectious Peak	μ =380.8, σ =9.3	μ =379.2, σ =9.5	μ =40.7, σ =6.1	μ =448.5, σ =18.4
Infectious Time to Peak	μ =300.8, σ =118.3	μ =325.9, σ =110.8	μ =254.4, σ =136.6	μ =18.6, σ =4.4
Masked	μ =702.0, σ =14.6	μ =702.0, σ =14.5	μ =72.0, σ =4.9	μ =700.3, σ =14.5
Vaccinated	μ =824.0, σ =12.2	μ =824.0, σ =12.2	μ =85.0, σ =4.2	μ =833.4, σ =10.6

Table 4.1: Statistics associated with examined measures across all four models with static parameter configuration as follows: isolation rate = 0.3, basic reproductive ratio = 6, chance of masking = 0.7, mask infection factor = 0.8, vaccine infection factor = 0.4.

models. It is also the case that such low agent counts can cause certain effects to be pronounced relative to the scale of the model. For instance, in smaller agent count simulations, it is more likely that the infection dies out completely when compared to higher agent count simulations. The NetLogo model performs quite differently from the Mesa and Repast models, seeing many more susceptible agents at equilibrium and fewer agents in the exposed, infectious, and recovered classes. This could be due to the necessary geography simplifications implemented, but it could also be due to some internal workings of NetLogo that are causing a bias in some direction. In the case of searching for measures with low standard deviations that may be sensitive to parameter changes, we see that all of the values at equilibrium appear to fit this

description. This type of behaviour might serve as an indicator to modelers that attention could be put on them when attempting to identify correct parameter values.

4.4 Stopping Point Selection

In order to perform sensitivity analysis model output for the vast majority of techniques, a relatively large number of simulations must be run. In general, the larger the number of simulations that are run, the more accurately the sensitivity analysis measures reflect the true nature of the system. However, at a certain point, very little information can be gained from running further simulations. It is also important to note that in many cases, the *order of importance* of parameters can be determined by the sensitivity analysis methods at quite a low number of simulations. It is only the precise values that are being refined, though this can lead the order of parameters to change if they are nearly equally important (in which case, they should generally be treated as having equal contribution anyway, since these sensitivity analysis methods are not perfect). In any case, simulations were run for all sensitivity analysis methods (except for the Fractional Factorial method and the Regional method) using the Mesa model, ranging the number of trials from roughly 1000 to 5000 on 1000 trial intervals and from 5000 to 75000 on 5000 trial intervals (some methods cannot reach those values exactly due to the required sampling methods) and the sensitivity results were tracked accordingly. An example of this can be seen in Figure C.6, and the others can be seen in the appendix C. Note that Figure C.6 shows the Delta method because its sampling method allows for exact trial counts to the thousand, and because its sensitivity value that we analyze, "Delta", is nicely interpretable. With methods like Sobol' analysis, though its output is easily interpretable, the sampling method used

does not allow for scaling up the trial count by 5000. It also shows the results for only the "isolation rate" parameter, but the other parameters produce graphs that look much the same. The other disease models perform quite similarly, so similar results can be expected for this analysis.

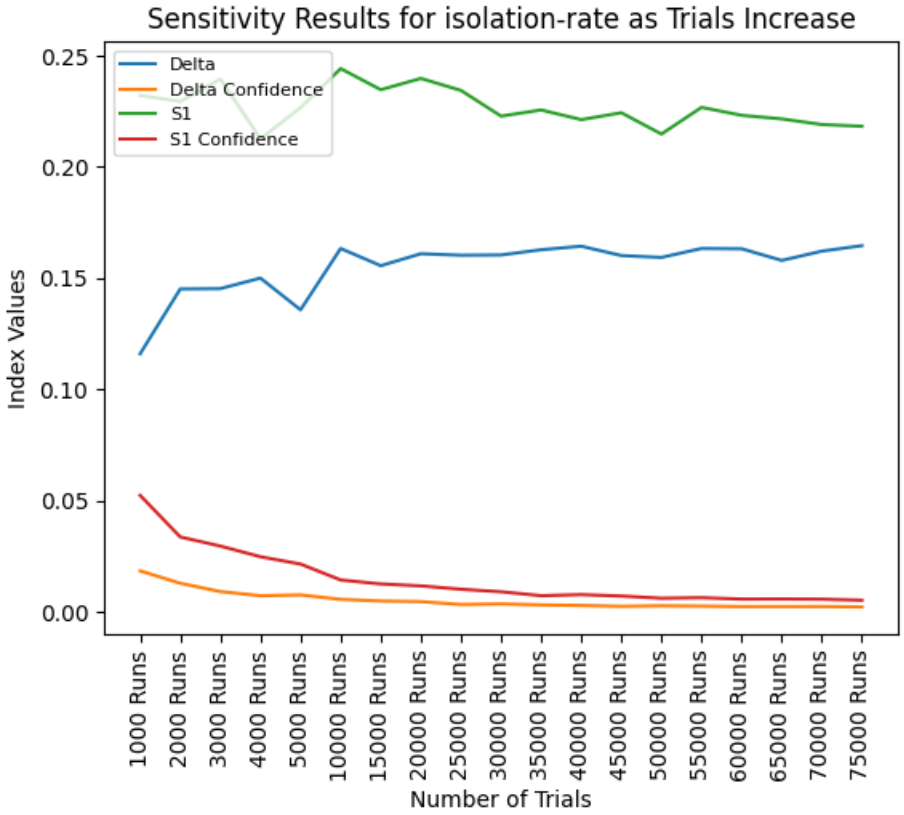

Figure 4.2: Isolation rate sensitivity results varying trial count for the Delta method. In selecting stopping point, we examine the "Delta" value, shown as the line in blue.

Though one will not *worsen* the results by running "too many" simulations, in the way that deep learning models can overfit to datasets, it becomes nonsensical at a point, which varies from model to model depending on stochasticity and overall model complexity. This concept also varies from sensitivity analysis method to sensitivity

SA Method	Index	Stopping Point
Sobol' Analysis	ST (Total Order Sensitivity)	98304
Method of Morris	Mu Star (Absolute Value of Mu)	45000
PAWN	Mean	30000
FAST	ST (Total Order Sensitivity)	30000
RBD-FAST	S1 (First Order Sensitivity)	35000
Delta	Delta	30000
DGSM	DGSM	75000
HDMR	ST (Total Order Sensitivity)	35000
Discrepancy Analysis	S Discrepancy	35000

Table 4.2: Sensitivity analysis methods and their associated stopping points relative to the index chosen.

analysis method, and the point of minimal gains should be analysed in all cases to ensure that the sensitivity analysis results in the best possible approximations. This point is to be defined by the researcher depending on the level of precision needed.

In our case, I define the stopping point as follows. First, I select a sensitivity index to analyze for a given method. Given that many methods have more than one potentially important value that they report, we must choose one to use as a reference. Then, at each trial count, I calculate the standard deviation of all of the previous index values, including the current one. If the absolute value of the current value subtracted by the previous value is less than or equal to the standard deviation, I increase a counter (which starts at zero) by one. If the counter reaches three, then we have reached our stopping point and will use that trial count in sensitivity analysis. If the absolute value of the difference in values is greater than the standard deviation, then I reset the counter to zero. The stopping points and indices found via the analysis on the Mesa model are shown in table 4.2.

Note that the Fractional Factorial and Regional methods were not used here.

The FF method does not allow for the variation of trial counts (its trial count is always 16), so this sort of procedure would simply not make sense. In the case of the Regional method, doing this sort of analysis is also somewhat nonsensical. The Regional method provides more of a framework for performing sensitivity analysis rather than acting as a method on its own. Theoretically, any sensitivity analysis method could be applied to the partitioned parameters laid out by the Regional method. Another complication is that there are many points of sensitivity index calculation in the regional method - it is not just one per trial count, but rather it is equal to the number of bins set by the modeler performing the analysis. As such, this form of stopping method decision is not readily applicable to it in the form it exists in with the SALib library. As such, its trial count for upcoming sensitivity analysis was set to be equal to 75,000 (the maximum per the range tested).

4.5 Parameter Analysis

The goal of this research is not only to provide some sort of framework for performing sensitivity analysis on agent-based disease models but also to outline *how* this can be used in decision-making. Take the following measures that we perform sensitivity analysis on:

- Susceptible Agents (equilibrium count)
- Exposed Agents (equilibrium count)
- Infectious Agents (equilibrium count, peak count, and time to reach peak)
- Masked Agents (static count)

- Vaccinated Agents (static count)

If we have a general understanding of parameter importance with respect to all of these measures, then we can make a more informed deduction when it comes to knowing why values output by the model may differ from known ground truths. For instance, let us assume that we have some known ground truth dynamics that capture all of the above measures. Now, assume we have some other dynamics that do not quite match this ground truth in some ways (whether this be the mask count being off, the susceptible agent count at equilibrium being off, or some other measure being off). If we know the parameter importance for each measure, then we can observe which measures are perturbed and which are not, and reason that the perturbation is likely a result of some subset of parameters not being correctly initialized.

The exact process for performing this sort of analysis is as follows: Let us assume that we have some ground truth values for each measure and assume that we have run a number of simulations on some static parameter configuration such that we can reasonably assume some mean and standard deviation for each measure. Let us also assume that we have gathered parameter importance for all measures and for some sensitivity analysis method. We can normalize all values for each measure to be between 0 and 1, where 1 marks the most important parameter, and 0 marks the least important parameter. Then, for each parameter, we set an initial score of 0, and we compare all measures between the ground truth and the generated results. If the ground truth values are greater than or equal to one standard deviation away from the generated results, we add the parameter's sensitivity value for that measure multiplied by the number of standard deviations away from the ground truth to the parameter's score. If the ground truth values are less than one standard deviation, then we add

1 minus the parameter's sensitivity value to the sensitivity score. The logic here is that if a measure is significantly perturbed, we give higher scores to parameters that have a big influence on its value. If a measure is in line with the ground truth, then we give a very small score to a parameter that affects that measure. Once we have gone through this process for all parameters, we can sort them in descending order to produce an ordered list containing estimates for which parameters are likely to be incorrect (the parameters at the top of the list with the highest scores are the most likely to be incorrect, while the parameters at the bottom of the list with the lowest scores are the least likely to be incorrect). A general symbolic definition of this process can be seen below.

Assume we have ground truth values for measures denoted as GT_1, GT_2, ..., GT_n and simulated measure values denoted as S_1, S_2, ..., S_n where n is the number of measures. Also, assume that we have some sensitivity matrix SM where the rows P_1, P_2, ..., P_m represent the parameters used, M_1, M_2, ..., M_n represent the measures used, and each cell contains the normalized sensitivity values for each parameter on a given measure. Then I assign a score to some parameter P_i as:

$$SCORE(P_i) = \sum_{|S_j - GT_j| \geq \sigma(M_j)} \left(SM_{(i,j)} * \frac{|S_j - GT_j|}{\sigma(M_j)} \right) + \sum_{|S_j - GT_j| < \sigma(M_j)} (1 - SM_{(i,j)})$$

Here, $\sigma(M_j)$ must be a non-zero standard deviation calculated from the simulated outcomes. This algorithm provides modelers with the capability of taking their sensitivity analysis results and applying them directly with the aim of further highlighting parameters that may require additional research. Given that this algorithm is *not*

computationally intensive relative to the overall sensitivity analysis process, it provides modelers with further concrete information using data that they have *already* retrieved and calculated by way of sensitivity analysis.

Chapter 5

Evaluation

5.1 Sensitivity Analysis

The sensitivity analysis process was carried out in full for both the Mesa and Repast models. These models are very computationally efficient compared to the RDDL and NetLogo models, and so running upwards of 100,000 trials on these models only took approximately three days. For the RDDL and NetLogo models, this would take *significantly* longer. It was not deemed necessary to carry out the full analysis of the NetLogo model when the overarching process is identical to that of the other models. In the case of the RDDL model, running 75,000 trials would take around 12 days, so it was run for just 15,000 trials for all methods.

The numerical results of the analysis on the measures in the Mesa model using the Sobol' method (total sensitivity index) can all be viewed in Table 5.1, with the tables for the other methods for the Mesa, Repast, and RDDL models found in Appendix A.

We chose only to focus on the Mesa model here and for all future discussions in this chapter, as the other models produce very similar results and the same concepts

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.366	0.644	0.205	0.026	0.055
Exposed Equilibrium	0.369	0.644	0.207	0.026	0.056
Infectious Equilibrium	0.365	0.643	0.205	0.026	0.055
Recovered Equilibrium	0.364	0.644	0.205	0.026	0.055
Peak Infectious Count	0.242	0.648	0.206	0.032	0.059
Time to Reach Peak Infectious Count	0.831	0.919	0.717	0.449	0.518
Masked Count	0.008	0.008	0.008	0.998	0.008
Vaccinated Count	0.984	1.008	0.998	0.997	0.980

Table 5.1: Sobol' Analysis results (total order sensitivity index) from the Mesa model for all measures and parameters.

presented can be applied to them directly. The Sobol' method was examined here due to its popularity and interpretability, but the results for the other methods are largely similar. For example, if we look at the infectious equilibrium measure (one of the more complicated and sensitive measures), the Sobol' method produces results shown numerically in Table 5.1, and also visually in Figure 5.1. If we look at similar sensitivity graphs shown for the Morris, Fast, Delta, and FF methods as examples in Figure 5.2, we see that the results they produce is quite similar when examining parameter importance relative to each other.

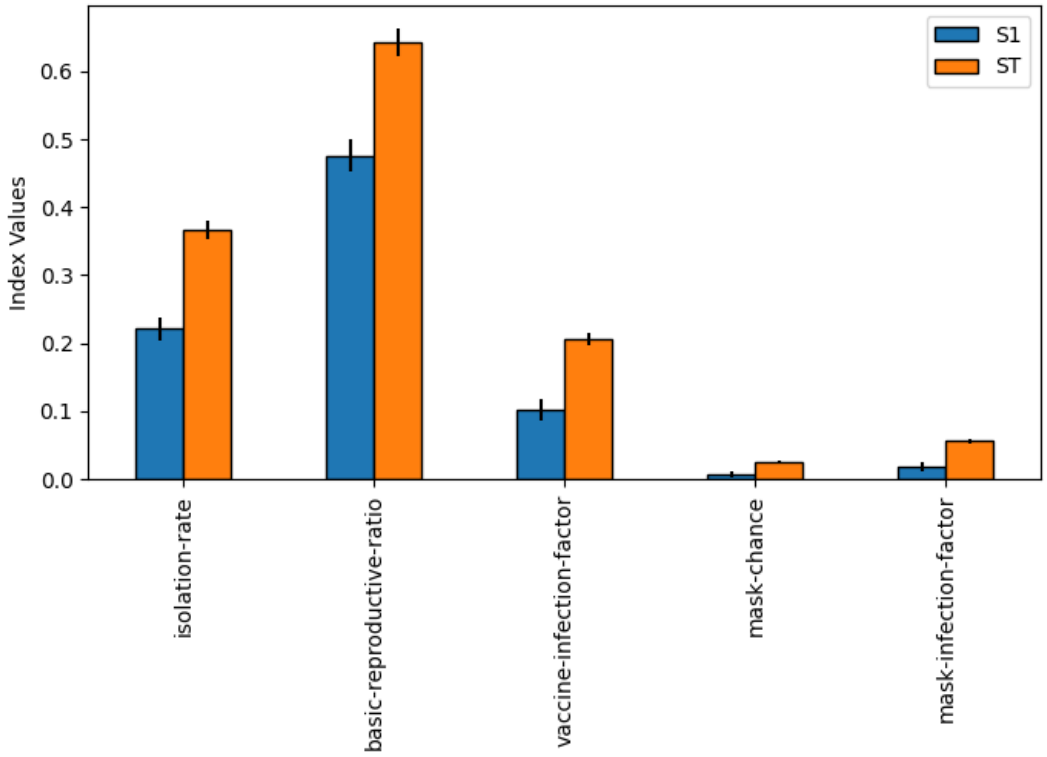

Figure 5.1: Sensitivity graph for Sobol' analysis on the Infectious Equilibrium measure via the Mesa model

While the results are not *exactly* identical, per their respective sensitivity measures, they are indeed very similar, marking the isolation rate, basic reproductive ratio, and vaccine infection parameters as being more important than the mask chance and mask infection factor parameters. The ordering of importances is also unchanged (although the magnitude of importances may differ slightly), except in the case of the FF method, where the mask chance and mask infection factor parameters are switched relative to the other methods. This is an example of why it is important for modelers to do more than one form of sensitivity analysis where possible, seeing as how the order and magnitude of the calculated parameter importances may vary slightly. In cases where parameters all similarly influence a given measure, this effect

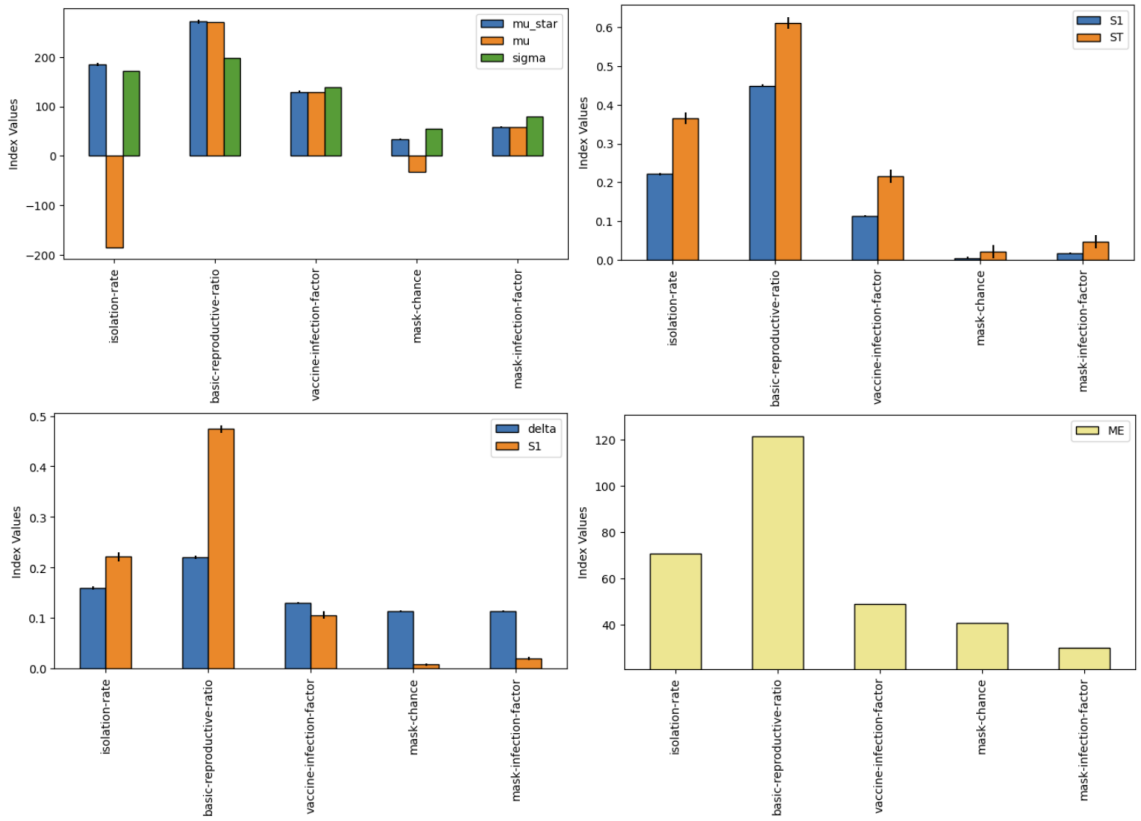

Figure 5.2: Sensitivity graph for Morris (top left), Fast (top right), Delta (bottom left), and FF (bottom right) analysis on the Infectious Equilibrium measure via the Mesa model

is especially pronounced.

We can also look at some more basic properties related to how output changes with different parameter configurations. With these methods, unrelated to sensitivity analysis, modelers can gain a preliminary understanding of how parameters affect model output with minimal time investment. For instance, let us examine four scatter plots, shown in Figure 5.3 obtained after running 30,000 trials for Fast analysis on the Mesa model, showing how different parameters affect the number of agents at equilibrium.

With respect to the infectious equilibrium measure, the four parameters shown

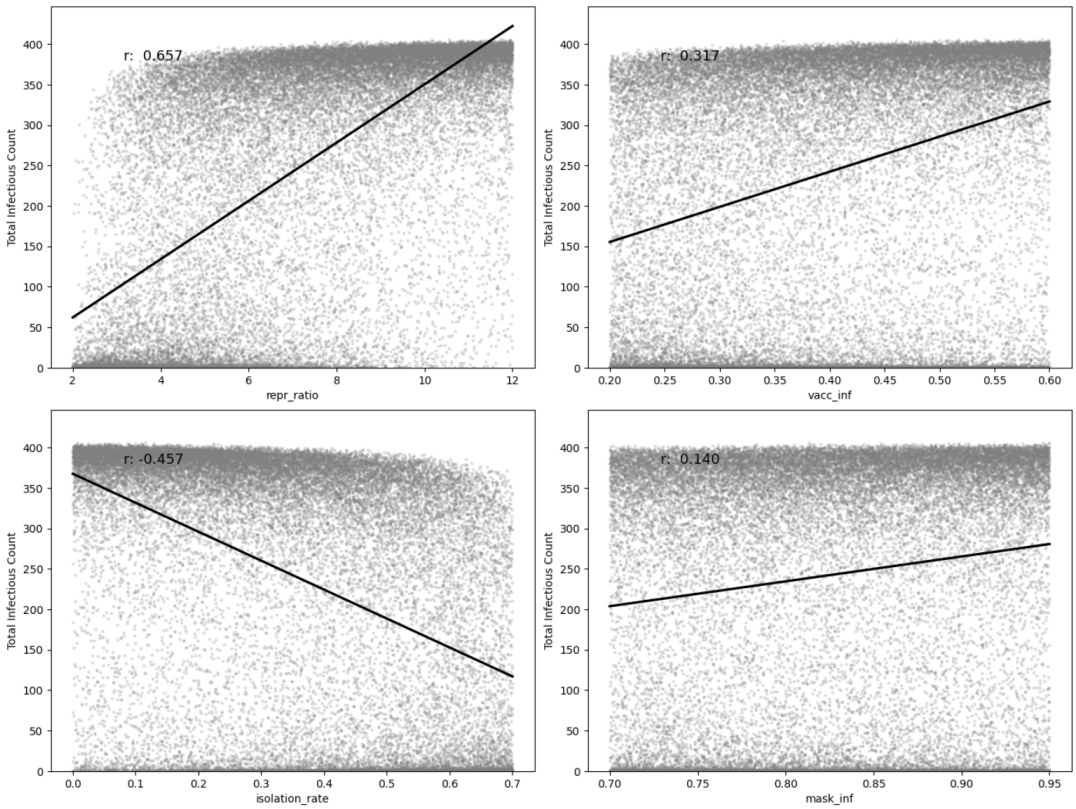

Figure 5.3: Scatter plots showing how parameter values affect infectious equilibrium values in the Mesa model after running 30,000 trials during Fast analysis. A line of best fit is superimposed, showing a general trend in parameter impact.

all have some level of impact on the output. But, for parameters that have a greater impact on output than others, there tends to be a tighter correlation between parameter value and model output. For instance, in the four scatter plots shown in Figure 5.3, there is a line of best fit with a calculated correlation coefficient. Though this line very loosely fits the data, with the data likely not being linear, to begin with, the magnitude of the correlation coefficients for each parameter, when sorted, results in the same order of parameter importance calculated in the more sophisticated sensitivity analysis. Still, these should not be taken as definitive results - they should only

be used as a preliminary investigation from modelers. A very pronounced example of this can be seen in Figure 5.4, where we see how the "mask chance" parameter influences the number of masked agents. This is an obvious result but is an example of one extreme, where one parameter dwarfs the others in terms of importance.

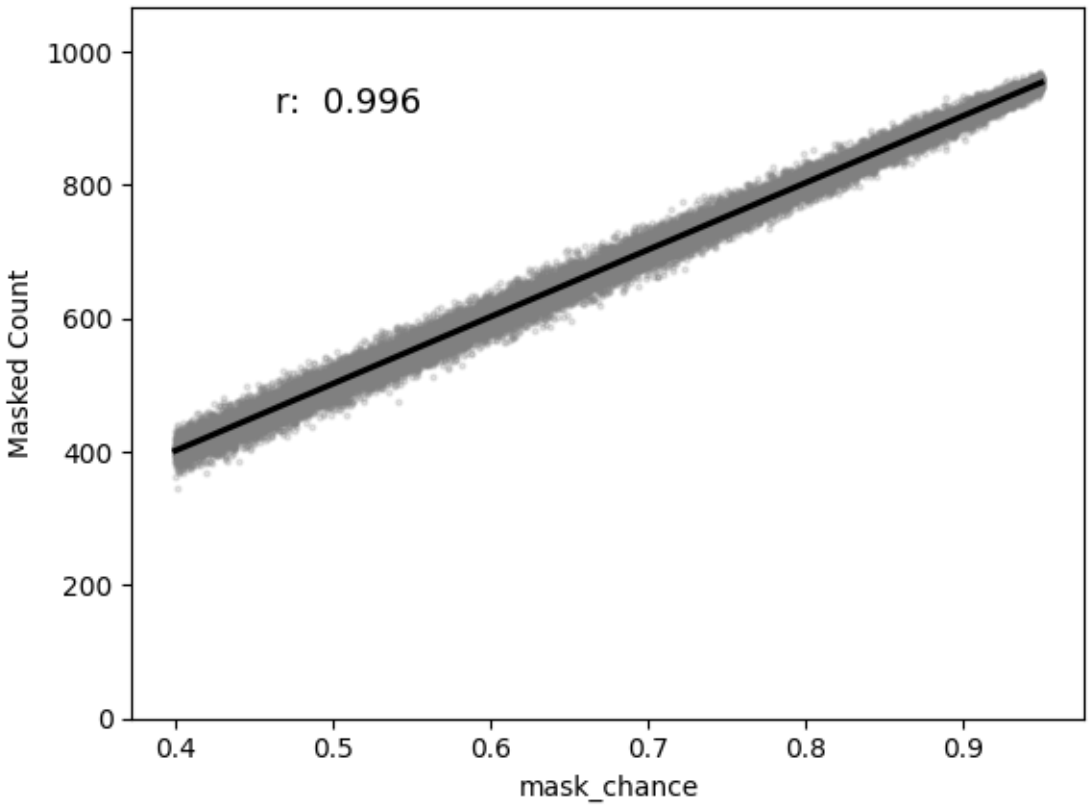

Figure 5.4: Scatter plots showing how the mask chance parameter value affects infectious equilibrium values in the Mesa model after running 30,000 trials during Fast analysis. A line of best fit is superimposed, showing a general trend in parameter impact.

We can also look at the distribution of measures of interest to see if further steps should be taken. In the same 30,000 runs, the SEIR counts at equilibrium were recorded and are shown in Figure 5.5. Similarly, the peak values observed at each class are shown in Figure 5.6 and the time to reach peak values is shown in Figure

5.7

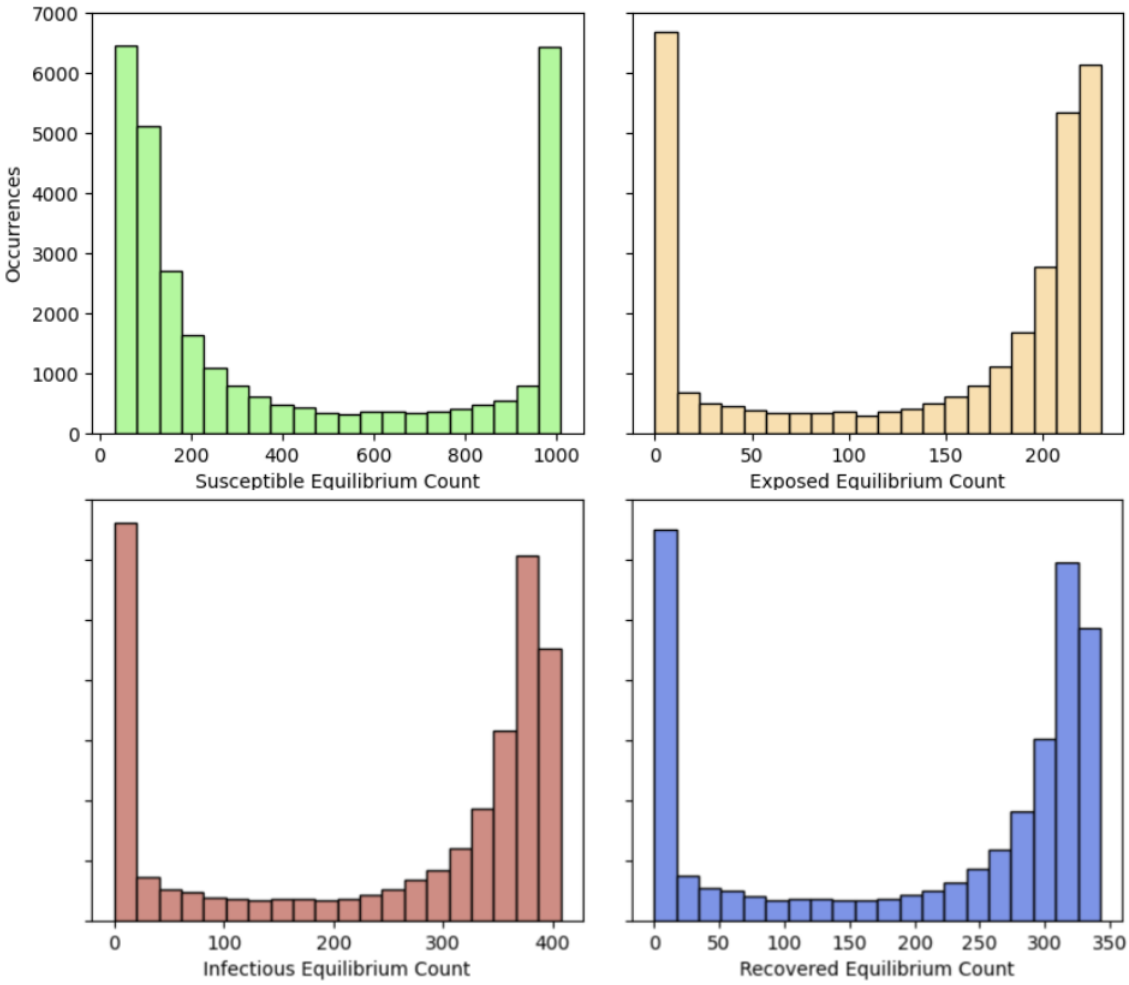

Figure 5.5: Bar graphs showing the counts at equilibrium for the four SEIR classes over 30,000 trials with varying parameter values.

It is important that the measures being examined have sufficient value ranges when running sensitivity analysis so that the analysis can produce results that are as meaningful as possible. Setting parameter value ranges about some estimated "ground truth" value such that these measures show interesting results is essential in the task of sensitivity analysis. In some cases, the distribution of these measure

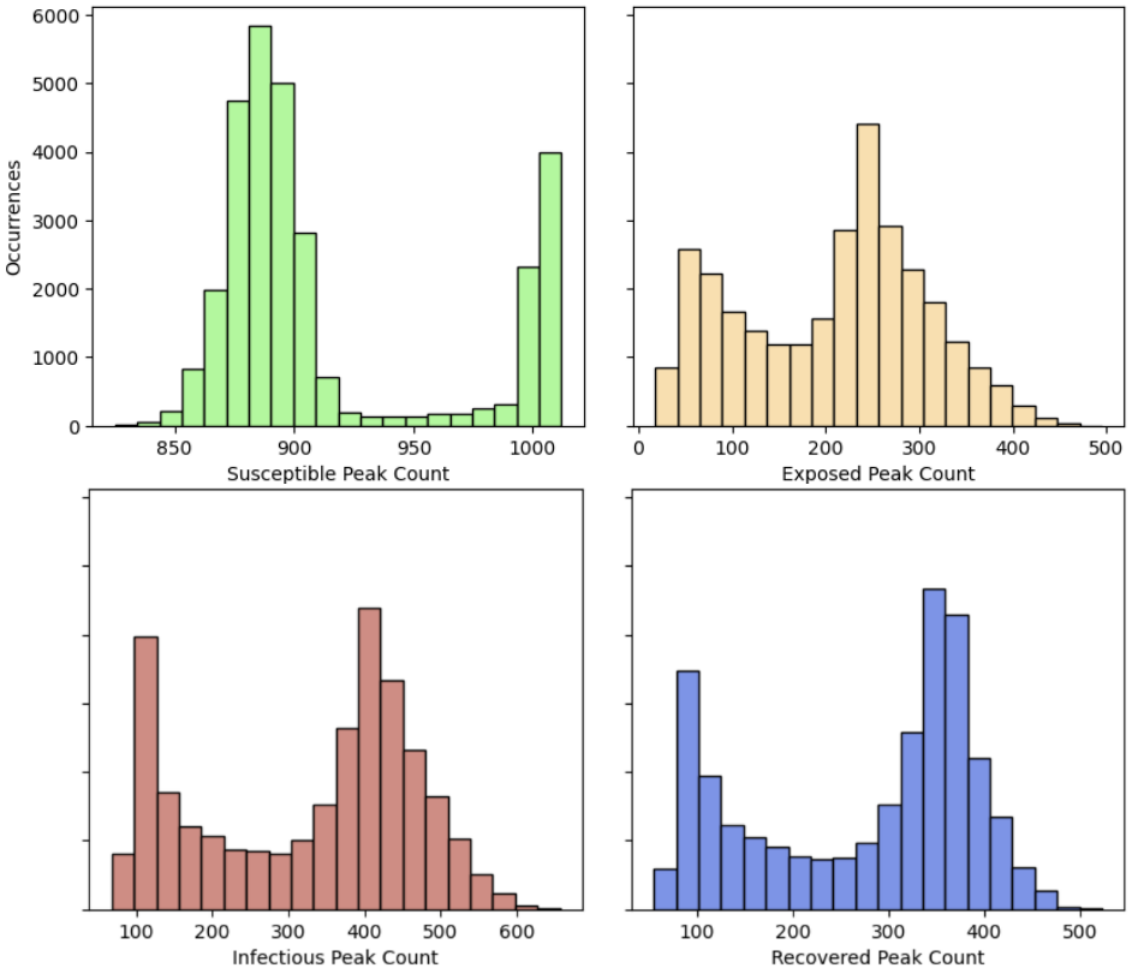

Figure 5.6: Bar graphs showing the peak counts for the four SEIR classes over 30,000 trials with varying parameter values.

values may affect how the sensitivity analysis should be done, but that is not explored in this work.

It is important to note that sensitivity analysis is not perfect, and different methods can produce slightly different results. In our case, the order of importance of parameters with respect to each measure is consistent over nearly all methods. However, in cases where there are two or more parameters of almost equal importance relative to a measure, it is possible that different methods may rank the parameters

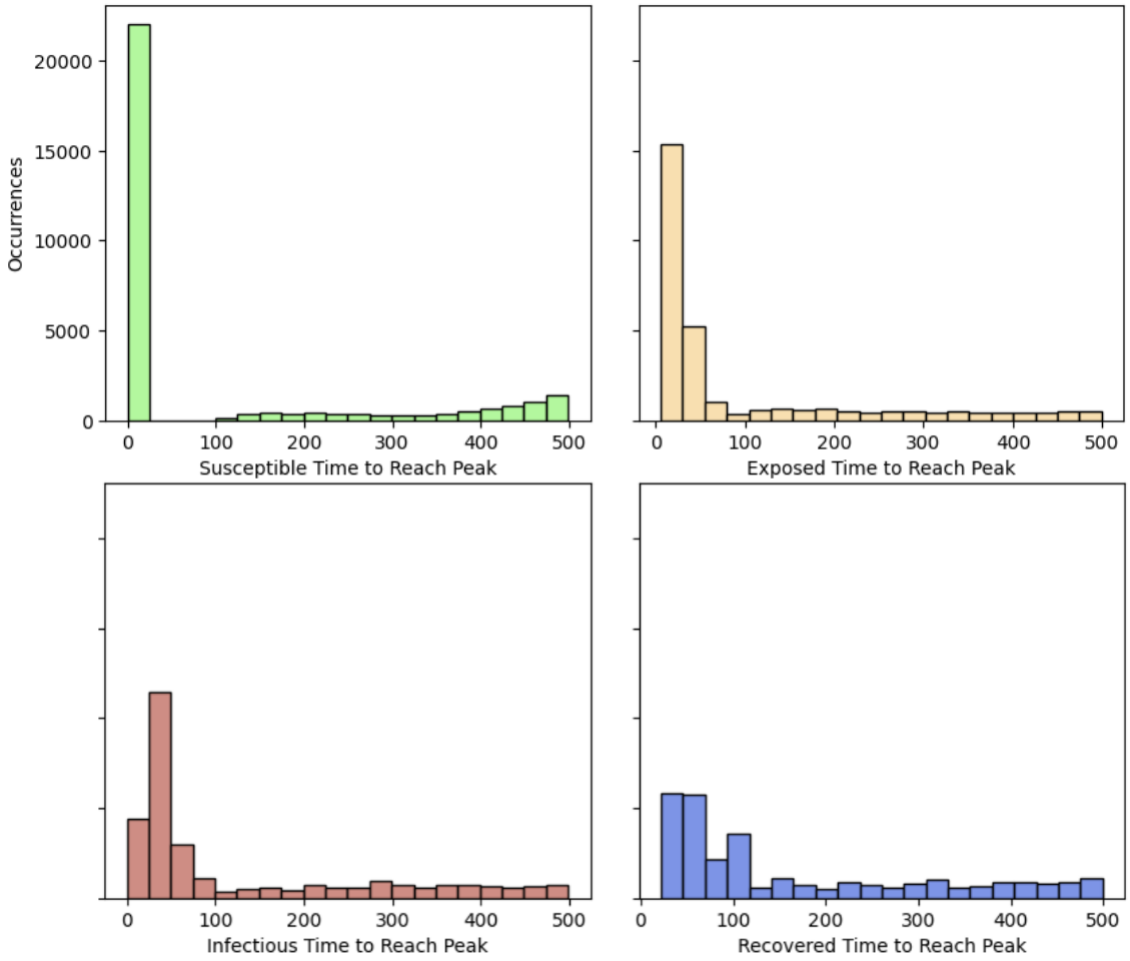

Figure 5.7: Bar graphs showing the time to reach peak counts for the four SEIR classes over 30,000 trials with varying parameter values.

differently in importance. As such, the values reported here should be taken with this in mind, noting that there is a margin of error arising from parameter value range initialization, sampling method used, model construction, and method chosen. Typically, model analysis ends here, and the relative importances are used at the modeler's discretion. But, while these values are generally not taken further, this work takes the sensitivity analysis *values* calculated and uses them in downstream analysis with the aim of identifying problematic parameters.

5.2 Parameter Analysis

With this method of parameter analysis via the comparison of ground truth measures with a model's output measures, modelers can begin to narrow their search into smaller subsets of parameters that may be initialized with incorrect values in the case where compared measures differ. This acts as a formal, mathematically-backed argument that modelers can use to justify further research into obtaining more accurate values with respect to particular parameters against the alternative of gathering data with a less specific model-oriented goal.

For the sake of example in this work, I use the Mesa model to generate outputs that I derive measure values from under different parameter settings and compare it to the "ground truth" measure values that are said to be the measure values obtained having run the Mesa model with a static parameter configuration over a number of runs. For the sensitivity results that are required in this algorithm, I use the "Total Sensitivity" results calculated via the Sobol' method as shown in Table 5.1. More specifically, the ground truth measure values are taken from running the Mesa model 10,000 times using the same parameter configurations as when the "base models" were generated, where the isolation rate was 0.3, the basic reproductive ratio was 6, the chance of masking was 0.7, the infection factor from masking was 0.8, and the infection factor from vaccinating was 0.4. To generate "perturbed" results, five different parameter configurations were used to explore this new method of parameter analysis, with the model being run 1000 times for each configuration. The results for all parameter configurations can be seen in Table 5.2

The first two parameter configurations that were examined were when the "mask

Parameter	Mask Chance: 0.1	Mask Chance: 0.5	Isolation Rate: 0.4	Basic Reproduction Number: 6	All Changed
Mask Chance	1.0	0.23	0.09	0.08	0.01
Basic Reproduction Number	0.74	1.0	1.0	1.0	1.0
Isolation Rate	0.38	0.52	0.57	0.57	0.53
Vaccine Infection Factor	0.18	0.24	0.32	0.32	0.25
Mask Infection Factor	0.0	0.0	0.0	0.0	0.0

Table 5.2: Parameter likelihood results after running measure comparison algorithm across five distinct parameter configurations.

chance" parameter was changed to 0.1 and to 0.5, leaving all other parameters unchanged. This was to see how this method of parameter analysis was performed under circumstances where *only one* parameter was changed, where a measure exists that is *directly* affected by it, and where *no other* parameters affect it. In the case where the "mask chance" parameter was set to 0.1, we see that the "mask chance" parameter is correctly identified to be most likely to be perturbed. Its value of 0.1 produced measure values that were far enough away from the ground truth measure values that, despite other measures seeing significant change (which is why the other parameters are not all 0 in the ordered list), it was still listed at the top. This is an example of the algorithm working exactly as one might expect - one parameter only is perturbed, and the algorithm correctly identifies it.

In the case where the "mask chance" parameter was set to 0.5, the algorithm does *not* perform as one might initially expect, meaning that it does *not* get marked as

being one of the most likely parameters to have been perturbed per the algorithm. This is due to the fact that changing that "mask chance" parameter by this amount is enough to change other measures enough that the parameters fueling the other measures were calculated to be more likely to be the ones that were perturbed. In particular, all of the measures were changed significantly (i.e. where the difference in measure values is greater than one standard deviation) except for the "vaccinated count" and "time to reach infectious peak" measures. Also, even though the only parameter to have been changed here was the chance of masking, the "masked count" measure was only the *third* most perturbed measure in terms of standard deviation, behind the infectious and susceptible equilibrium measures. Thus, since the "mask chance" parameter has a relatively low impact on those measures, the other parameters were boosted up more significantly in terms of the ordered list. It is also important to note that even despite this, the "mask chance" parameter was calculated to be more likely to be perturbed than the "mask infection factor" parameter and was placed almost identically to the "vaccine infection factor" parameter, even though those parameters influence the equilibrium measures more significantly than the "mask chance" parameter (as seen in the example Table 5.1). This indicates that even under circumstances with multiple measures being impacted by a parameter change, having a specific measure still gives the algorithm the potential to pick up that particular parameter change.

Now, let us examine the other three cases. We have three distinct cases where: (1) I set the vaccination infection factor to 0.8, keeping all other parameters unchanged, (2) I set the basic reproductive ratio to 10, keeping all other parameters unchanged, and (3) I change all parameters so that the isolation rate is changed to 0.4, the basic

reproductive ratio is set to 4, the vaccine infection factor is set to 0.6, the chance of masking is set to 0.5, and the mask infection factor is set to 0.7. These results are much less clear and do not correctly identify the parameters that were perturbed. However, there is still information that can be drawn from this. The main point that these results indicate is that even after applying this algorithm to the selected measures, the order and likelihood of parameter perturbations that the algorithm calculates is relatively consistent, no matter the parameter configuration. This sort of result may indicate to modelers that for the most important measures (decided by them), parameters such as the basic reproduction number are *far more important* than parameters such as the chance of an agent masking or the mask infection factor in their respective ranges per the sensitivity analysis. Even though these results are lacking in some regards, this information is highly useful when deciding which parameters to investigate if the aim is to improve the quality of the model with respect to certain measures. It may be that it is simply not worth the effort to put extra time and funding into obtaining highly accurate certain parameter values when there are others that have a dramatically higher influence on output.

A general concept that these results also suggest is that identifying and gathering *specific* measures that are sufficiently sensitive under parameter change is highly desirable if the end goal is parameter analysis of this kind (where "specific" refers to a measure that is impacted by as few parameters as possible). With more general measures that are influenced by all parameters, the results will often be less informative than in the converse case, as we saw in the case of the "mask chance" parameter-measure combination. The problem with this is that it is limited by the availability of such specific measures. It is often the case that more general measures

are easier to observe in the real world due to their overall perceived "importance" - the infectious agents at equilibrium are arguably more "important" than the number of masked agents in terms of immediate pandemic response, and so it may be more difficult to gather such specific measure data. This problem then presents a need to gather data related to more specific measures, even if they are not necessarily deemed the most "important" to policymakers - the measures may end up being able to significantly improve model performance, which in turn can cause forecasting on the more "important" measures to be much more accurate.

Chapter 6

Conclusions

6.1 Summary

Having a deep understanding of how the parameters affect models of all kinds is a problem that is becoming increasingly relevant with a higher demand for accurate and complex models. While there are plenty of existing tools that facilitate the usage of sensitivity analysis, their implementation is limited and concrete steps that can be taken after this have scarcely been researched or documented. This work provides an end-to-end example of how agent-based disease models can be created using a variety of modeling tools and demonstrates the use of various sensitivity analysis methods on a number of measures and parameters. Then, as an extension to these existing concepts, I presented a method of parameter analysis that aims to identify which parameters must be modified in order for a model to produce measure values that match some "ground truth" observed values. This algorithm was found to be adequate in solving the task at hand, highlighting sensitive parameters based on how they affect certain output measures. Following these results, the recommendation to disease modelers is to carefully consider the impact their parameters (that may

be incorrectly initialized) have on key measures of the model outcomes, and doing this through a thorough examination of sensitivity analysis methods. Regardless of the type of model used, modelers should perform detailed sensitivity analysis (ideally with more than one method) with well-thought-out measures so that the downstream analysis presented via the parameter analysis algorithm can be taken full advantage of.

6.2 Limitations

With the work of large-scale disease modeling and sensitivity analysis where simulations on the scale of tens or even hundreds of thousands are required, one obvious limitation is in the computational power available to the modeler. While the Mesa and Repast models were quick in their execution, the RDDL and NetLogo models were quite the opposite. It is also the case that increasing the complexity of the models in general would further increase simulation time. Given that these models are not as complex as many models that exist in the literature, the possible computation time is highly likely to increase if a more realistic model were to be created. This is a common problem in the world of disease modeling. Typically, results need to be acquired as *quickly as possible* since the aim of public health officials is to reduce the spread of disease as soon as they can.

Another main source that could have contributed to more fruitful results is the measure selection for sensitivity analysis. The entire sensitivity analysis process takes a substantial amount of time, and a hypothesis was made that the measures chosen may be impacted by parameters in different ways, having some parameters heavily influence one while others influenced other measures. This sort of behaviour was

guaranteed in the "masked agent count" measure with the "mask chance" parameter, but it did not occur at all with the other measures - they essentially all had the same sensitivity results, with parameters affecting all of the measure values in an almost identical way. If more specific measures were created from slightly more complex behaviours and data logging, it is possible that the sensitivity analysis results would have been more diverse, lending to more interesting and illustrative results when applying the custom parameter analysis algorithm due to its preliminary nature.

The issue of data availability is one that plagues modelers in essentially every disease modeling task, and it is one that also affected the models in this work. In an ideal world, a model perfectly representing Kingston (to some level of abstraction) could be created. This would imply that statistics related to housing, age, and occupation (to name a few) would have to be entirely accurate. However, this is not the case, and a level of assumption is always made about them. This, coupled with the geography of Kingston not being perfectly captured by the OSMnx Python package (which was used to ingest building into the model) leads the models' representations of Kingston to be slightly inaccurate. While the main idea of this work would be unaffected by this, it is the case that the models may have had their measure values differ from an ideal model where all of the statistics were perfectly represented (i.e. the real world). This discrepancy between models and the real world is something that modelers should always be cognisant of.

6.3 Future Work

The future work in this line of research can be broken down into two categories: future work for researchers aiming to improve upon the ideas presented here and future work

for modelers aiming to implement this work.

In terms of future lines of research, the obvious choice would be to work on improving the parameter analysis algorithm. Since this is quite mathematical in nature, extensive collaboration between modelers, programmers, and mathematicians or statisticians would be essential so that the nature of the problem could be properly conveyed, a high-quality solution could be produced, and it could then be translated into code. There is no concrete path laid out for this, and so it is possible that an entirely different approach is taken - either way, the problem of identifying incorrectly initialized parameters is one that can be readily improved upon. Another non-mathematical approach to this problem would be to incorporate machine learning in some form. By creating a machine learning model that learns how parameter adjustment affects measure values, it is possible that information could be gained via model explainability. One could then set up such a construction that allows a model to identify which parameters must be "tweaked" in order to elicit ground truth measure values - effectively solving the problem but with a different approach. Having other metrics in addition to sensitivity analysis could also be beneficial if included in some parameter analysis algorithms. The concept of "robustness" is one that has seen discussion in the sphere of sensitivity analysis and could also be a metric that modelers and public health officials may deem important [69].

From a modeler's perspective, future work would be in the form of setting themselves up for the best possible model with this algorithm in mind. This would mean creating accurate models with the structure to support high-quality and informative measures so that parameter analysis can be as useful as possible. As a result, comparing a highly realistic model of this nature to actual ground truth values observed

in the real world would be an interesting test of how the algorithm performs.

Bibliography

[1] A. Adiga, D. Dubhashi, B. Lewis, M. Marathe, S. Venkatramanan, and A. Vullikanti. Models for COVID-19 pandemic: A comparative analysis. *Journal of the Indian Institute of Science*, 100(4):793–807, 2020. doi: 10.1007/s41745-020-00200-6. URL https://doi.org/10.1007/s41745-020-00200-6.

[2] Y. Alimohamadi, M. Taghdir, and M. Sepandi. Estimate of the basic reproduction number for COVID-19: A systematic review and meta-analysis. *Journal of Preventive Medicine and Public Health*, 53(3):151–157, 2020. doi: 10.3961/jpmph.20.0765. URL https://doi.org/10.3961/jpmph.20.076.

[3] D. Balcan, V. Colizza, B. Gonçalves, H. Hu, J. Ramasco, and A. Vespignani. Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the national academy of sciences*, 106(51):21484–21489, 2009. doi: 10.1073/pnas.0906910106. URL https://doi.org/10.1073/pnas.0906910106.

[4] J. Basseal, C. Bennett, P. Collingon, B. Currie, D. Durrheim, J. Leask, E. McBryde, P. McIntyre, F. Russell, D. Smith, T. Sorrell, and B. Marais. Key lessons from the COVID-19 public health response in Australia. *The Lancet Regional Health–Western Pacific*, 30:100616, 2023. doi: 10.1016/j.lanwpc.2022.100616. URL https://doi.org/10.1016/j.lanwpc.2022.100616.

[5] J. Bennett. *OpenStreetMap*. Packt Publishing Ltd, 2010.

[6] G. Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. doi: 10.1016/j.compenvurbsys.2017.05.004. URL `https://doi.org/10.1016/j.compenvurbsys.2017.05.004`.

[7] L. Bollen and W. R. van Joolingen. Simsketch: Multiagent simulations based on learner-created sketches for early science education. *IEEE Transactions on Learning Technologies*, 6(3):208–216, 2013. doi: 10.1109/TLT.2013.9. URL `https://doi.org/10.1109/TLT.2013.9`.

[8] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007. doi: 10.1016/j.ress.2006.04.015. URL `https://doi.org/10.1016/j.ress.2006.04.015`.

[9] T. Britton. Basic prediction methodology for COVID-19: Estimation and sensitivity considerations. *MedRxiv*, 2020. doi: 10.1101/2020.03.27.20045575. URL `https://doi.org/10.1101/2020.03.27.20045575`.

[10] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2010. doi: 10.1002/SPE.995. URL `https://doi.org/10.1002/spe.995`.

[11] F. Campolongo, J. Cariboni, and A. Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental modelling & software*, 22(10):

1509–1518, 2007. doi: 10.1016/j.envsoft.2006.10.004. URL https://doi.org/10.1016/j.envsoft.2006.10.004.

[12] S. Canada. Census profile, 2021 census of population - profile table, 2021. URL https://www12.statcan.gc.ca/census-recensement/2021/dp-pd/prof/details/page.cfm?Lang=E&GENDERlist=1%2C2%2C3&STATISTIClist=1%2C4&HEADERlist=0&DGUIDlist=2021A00053510010&SearchText=Kingston.

[13] L. Cao, Q. Liu, and W. Hou. COVID-19 modeling: A review. *CoRR*, abs/2104.12556, 2021. doi: 10.48550/arXiv.2104.12556. URL https:doi.org/10.48550/arXiv.2104.12556.

[14] K. Carley, D. Fridsma, E. Casman, A. Yahja, N. Altman, L. Chen, B. Kaminsky, and D. Nave. BioWar: Scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, 36(2):252–265, 2006. doi: 10.1109/TSMCA.2005.851291. URL https://doi.org/10.1109/TSMCA.2005.851291.

[15] B. Chidley and C. Muise. Using probabilistic planning to model the spread of covid-19 in kingston, ontario. *34th International Conference on Automated Planning and Scheduling*, 2024. URL https://icaps24.icaps-conference.org/program/workshops/rddps-papers/Chidley-RDDPS24.pdf.

[16] G. Chowell, M. Miller, and C. Viboud. Seasonal influenza in the United States, France, and Australia: Transmission and prospects for control. *Epidemiology & Infection*, 136(12):852–864, 2008. doi: 10.1017/S0950268807009144. URL https://doi.org/10.1017/S0950268807009144.

[17] M. Ciotti, M. Ciccozzi, A. Terrinoni, W.-C. Jiang, C.-B. Wang, and S. Bernardini. The COVID-19 pandemic. *Critical reviews in clinical laboratory sciences*, 57(6):365–388, 2020. doi: 10.1080/10408363.2020.1783198. URL `https://doi.org/10.1080/10408363.2020.1783198`.

[18] K. Claessen and J. Hughes. Quickcheck: a lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000*, pages 268–279. ACM, 2000. doi: 10.1145/351240.351266. URL `https://doi.org/10.1145/351240.351266`.

[19] N. Collier. Repast: An extensible framework for agent simulation. *Natural Resources and Environmental Issues*, 8(4), 2001.

[20] S. de Marchi and S. Page. Agent-based models. *Annual Review of Pilicial Science*, 17:1–20, 2014. doi: 10.1146/annurev-polisci-080812-191558. URL `https://doi.org/10.1146/annurev-polisci-080812-191558`.

[21] P. Delamater, E. Street, T. Leslie, T. Yang, and K. Jacobson. Complexity of the basic reproduction number (R0). *Emerging infectious diseases*, 25(1):1–4, 2019. doi: 10.3201/eid2501.171901. URL `https://doi.org/10.3201/eid2501.171901`.

[22] K. Dietz. The estimation of the basic reproduction number for infectious diseases. *Statistical methods in medical research*, 2(1):23–41, 1993. doi: 10.1177/096228029300200103. URL `https://doi.org/10.1177/096228029300200103`.

[23] M. Doherty, P. Buchy, B. Standaert, C. Giaquinto, and D. Prado-Cohrs. Vaccine impact: Benefits for human health. *Vaccine*, 34(52):6707–6714, 2016. doi: 10.1016/j.vaccine.2016.10.025. URL https://doi.org/10.1016/j.vaccine.2016.10.025.

[24] N. Ferguson, D. Lardon, G. Nedjati-Gilani, N. Imai, K. Ainslie, M. Baguelin, S. Bhatia, A. Boonyasiri, Z. Cucunubá, G. Cuomo-Dannenburg, A. Dighe, I. Dorigatti, H. Fu, K. Gaythorpe, W. Green, A. Hamlet, W. Hinsley, L. Okell, S. van Elsland, H. Thompson, R. Verity, E. Volz, H. Wang, Y. Wang, P. Walker, C. Walters, P. Winskill, C. Whittaker, C. Donnely, S. Riley, and A. Ghani. Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand. *Imperial College COVID-19 Response Team*, 20(4), 2020. doi: 10.25561/77482. URL https://doi.org/10.25561/77482.

[25] Y. Gan, Q. Duan, W. Gong, C. H. Tong, Y. Sun, W. Chu, A. Ye, C. Miao, and Z. Di. A comprehensive evaluation of various sensitivity analysis methods: A case study with a hydrological model. *Environmental Modelling & Software*, 51:269–285, 2014. doi: 10.1016/J.ENVSOFT.2013.09.031. URL https://doi.org/10.1016/j.envsoft.2013.09.031.

[26] M. Gimelfarb, A. Taitler, and S. Sanner. Jaxplan and gurobiplan: Optimization baselines for replanning in discrete and mixed discrete and continuous probabilistic domains. In *34th International Conference on Automated Planning and Scheduling*, 2024. URL https://openreview.net/forum?id=7IKtmUpLEH.

[27] P. Godfrey-Smith. Models and fictions in science. *Philosophical Studies*, 143:

101–116, 2009. doi: 10.1007/s11098-008-9313-2. URL https://doi.org/10.1007/s11098-008-9313-2.

[28] T. Gorochowski, A. Matyjaszkiewicz, T. Todd, N. Oak, K. Kowalska, S. Reid, K. Tsaneva-Atanasova, N. Savery, C. Grierson, and M. di Bernardo. BSim: An agent-based tool for modeling bacterial populations in systems and synthetic biology. *PloS one*, 7(8):e42790, 2012. doi: 10.1371/journal.pone.0042790. URL https://doi.org/10.1371/journal.pone.0042790.

[29] F. Guerra, S. Bolotin, G. Lim, J. Heffernan, S. Deeks, Y. Li, and N. Crowcroft. The basic reproduction number (r0) of Measles: A systematic review. *The Lancet Infectious Diseases*, 17(12):e420–e428, 2017. doi: 10.1016/S1473-3099(17)30307-9. URL https://doi.org/10.1016/S1473-3099(17)30307-9.

[30] A. Haghnegahdar, S. Razavi, F. Yassin, and H. Wheater. Multicriteria sensitivity analysis as a diagnostic tool for understanding model behaviour and characterizing model uncertainty. *Hydrological Processes*, 31(25):4462–4476, 2017. doi: 10.1002/hyp.11358. URL https://doi.org/10.1002/hyp.11358.

[31] M. Harmanani. Modelling the spread of COVID-19 in indoor spaces using Automated Probabilistic Planning. *ICAPS Scheduling and Planning Applications woRKshop*, 2023. doi: 10.48550/arXiv.2308.08190. URL https://doi.org/10.48550/arXiv.2308.08190.

[32] J. Herby, L. Jonung, and S. Hanke. A literature review and meta-analysis of the effects of lockdowns on COVID-19 mortality-II. *medRxiv*, 2023. doi: 10.1101/2023.08.30.23294845. URL https://doi.org/10.1101/2023.08.30.23294845.

[33] J. Herman and W. Usher. SALib: An open-source python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), 2017. doi: 10.21105/joss.00097. URL https://doi.org/10.21105/joss.00097.

[34] R. Hinch, W. Probert, A. Nurtay, M. Kendall, C. Wymant, M. Hall, K. Lythgoe, A. B. Cruz, L. Zhao, A. Stewart, L. Ferretti, D. Montero, J. Warren, N. Mather, M. Abueg, N. Wu, O. Legat, K. Bentley, T. Mead, K. Van-Vuuren, D. Feldner-Busztin, T. Ristori, A. Finkelstein, D. Bonsall, L. Abeler-Dörner, and C. Fraser. OpenABM-Covid19 - An agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. *PLoS Computational Biology*, 17(7), 2021. doi: 10.1371/JOURNAL.PCBI.1009146. URL https://doi.org/10.1371/journal.pcbi.1009146.

[35] E. Hunter, B. Namee, and J. Kelleher. An open-data-driven agent-based model to simulate infectious disease outbreaks. *PloS one*, 13(12):e0208775, 2018. doi: 10.1371/journal.pone.0208775. URL https://doi.org/10.1371/journal.pone.0208775.

[36] B. Iooss and P. Lemaître. A review on global sensitivity analysis methods. *Uncertainty management in simulation-optimization of complex systems: algorithms and applications*, 59:101–122, 2024. doi: 10.1007/978-1-4899-7547-8_5. URL https://doi.org/10.1007/978-1-4899-7547-8_5.

[37] T. Iwanaga, W. Usher, and J. Herman. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. *Socio-Environmental Systems Modelling*, 4:18155, 2022. doi: 10.18174/sesmo.18155. URL https://doi.org/10.18174/sesmo.18155.

[38] S. Kamerlin and P. Kasson. Managing coronavirus disease 2019 spread with voluntary public health measures: Sweden as a case study for pandemic control. *Clinical Infectious Diseases*, 71(12):3174–3181, 2020. doi: 10.1093/cid/ciaa864. URL `https://doi.org/10.1093/cid/ciaa864`.

[39] E. Kaplan, D. Wang, M. Wang, A. Malik, A. Zulli, and J. Peccia. Aligning SARS-CoV-2 indicators via an epidemic model: application to hospital admissions and RNA detection in sewage sludge. *Health Care Management Science*, 24:320–329, 2021. doi: 10.1007/s10729-020-09525-1. URL `https://doi.org/10.1007/s10729-020-09525-1`.

[40] K. Karamanos, A. Gkiolmas, A. Chalkidis, C. Skordoulis, M. Papaconstantinou, and D. Stavrou. Ecosystem food-webs as dynamic systems: Educating undergraduate teachers in conceptualizing aspects of food-webs' systemic nature and comportment. *Advances in Systems Science and Applications*, 12(4), 2012.

[41] C. Kerr, R. Stuart, D. Mistry, R. Abeysuriya, K. Rosenfeld, G. Hart, R. Núñez, J. Cohen, P. Selvaraj, B. Hagedorn, L. George, M. Jastrzebski, A. Izzo, G. Fowler, A. Palmér, D. Delport, N. Scott, S. Kelly, C. Bennette, B. Wagner, S. Chang, A. Oron, E. Wenger, J. Panovska-Griffiths, M. Famulare, and D. Klein. Covasim: An agent-based model of COVID-19 dynamics and interventions. *PLoS Computational Biology*, 17(7), 2021. doi: 10.1371/JOURNAL.PCBI.1009149. URL `https://doi.org/10.1371/journal.pcbi.1009149`.

[42] J. Kleijnen. Sensitivity analysis and optimization of system dynamics models: Regression analysis and statistical design of experiments. *System Dynamics Review*, 11(4):275–288, 1995. doi: 10.1002/sdr.4260110403. URL `https:`

//doi.org/10.1002/sdr.4260110403.

[43] J. Klein. BREVE: a 3D environment for the simulation of decentralized systems and artificial life. In *Proc. of the Int. Conf. on Artificial Life*, pages 329–334, 2002.

[44] M. Komosinski and S. Ulatowski. Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In *Advances in Artificial Life, 5th European Conference, ECAL'99, Lausanne, Switzerland, September 13-17, 1999, Proceedings*, volume 1674 of *Lecture Notes in Computer Science*, pages 261–265. Springer, 1999. doi: 10.1007/3-540-48304-7_33. URL https://doi.org/10.1007/3-540-48304-7_33.

[45] R. Laubenbacher, A. Jarrah, H. Mortveit, and S. Ravi. A mathematical formalism for agent-based modeling. *Computational Complexity*, pages 88–104, 2012. doi: 10.1007/978-1-4614-1800-9_6. URL https://doi.org/10.1007/978-1-4614-1800-9_6.

[46] G. Li, H. Rabitz, P. Yelvington, Oluwayemisi, F. Bacon, C. Kolb, and J. Schoendorf. Global sensitivity analysis for systems with independent and/or correlated inputs. *The journal of physical chemistry A*, 114(19):6022–6032, 1987. doi: 10.1021/jp9096919. URL https://doi.org/10.1021/jp9096919.

[47] F. Lorig, N. Dammenhayn, D.-J. Müller, and I. Timm. Measuring and comparing scalability of agent-based simulation frameworks. In *Multiagent System Technologies*, pages 42–60. Springer, 2015. doi: 10.1007/978-3-319-27343-3_3. URL https://doi.org/10.1007/978-3-319-27343-3_3.

[48] Q.-B. Lu, Y. Zhang, M.-J. Liu, H.-Y. Zhang, N. Jalali, A.-R. Zhang, H. Zhao, Q.-Q. Song, T.-S. Zhao, J. Zhao, H.-Y. Liu, J. Du, A.-Y. Teng, Z.-W. Zhou, S.-X. Zhou, T.-L. Che, T. Wang, T. Yang, X.-G. Guan, X.-F. Peng, Y.-N. Wang, Y.-Y. Zhang, S.-M. Lv, B.-C. Liu, W.-Q. Shi, X.-A. Zhang, X.-G. Duan, W. Liu, Y. Yang, and L.-Q. Fang. Epidemiological parameters of COVID-19 and its implication for infectivity among patients in China, 1 January to 11 February 2020. *Eurosurveillance*, 25(40):2000250, 2020. doi: 10.2807/1560-7917.ES.2020.25.40.2000250. URL `https://doi.org/10.2807/1560-7917.ES.2020.25.40.2000250`.

[49] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. MASON: A multiagent simulation environment. *Simulation*, 81(7), 2005. doi: 10.1177/0037549705058073. URL `https://doi.org/10.1177/0037549705058073`.

[50] I. Mahmood, H. Arabnejad, D. Suleimenova, I. Sassoon, A. Marshan, A. E. Serrano-Rico, P. Louvieris, A. Anagnostou, S. J. E. Taylor, D. Bell, and D. Groen. FACS: A geospatial agent-based simulator for analysing COVID-19 spread and public health measures on local regions. *Journal of Simulation*, 16(4):355–373, 2022. doi: 10.1080/17477778.2020.1800422. URL `https://doi.org/10.1080/17477778.2020.1800422`.

[51] D. Masad and J. Kazil. Mesa: An agent-based modeling framework. In *Proceedings of the 14th Python in Science Conference*, pages 51–58. SciPy, 2015. doi: 10.25080/Majora-7b98e3ed-009. URL `https://doi.org/10.25080/Majora-7b98e3ed-009`.

[52] M. Mathur and T. VanderWeele. Sensitivity analysis for unmeasured confounding

in meta-analyses. *Journal of the American Statistical Association*, 115(529):163–172, 2020. doi: 10.1080/01621459.2018.1529598. URL https://doi.org/10.1080/01621459.2018.1529598.

[53] C. McPhail, H. Maier, J. Kwakkel, M. Giuliani, A. Castelletti, and S. Westra. Robustness metrics: How are they calculated, when should they be used and why do they give different results? *Earth's Future*, 6(2):169–191, 2018. doi: doi.org/10.1002/2017EF000649. URL https://doi.org/doi.org/10.1002/2017EF000649.

[54] M. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991. doi: 10.1080/00401706.1991.10484804. URL https://doi.org/10.1080/00401706.1991.10484804.

[55] n.a. Royal military college of canada, 2024. URL https://www.ouinfo.ca/universities/rmc.

[56] n.a. St. lawrence college, 2024. URL https://www.collegesinstitutes.ca/members/st-lawrence-college/.

[57] H. Nishiura, N. Linton, and A. Akhmetzhanov. Serial interval of novel coronavirus (COVID-19) infections. *International journal of infectious diseases*, 93:284–286, 2020. doi: 10.1016/j.ijid.2020.02.060. URL https://doi.org/10.1016/j.ijid.2020.02.060.

[58] P. H. A. of Canada. Covid-19 vaccination: Vaccination coverage, 2023. URL https://health-infobase.canada.ca/covid-19/vaccination-coverage/.

[59] W. H. Organization et al. Consensus document on the epidemiology of severe acute respiratory syndrome (sars). Technical report, World Health Organization, 2003.

[60] J. Parker and J. Epstein. A distributed platform for global-scale agent-based models of disease transmission. *ACM Transactions on Modeling and Computer Simulation*, 22(1):2:1–2:25, 2011. doi: 10.1145/2043635.2043637. URL `https://doi.org/10.1145/2043635.2043637`.

[61] W. Parker. Does matter really matter? Computer simulations, experiments, and materiality. *Synthese*, 169(3):483–496, 2009. doi: 10.1007/S11229-008-9434-3. URL `https://doi.org/10.1007/s11229-008-9434-3`.

[62] F. Pianosi and T. Wagener. A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environmental Modelling & Software*, 67:1–11, 2015. doi: 10.1016/j.envsoft.2015.01.004. URL `https://doi.org/10.1016/j.envsoft.2015.01.004`.

[63] F. Pianosi, K. Beven, J. Freer, J. Hall, J. Rougier, D. Stephenson, and T. Wagener. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016. doi: 10.1016/j.envsoft.2016.02.008. URL `https://doi.org/10.1016/j.envsoft.2016.02.008`.

[64] A. Potochnik. Optimality modeling and explanatory generality. *Philosophy of Science*, 74(5):680–691, 2007. doi: 10.1086/525613. URL `https://doi.org/10.1086/525613`.

[65] R. Pung, C. Chiew, B. Young, S. Chin, M. Chen, H. Clapham, A. Cook, S. Maurer-Stroh, M. Toh, C. Poh, M. Low, J. Lum, V. Koh, T. Mak, L. Cui, R. Lin, D. Heng, Y. Leo, D. Lye, and V. Lee. Investigation of three clusters of COVID-19 in Singapore: Implications for surveillance and response measures. *The Lancet*, 395(10229):1039–1046, 2020. doi: 10.1016/S0140-6736(20)30528-6. URL `https://doi.org/10.1016/S0140-6736(20)30528-6`.

[66] A. Puy, P. Roy, and A. Saltelli. Discrepancy measures for sensitivity analysis. *arXiv preprint arXiv:2206.13470*, 2022.

[67] I. Rahimi, F. Chen, and A. Gandomi. A review on COVID-19 forecasting models. *Neural Computing and Applications*, 35(33):23671–23681, 2023. doi: 10.1007/s00521-020-05626-8. URL `https://doi.org/10.1007/s00521-020-05626-8`.

[68] S. Razavi and H. Gupta. A multi-method generalized global sensitivity matrix approach to accounting for the dynamical nature of earth and environmental systems models. *Environmental Modelling & Software*, 114:1–11, 2019. doi: 10.1016/J.ENVSOFT.2018.12.002. URL `https://doi.org/10.1016/j.envsoft.2018.12.002b`.

[69] S. Razavi, A. J. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, E. Plischke, S. L. Piano, T. Iwanaga, W. E. Becker, S. Tarantola, J. H. A. Guillaume, J. D. Jakeman, H. V. Gupta, N. Melillo, G. Rabitti, V. Chabridon, Q. Duan, and H. R. Maier. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954, 2021. doi: 10.1016/J.ENVSOFT.2020.104954. URL `https://doi.org/10.1016/j.envsoft.2020.104954`.

[70] Registrar. 2023-24 enrolment report, 2024. URL https://www.queensu.ca/registrar/sites/uregwww/files/uploaded_files/2023-24%20Enrolment%20Report.pdf.

[71] M. Resnick. Starlogo: An environment for decentralized modeling and decentralized thinking. In *Conference Companion on Human Factors in Computing Systems*, pages 11–12, 1996.

[72] A. Saltelli, S. Tarantola, and K. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1): 39–56, 1999. doi: 10.1080/00401706.1999.10485594. URL https://doi.org/10.1080/00401706.1999.10485594.

[73] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: the Primer*. John Wiley & Sons, 2008.

[74] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer physics communications*, 181(2):259–270, 2010. doi: 10.1016/J.CPC.2009.09.018. URL https://doi.org/10.1016/j.cpc.2009.09.018.

[75] A. Saltelli, Ângela Pereira, J. V. der Sluijs, and S. Funtowicz. What do I make of your latinorum? Sensitivity auditing of mathematical modelling. *International Journal of Foresight and Innovation Policy*, 9(2-3-4):213–234, 2014. doi: 10.1504/IJFIP.2013.058610. URL https://doi.org/10.1504/IJFIP.2013.058610.

[76] A. Saltelli, K. A. Kalinina, W. Becker, P. Fennell, F. Ferretti, N. Holst, S. Li, and Q. Wu. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. *Environ. Model. Softw.*, 114:29–39, 2019. doi: 10.1016/J.ENVSOFT.2019.01.012. URL `https://doi.org/10.1016/j.envsoft.2019.01.012`.

[77] S. Sanner. Relational Dynamic Influence Diagram Language (RDDL): Language description. *Unpublished ms. Australian National University*, 2010.

[78] P. Silva, P. Batista, H. S. Lima, M. A. Alves, F. Guimarães, and R. Silva. COVID-ABS: an agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions. *Chaos, Solitons & Fractals*, 139:110088, 2020. doi: 10.1016/j.chaos.2020.110088. URL `https://doi.org/10.1016/j.chaos.2020.110088`.

[79] H. Sjödin, A. Johansson, Å. Brännström, Z. Farooq, H. Kriit, A. Wilder-Smith, C. Åström, J. Thunberg, M. Söderquist, and J. Rocklöv. COVID-19 healthcare demand and mortality in Sweden in response to non-pharmaceutical mitigation and suppression scenarios. *International journal of epidemiology*, 49(5):1443–1453, 2020. doi: 10.1093/ije/dyaa121. URL `https://doi.org/10.1093/ije/dyaa121`.

[80] I. Sobol and S. Kucherenko. Derivative based global sensitivity measures and their link with global sensitivity indices. *Mathematics and Computers in Simulation*, 79(10):3009–3017, 2009. doi: 10.1016/j.matcom.2009.01.023. URL `https://doi.org/10.1016/j.matcom.2009.01.023`.

[81] V. Srikrishnan and K. Keller. Small increases in agent-based model complexity

can result in large increases in required calibration data. *Environmental Modelling & Software*, 138:104978, 2021. doi: 10.1016/j.envsoft.2021.104978. URL https://doi.org/10.1016/j.envsoft.2021.104978.

[82] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987. doi: 10.1080/00401706.1987.10488205. URL https://doi.org/10.1080/00401706.1987.10488205.

[83] K. Sy, L. White, and B. Nichols. Population density and basic reproductive number of COVID-19 across United States counties. *PloS One*, 16(4):e0249271, 2021. doi: 10.1371/journal.pone.0249271. URL https://doi.org/10.1371/journal.pone.0249271.

[84] A. Taitler, M. Gimelfarb, S. Gopalakrishnan, M. Mladenov, X. Liu, and S. Sanner. pyrddlgym: From rddl to gym environments. *arXiv preprint arXiv:2211.05939*, 2022.

[85] S. Tarantola, N. Giglioli, J. Jesinghaus, and A. Saltelli. Can global sensitivity analysis steer the implementation of models for environmental assessments and decision-making? *Stochastic Environmental Research and Risk Assessment*, 16:63–76, 2002. doi: 10.1007/s00477-001-0085-x. URL https://doi.org/10.1007/s00477-001-0085-x.

[86] P. Teller. Twilight of the perfect model model. *Erkenntnis*, 55(3):393–415, 2001. URL https://www.jstor.org/stable/20013097.

[87] J. Tissot and C. Prieur. Bias correction for the estimation of sensitivity indices based on random balance designs. *Reliability Engineering & System Safety*, 107:

205–213, 2012. doi: 10.1016/j.ress.2012.06.010. URL https://doi.org/10.1016/j.ress.2012.06.010.

[88] S. Tisue and U. Wilensky. Netlogo: A simple environment for modeling complexity. In *International Conference on Complex Systems*, volume 21, pages 16–21. Citeseer, 2004.

[89] W. van der Toorn, D. Oh, D. Bourquain, J. Michel, E. Krause, A. Nitsche, and M. von Kleist. An intra-host SARS-CoV-2 dynamics model to assess testing and quarantine strategies for incoming travelers, contact management, and deisolation. *Patterns*, 2(6):100262, 2021. doi: 10.1016/J.PATTER.2021.100262. URL https://doi.org/10.1016/j.patter.2021.100262.

[90] S. Venkatramanan, J. Chen, A. Fadikar, S. Gupta, D. Higdon, B. Lewis, M. Marathe, H. Mortveit, and A. Vullikanti. Optimizing spatial allocation of seasonal influenza vaccine under temporal constraints. *PLoS Computational Biology*, 15(9), 2019. doi: 10.1371/JOURNAL.PCBI.1007111. URL https://doi.org/10.1371/journal.pcbi.1007111.

[91] P. Waddell. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American planning associatio*, 68(3):297–314, 2002. doi: 10.1080/01944360208976274. URL https://doi.org/10.1080/01944360208976274.

[92] R. Wölfel, V. Corman, W. Guggemos, M. Seilmaier, S. Zange, M. Müller, D. Niemeyer, T. Jones, P. Vollmar, C. Rothe, M. Hoelscher, T. Bleicker,

S. Brünink, J. Schneider, R. Ehmann, K. Zwirglmaier, C. Drosten, and C. Wendtner. Virological assessment of hospitalized patients with COVID-19. *Nature*, 581(7809):465–469, 2020. doi: 10.1038/s41586-020-2984-3. URL `https://doi.org/10.1038/s41586-020-2984-3`.

[93] Z. Wong, C. Bui, A. Chughtai, and C. Macintyre. A systematic review of early modelling studies of Ebola virus disease in West Africa. *Epidemiology & Infection*, 45(6):1069–1094, 2017. doi: 10.1017/S0950268817000164. URL `https://doi.org/10.1017/S0950268817000164`.

[94] Y. Xiang, Y. Jia, L. Chen, L. Guo, B. Shu, and E. Long. COVID-19 epidemic prediction and the impact of public health interventions: A review of COVID-19 epidemic models. *Infectious Disease Modelling*, 6:324–342, 2021. doi: 10.1016/j.idm.2021.01.001. URL `https://doi.org/10.1016/j.idm.2021.01.001`.

Appendix A

Sensitivity Results

The below graphs show the results of sensitivity analysis across the Mesa, Repast, and RDDL models. All methods, except for the Regional method, are shown for each model. The Regional method was not included due to its nature. Rather than outputting 1 singular value for each parameter/measure combination, it outputs values equal to the number of bins specified - in this case, there were 20 bins.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	446.1	654.1	312.7	82.6	140.6
Exposed Equilibrium	105.7	154.5	73.7	19.8	33.3
Infectious Equilibrium	185.0	271.5	129.9	34.4	58.3
Recovered Equilibrium	155.2	228.0	109.0	29.0	49.0
Peak Infectious Count	154.8	268.9	141.8	44.4	66.3
Time to Reach Peak Infectious Count	231.6	263.7	205.6	128.5	148.5
Masked Count	23.6	23.0	23.1	418.5	23.4
Vaccinated Count	20.8	20.7	20.9	20.6	20.8

Table A.1: Morris results (Mu Star) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.211	0.278	0.136	0.034	0.056
Exposed Equilibrium	0.211	0.277	0.135	0.034	0.055
Infectious Equilibrium	0.210	0.278	0.135	0.034	0.056
Recovered Equilibrium	0.211	0.278	0.136	0.034	0.056
Peak Infectious Count	0.195	0.304	0.145	0.036	0.059
Time to Reach Peak Infectious Count	0.178	0.232	0.090	0.025	0.034
Masked Count	0.007	0.009	0.007	0.592	0.012
Vaccinated Count	0.011	0.007	0.009	0.007	0.008

Table A.2: PAWN results (Mean) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.366	0.612	0.217	0.021	0.047
Exposed Equilibrium	0.370	0.613	0.280	0.021	0.148
Infectious Equilibrium	0.366	0.612	0.217	0.021	0.047
Recovered Equilibrium	0.364	0.612	0.216	0.021	0.047
Peak Infectious Count	0.243	0.617	0.218	0.025	0.051
Time to Reach Peak Infectious Count	0.769	0.863	0.669	0.456	0.456
Masked Count	0.007	0.009	0.007	0.999	0.009
Vaccinated Count	0.876	0.868	0.873	0.873	0.874

Table A.3: Fast results (ST) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.219	0.478	0.107	0.007	0.018
Exposed Equilibrium	0.219	0.475	0.106	0.007	0.018
Infectious Equilibrium	0.219	0.478	0.107	0.007	0.018
Recovered Equilibrium	0.219	0.479	0.107	0.007	0.018
Peak Infectious Count	0.165	0.559	0.146	0.009	0.026
Time to Reach Peak Infectious Count	0.051	0.070	0.005	0.000	0.001
Masked Count	0.000	0.000	0.000	0.992	0.000
Vaccinated Count	0.000	0.000	0.000	0.000	0.000

Table A.4: RBD-Fast results (S1) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.153	0.219	0.131	0.112	0.113
Exposed Equilibrium	0.160	0.225	0.129	0.113	0.112
Infectious Equilibrium	0.160	0.220	0.130	0.112	0.113
Recovered Equilibrium	0.161	0.228	0.127	0.112	0.114
Peak Infectious Count	0.190	0.320	0.149	0.078	0.095
Time to Reach Peak Infectious Count	0.167	0.175	0.129	0.121	0.125
Masked Count	0.046	0.041	0.041	0.855	0.038
Vaccinated Count	0.038	0.040	0.034	0.039	0.041

Table A.5: Delta results (Delta) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	12324.0	49.2	16.6	24.4	47.7
Exposed Equilibrium	14951.3	51.3	17.3	29.4	45.0
Infectious Equilibrium	16994.7	49.2	16.8	24.9	36.9
Recovered Equilibrium	20444.3	49.3	16.8	23.6	50.4
Peak Infectious Count	836006.3	134.8	52.4	103.3	33.6
Time to Reach Peak Infectious Count	1510378.0	2366.6	1392.8	1217.1	5409.5
Masked Count	108512.8	199.7	36.8	17.1	16.1
Vaccinated Count	2321053.0	10395.1	5327.5	2380.4	4343.1

Table A.6: DGSM results (DGSM) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	170.1	292.9	118.4	98.6	71.2
Exposed Equilibrium	40.6	68.0	28.2	24.4	16.9
Infectious Equilibrium	71.0	121.8	49.0	40.8	29.9
Recovered Equilibrium	58.8	102.4	41.4	33.2	24.9
Peak Infectious Count	50.9	145.1	59.6	11.9	23.3
Time to Reach Peak Infectious Count	42.8	8.1	2.8	42.4	35.6
Masked Count	6.1	5.3	4.2	269.7	4.1
Vaccinated Count	0.4	0.1	2.6	0.6	5.4

Table A.7: FF results (ME) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.225	0.477	0.103	0.006	0.019
Exposed Equilibrium	0.227	0.476	0.102	0.006	0.019
Infectious Equilibrium	0.225	0.481	0.103	0.006	0.019
Recovered Equilibrium	0.226	0.480	0.103	0.006	0.019
Peak Infectious Count	0.172	0.557	0.144	0.008	0.028
Time to Reach Peak Infectious Count	0.050	0.074	0.004	0.000	0.001
Masked Count	0.000	0.000	0.000	0.992	0.000
Vaccinated Count	0.001	0.001	0.001	0.001	0.001

Table A.8: HDMR results (ST) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.200	0.201	0.200	0.200	0.200
Exposed Equilibrium	0.200	0.201	0.200	0.200	0.200
Infectious Equilibrium	0.200	0.201	0.200	0.200	0.200
Recovered Equilibrium	0.200	0.201	0.200	0.200	0.200
Peak Infectious Count	0.200	0.213	0.198	0.195	0.195
Time to Reach Peak Infectious Count	0.200	0.201	0.200	0.200	0.200
Masked Count	0.135	0.135	0.135	0.459	0.135
Vaccinated Count	0.200	0.200	0.200	0.200	0.200

Table A.9: Discrepancy results (S Discrepancy) from the Mesa model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.368	0.640	0.204	0.025	0.054
Exposed Equilibrium	0.374	0.642	0.206	0.026	0.056
Infectious Equilibrium	0.368	0.639	0.204	0.025	0.054
Recovered Equilibrium	0.365	0.639	0.202	0.025	0.054
Peak Infectious Count	0.251	0.651	0.208	0.031	0.059
Time to Reach Peak Infectious Count	0.868	0.953	0.807	0.677	0.724
Masked Count	0.008	0.008	0.008	1.00	0.007
Vaccinated Count	1.01	1.01	0.998	1.02	1.02

Table A.10: Sobol results (ST) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	457.2	642.6	313.4	85.2	138.3
Exposed Equilibrium	107.8	150.6	73.3	20.4	32.6
Infectious Equilibrium	190.6	268.2	130.8	36.0	57.8
Recovered Equilibrium	158.8	223.9	109.2	30.0	48.3
Peak Infectious Count	153.8	256.8	134.0	43.4	63.5
Time to Reach Peak Infectious Count	120.1	134.2	108.3	91.6	94.4
Masked Count	23.3	23.2	23.4	417.9	23.1
Vaccinated Count	20.6	20.8	20.7	21.0	21.0

Table A.11: Morris results (Mu Star) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.214	0.278	0.138	0.034	0.058
Exposed Equilibrium	0.216	0.276	0.136	0.034	0.058
Infectious Equilibrium	0.214	0.278	0.138	0.034	0.058
Recovered Equilibrium	0.212	0.278	0.137	0.035	0.058
Peak Infectious Count	0.199	0.300	0.147	0.034	0.060
Time to Reach Peak Infectious Count	0.105	0.184	0.063	0.021	0.032
Masked Count	0.009	0.010	0.007	0.592	0.008
Vaccinated Count	0.009	0.008	0.008	0.006	0.006

Table A.12: PAWN results (Mean) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.376	0.647	0.188	0.021	0.072
Exposed Equilibrium	0.381	0.649	0.190	0.022	0.074
Infectious Equilibrium	0.375	0.647	0.188	0.021	0.072
Recovered Equilibrium	0.373	0.646	0.187	0.021	0.071
Peak Infectious Count	0.256	0.660	0.189	0.026	0.085
Time to Reach Peak Infectious Count	0.786	0.876	0.688	0.604	0.581
Masked Count	0.007	0.008	0.007	0.999	0.009
Vaccinated Count	0.877	0.873	0.875	0.866	0.880

Table A.13: Fast results (ST) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.234	0.477	0.106	0.007	0.022
Exposed Equilibrium	0.234	0.473	0.105	0.007	0.021
Infectious Equilibrium	0.234	0.477	0.107	0.007	0.022
Recovered Equilibrium	0.234	0.479	0.107	0.007	0.022
Peak Infectious Count	0.179	0.555	0.142	0.009	0.029
Time to Reach Peak Infectious Count	0.018	0.069	0.002	0.000	0.000
Masked Count	0.000	0.000	0.000	0.992	0.000
Vaccinated Count	0.000	0.000	0.000	0.000	0.000

Table A.14: RBD-Fast results (S1) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.157	0.232	0.130	0.118	0.114
Exposed Equilibrium	0.156	0.228	0.127	0.116	0.115
Infectious Equilibrium	0.155	0.229	0.126	0.112	0.114
Recovered Equilibrium	0.162	0.243	0.126	0.110	0.113
Peak Infectious Count	0.185	0.319	0.148	0.079	0.090
Time to Reach Peak Infectious Count	0.077	0.114	0.073	0.058	0.061
Masked Count	0.034	0.042	0.047	0.855	0.042
Vaccinated Count	0.041	0.035	0.040	0.039	0.043

Table A.15: Delta results (Delta) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	6361.7	46.7	16.6	16.9	32.6
Exposed Equilibrium	7708.0	50.4	18.1	18.1	28.5
Infectious Equilibrium	8358.9	48.4	17.0	17.5	45.1
Recovered Equilibrium	11599.6	48.4	17.1	17.3	23.8
Peak Infectious Count	427728.5	226.4	116.4	79.6	41.8
Time to Reach Peak Infectious Count	3719088.0	9432.6	2753.6	1593.5	3103.0
Masked Count	389670.0	77.4	31.0	18.6	24.6
Vaccinated Count	24561060.0	10413.3	4497.6	3150.4	2736.2

Table A.16: DGSM results (DGSM) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	170.5	294.1	131.3	111.2	77.8
Exposed Equilibrium	39.3	68.6	30.3	27.9	18.6
Infectious Equilibrium	71.5	122.9	54.4	45.7	32.1
Recovered Equilibrium	59.7	102.9	45.6	37.8	27.4
Peak Infectious Count	46.1	140.0	58.6	23.5	38.0
Time to Reach Peak Infectious Count	16.9	15.6	16.3	12.7	8.3
Masked Count	0.1	2.3	1.6	277.3	1.8
Vaccinated Count	2.8	2.8	4.6	0.9	7.3

Table A.17: FF results (ME) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.289	0.541	0.122	0.008	0.024
Exposed Equilibrium	0.286	0.543	0.123	0.008	0.024
Infectious Equilibrium	0.285	0.545	0.123	0.009	0.024
Recovered Equilibrium	0.198	0.588	0.152	0.010	0.030
Peak Infectious Count	0.198	0.588	0.152	0.010	0.030
Time to Reach Peak Infectious Count	0.136	0.215	0.066	0.007	0.017
Masked Count	0.000	0.000	0.000	0.992	0.000
Vaccinated Count	0.004	0.004	0.004	0.004	0.004

Table A.18: HDMR results (ST) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.200	0.200	0.200	0.200	0.200
Exposed Equilibrium	0.200	0.200	0.200	0.200	0.200
Infectious Equilibrium	0.200	0.200	0.200	0.200	0.200
Recovered Equilibrium	0.200	0.200	0.200	0.200	0.200
Peak Infectious Count	0.200	0.215	0.197	0.193	0.194
Time to Reach Peak Infectious Count	0.200	0.201	0.200	0.199	0.199
Masked Count	0.125	0.125	0.125	0.502	0.125
Vaccinated Count	0.200	0.200	0.200	0.200	0.200

Table A.19: Discrepancy results (S Discrepancy) from the Repast model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.314	0.693	0.261	0.141	0.162
Exposed Equilibrium	0.313	0.693	0.261	0.144	0.164
Infectious Equilibrium	0.312	0.694	0.261	0.142	0.163
Recovered Equilibrium	0.311	0.694	0.263	0.139	0.164
Peak Infectious Count	0.278	0.771	0.322	0.185	0.209
Time to Reach Peak Infectious Count	0.902	0.975	0.890	0.776	0.772
Masked Count	0.078	0.078	0.074	0.997	0.074
Vaccinated Count	1.039	1.019	0.999	0.997	1.000

Table A.20: Sobol results (ST) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	35.3	63.0	32.2	16.5	19.9
Exposed Equilibrium	8.6	15.4	7.8	4.0	4.9
Infectious Equilibrium	14.3	25.6	13.0	6.7	8.0
Recovered Equilibrium	12.2	21.9	11.1	5.7	6.9
Peak Infectious Count	15.1	29.2	16.8	10.3	11.7
Time to Reach Peak Infectious Count	230.9	252.7	218.7	201.3	202.4
Masked Count	7.6	7.8	7.6	43.5	7.7
Vaccinated Count	7.2	7.3	7.2	7.2	7.2

Table A.21: Morris results (Mu Star) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.161	0.296	0.148	0.039	0.060
Exposed Equilibrium	0.161	0.294	0.149	0.039	0.060
Infectious Equilibrium	0.161	0.295	0.149	0.038	0.061
Recovered Equilibrium	0.161	0.295	0.149	0.039	0.061
Peak Infectious Count	0.129	0.296	0.151	0.036	0.057
Time to Reach Peak Infectious Count	0.099	0.180	0.067	0.017	0.030
Masked Count	0.011	0.013	0.014	0.518	0.012
Vaccinated Count	0.011	0.014	0.011	0.010	0.009

Table A.22: PAWN results (Mean) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.284	0.696	0.276	0.087	0.099
Exposed Equilibrium	0.286	0.698	0.276	0.088	0.100
Infectious Equilibrium	0.285	0.697	0.275	0.087	0.100
Recovered Equilibrium	0.283	0.696	0.276	0.088	0.100
Peak Infectious Count	0.245	0.728	0.306	0.133	0.145
Time to Reach Peak Infectious Count	0.770	0.872	0.770	0.686	0.671
Masked Count	0.067	0.066	0.066	0.992	0.065
Vaccinated Count	0.880	0.890	0.881	0.877	0.865

Table A.23: Fast results (ST) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.161	0.517	0.116	0.009	0.021
Exposed Equilibrium	0.161	0.515	0.116	0.009	0.021
Infectious Equilibrium	0.161	0.516	0.116	0.009	0.021
Recovered Equilibrium	0.161	0.518	0.116	0.009	0.021
Peak Infectious Count	0.102	0.545	0.133	0.009	0.025
Time to Reach Peak Infectious Count	0.026	0.074	0.001	0.000	0.000
Masked Count	0.000	0.000	0.000	0.926	0.000
Vaccinated Count	0.000	0.000	0.001	0.000	0.000

Table A.24: RBD-Fast results (S1) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.145	0.291	0.138	0.093	0.094
Exposed Equilibrium	0.149	0.254	0.137	0.087	0.086
Infectious Equilibrium	0.126	0.263	0.138	0.088	0.090
Recovered Equilibrium	0.134	0.260	0.135	0.082	0.086
Peak Infectious Count	0.140	0.315	0.148	0.068	0.076
Time to Reach Peak Infectious Count	0.115	0.122	0.084	0.088	0.083
Masked Count	0.034	0.051	0.044	0.661	0.048
Vaccinated Count	0.058	0.043	0.041	0.045	0.034

Table A.25: Delta results (Delta) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	100609.1	1160.0	364.1	1673.0	375.7
Exposed Equilibrium	167491.0	1176.4	376.8	1721.7	418.1
Infectious Equilibrium	174099.3	1163.6	373.9	1641.4	374.8
Recovered Equilibrium	175056.4	1153.0	375.3	1510.4	345.9
Peak Infectious Count	1203057.0	1607.5	519.8	3141.6	454.8
Time to Reach Peak Infectious Count	1696458.0	5082.5	2390.6	8157.1	70445.0
Masked Count	1068139.0	659.7	263.6	213.0	317.6
Vaccinated Count	2494313.0	10270.9	4654.7	5504.5	479.3

Table A.26: DGSM results (DGSM) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	14.1	29.3	13.2	8.4	3.7
Exposed Equilibrium	3.6	7.2	3.1	1.9	1.1
Infectious Equilibrium	5.9	11.9	5.1	3.3	1.8
Recovered Equilibrium	5.0	10.3	4.4	2.8	1.5
Peak Infectious Count	5.5	14.5	7.4	2.1	2.4
Time to Reach Peak Infectious Count	76.8	42.1	8.6	70.8	13.6
Masked Count	1.3	0.6	0.4	28.0	0.6
Vaccinated Count	0.5	0.1	0.8	0.1	1.3

Table A.27: FF results (ME) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.182	0.543	0.127	0.012	0.029
Exposed Equilibrium	0.180	0.544	0.126	0.012	0.028
Infectious Equilibrium	0.179	0.544	0.126	0.012	0.030
Recovered Equilibrium	0.180	0.544	0.126	0.011	0.029
Peak Infectious Count	0.107	0.558	0.140	0.012	0.030
Time to Reach Peak Infectious Count	0.097	0.182	0.065	0.011	0.022
Masked Count	0.001	0.001	0.001	0.925	0.001
Vaccinated Count	0.008	0.008	0.009	0.011	0.007

Table A.28: HDMR results (ST) from the RDDL model for all measures and parameters.

Measure	Isolation Rate	Basic Reproduction Number	Vaccine Infection Factor	Mask Chance	Mask Infection Factor
Susceptible Equilibrium	0.199	0.205	0.199	0.198	0.198
Exposed Equilibrium	0.199	0.204	0.199	0.198	0.199
Infectious Equilibrium	0.199	0.204	0.199	0.198	0.199
Recovered Equilibrium	0.199	0.204	0.199	0.198	0.198
Peak Infectious Count	0.199	0.206	0.199	0.198	0.198
Time to Reach Peak Infectious Count	0.200	0.201	0.200	0.200	0.200
Masked Count	0.191	0.191	0.191	0.236	0.191
Vaccinated Count	0.200	0.200	0.200	0.200	0.200

Table A.29: Discrepancy results (S Discrepancy) from the RDDL model for all measures and parameters.

Appendix B

SEIR Graphs

This appendix shows the three SEIR graphs obtained from the Repast, RDDL, and NetLogo models (with Mesa being shown directly in main body of this work). RDDL appears to be very different from the others - this is likely because due to its lowered agent count, stochasticity plays a much bigger role in how a simulation plays out. Since each probabilistic event impacts a greater percentage of agents, this makes it more likely that a disease may die out entirely or becomes endemic.

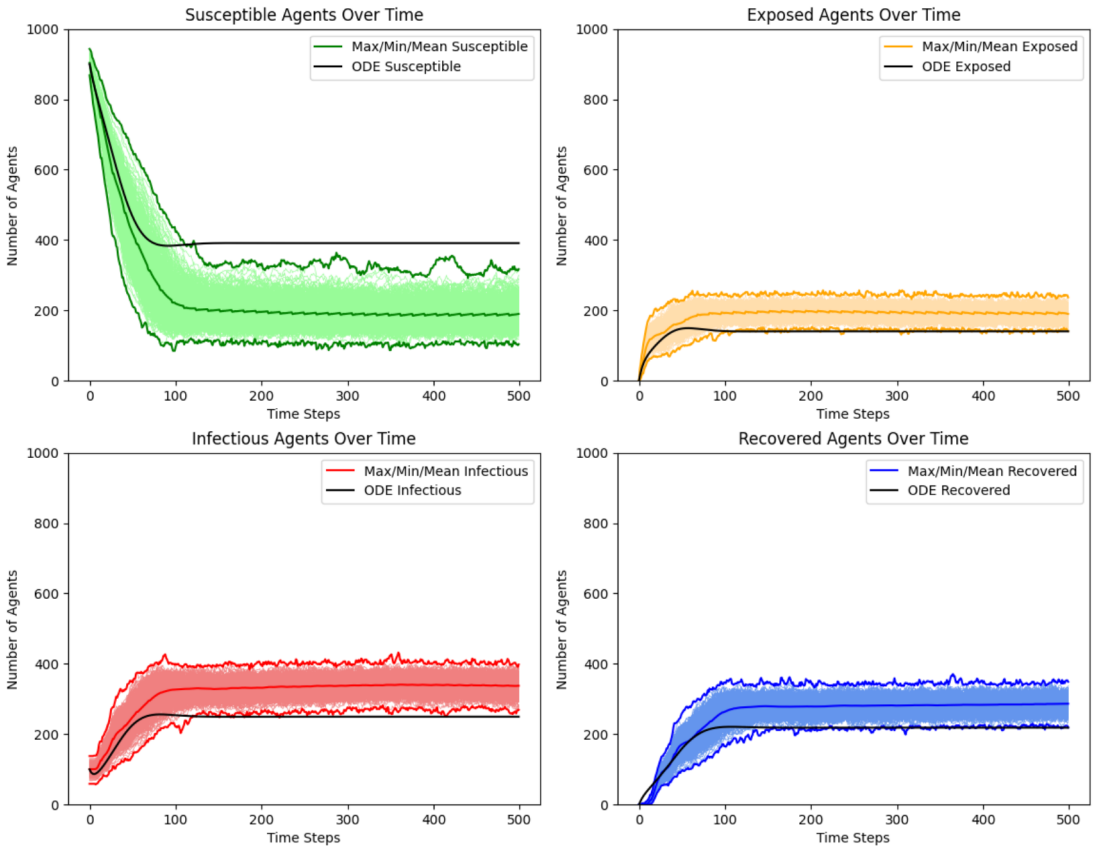

Figure B.1: SEIR graphs for the base parameter configuration on the Repast model

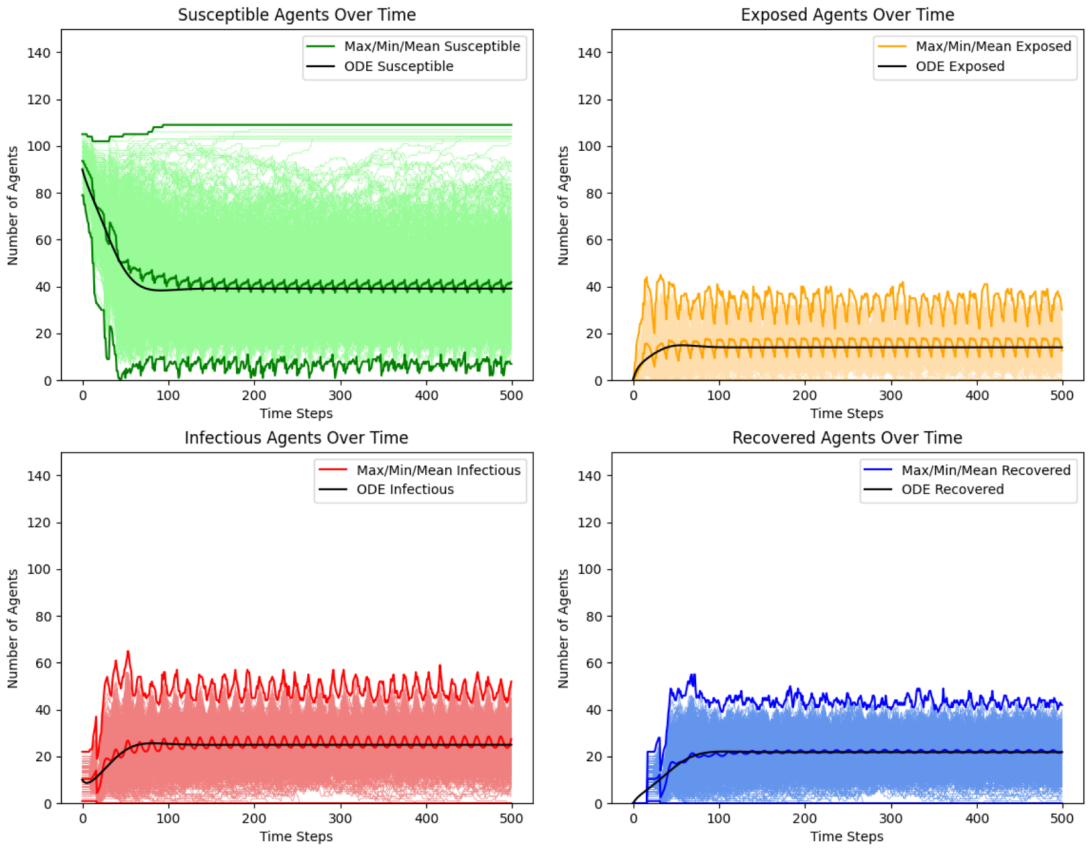

Figure B.2: SEIR graphs for the base parameter configuration on the RDDL model

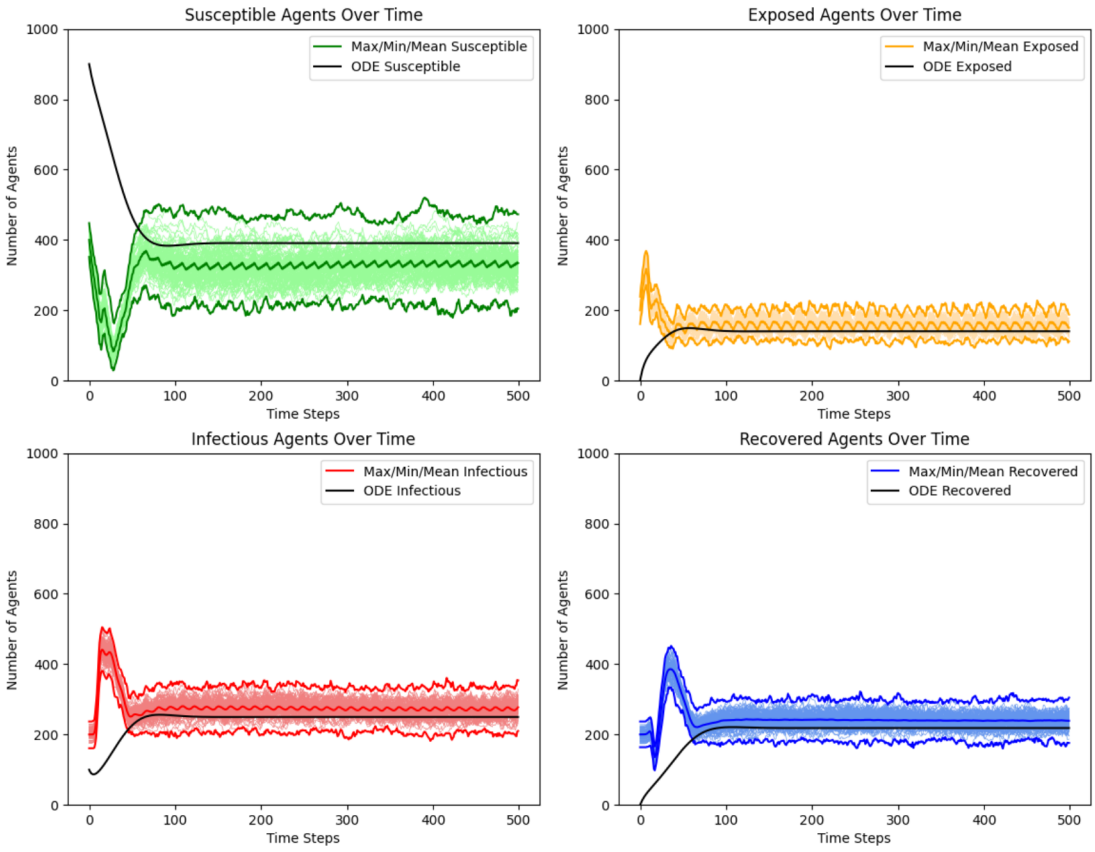

Figure B.3: SEIR graphs for the base parameter configuration on the NetLogo model

Appendix C

Stopping Point Analysis

This appendix shows the analysis performed in selecting stopping points. Each graph shows sensitivity results for the infectious equilibrium measure at varying trial counts.

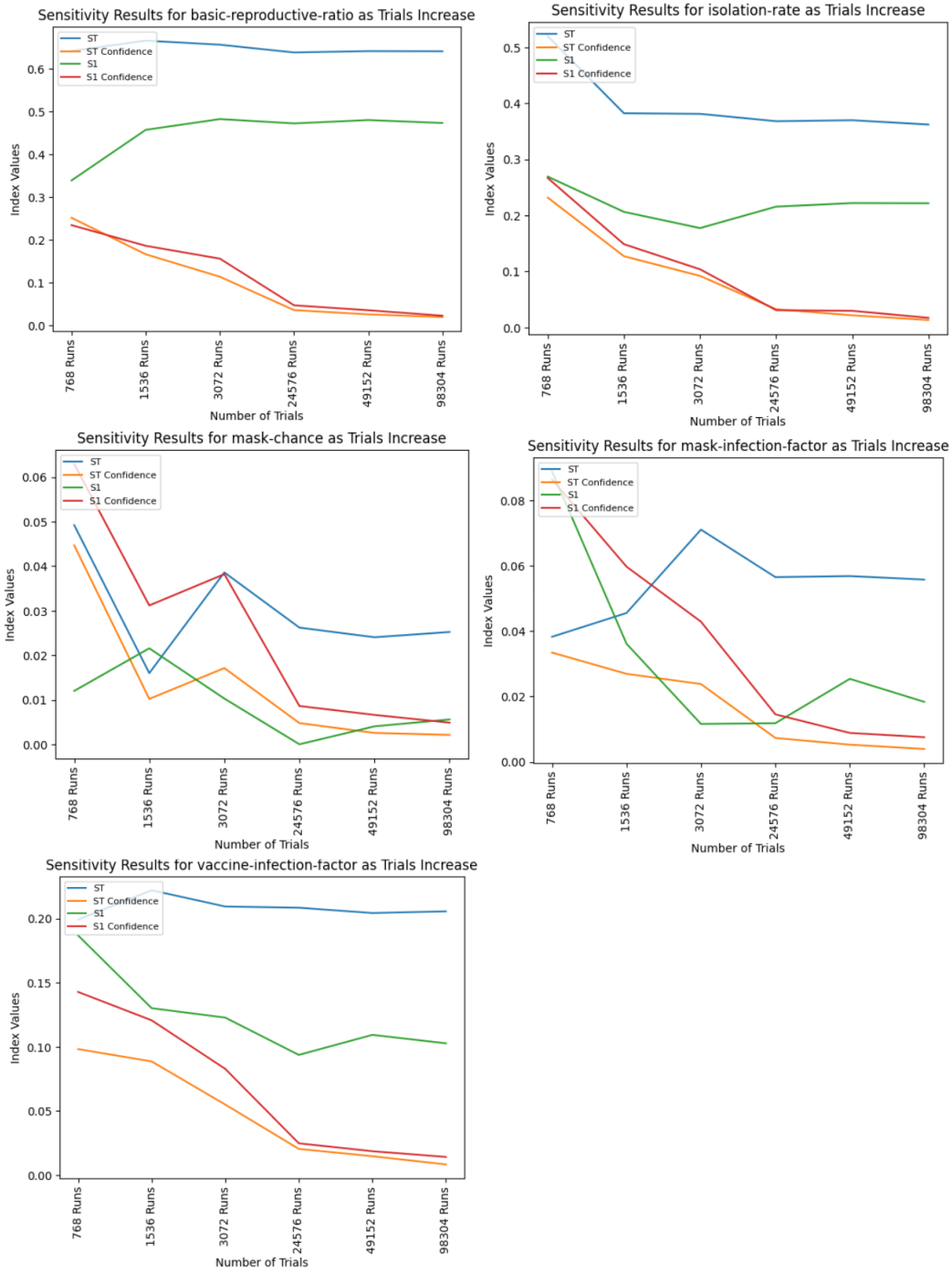

Figure C.1: Parameter sensitivity results varying trial count for the Sobol' method. In selecting stopping point, we examine the "ST" value, shown in blue on each of the five graphs.

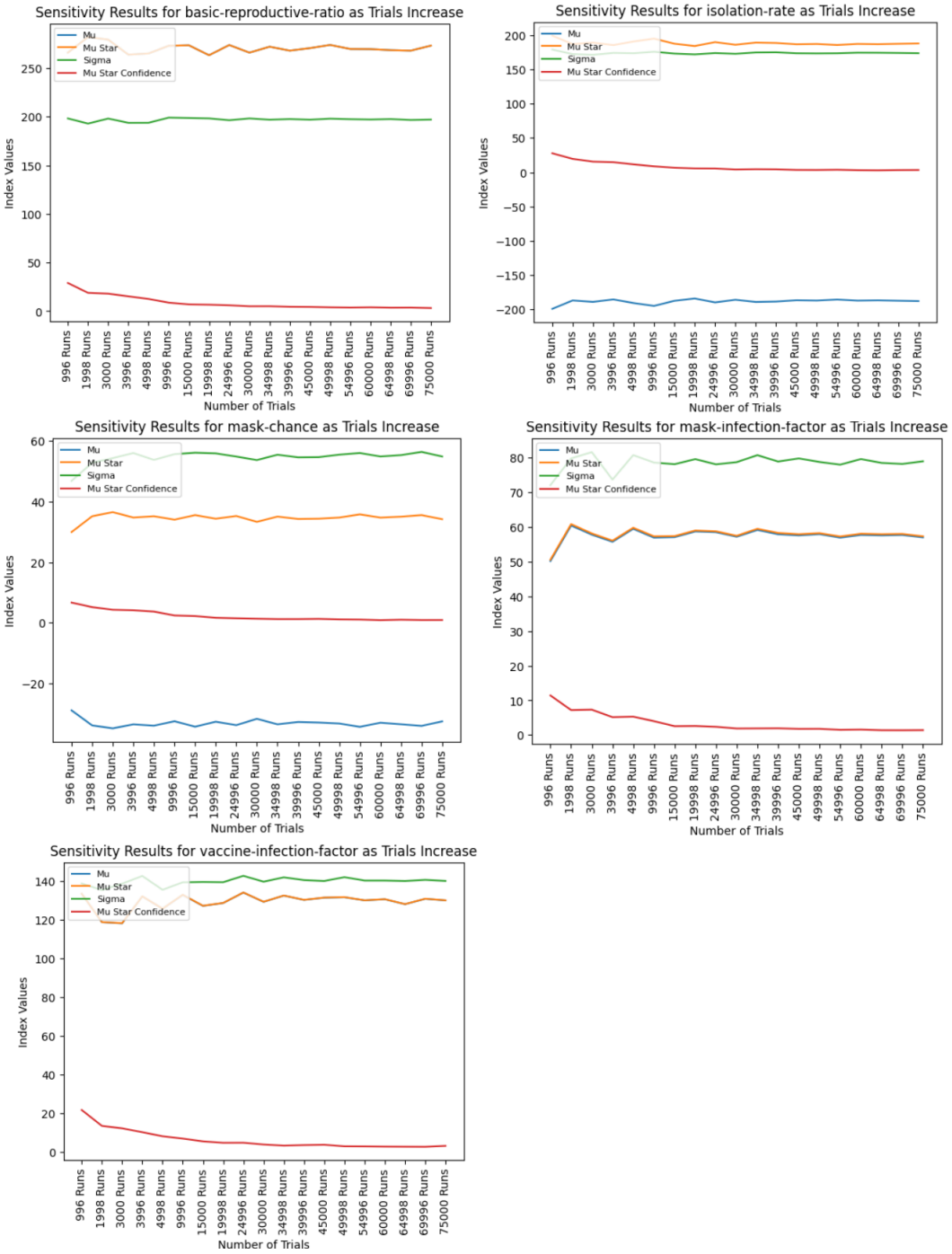

Figure C.2: Parameter sensitivity results varying trial count for the Morris method. In selecting stopping point, we examine the "Mu Star" value, shown in orange on each of the five graphs.

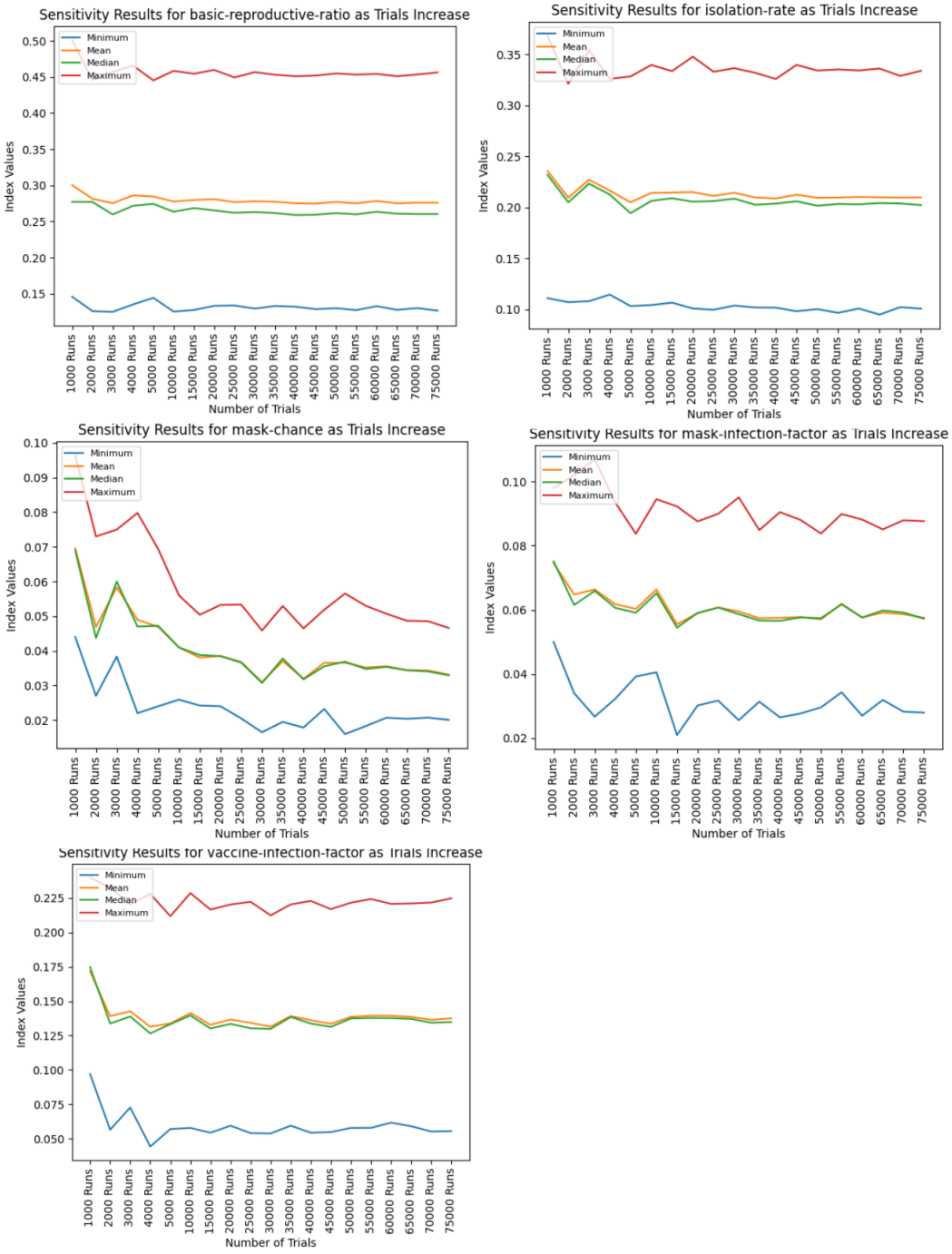

Figure C.3: Parameter sensitivity results varying trial count for the PAWN method. In selecting stopping point, we examine the "Mean" value, shown in orange on each of the five graphs.

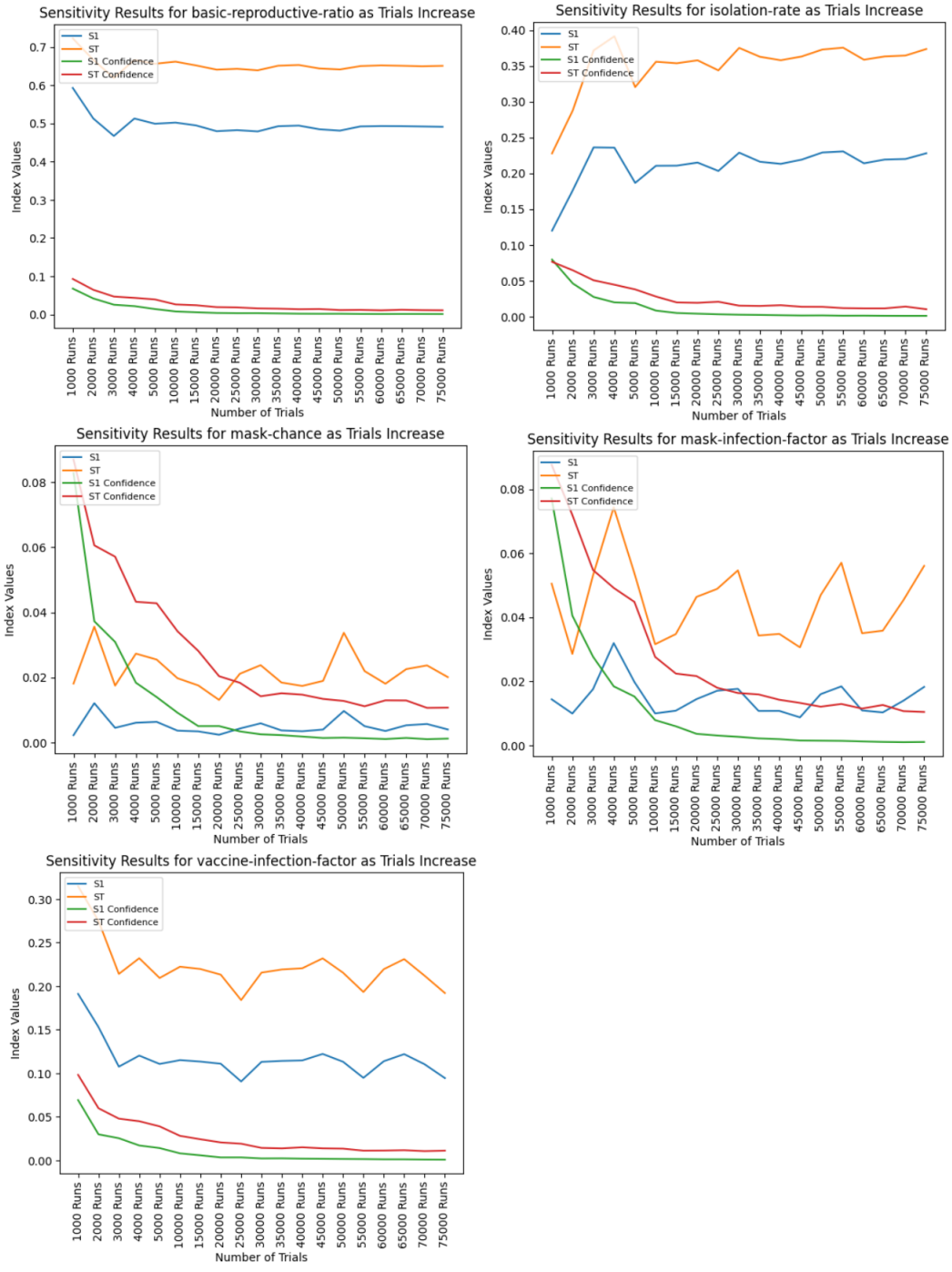

Figure C.4: Parameter sensitivity results varying trial count for the Fast method. In selecting stopping point, we examine the "ST" value, shown in orange on each of the five graphs.

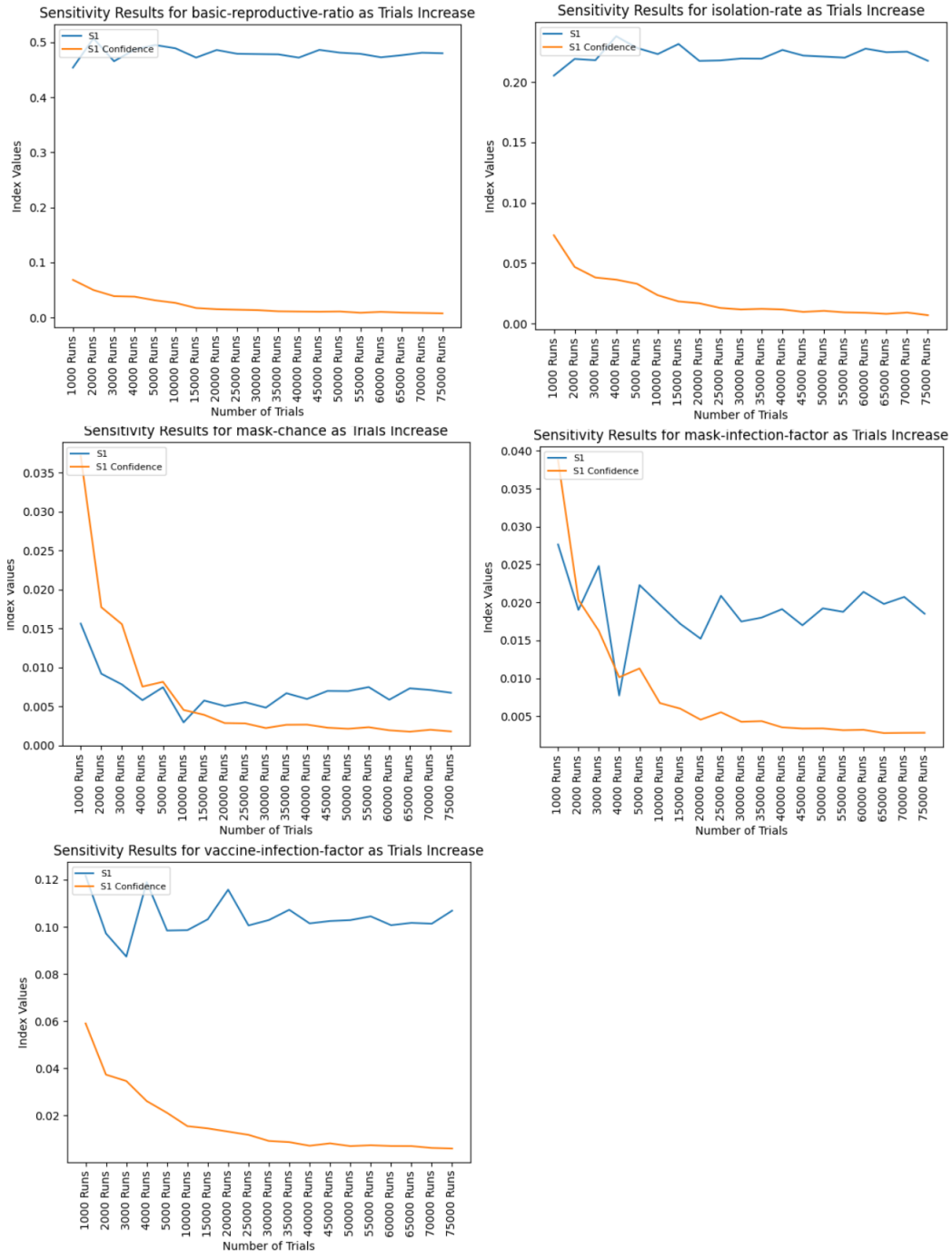

Figure C.5: Parameter sensitivity results varying trial count for the RBD-Fast method. In selecting stopping point, we examine the "S1" value, shown in blue on each of the five graphs.

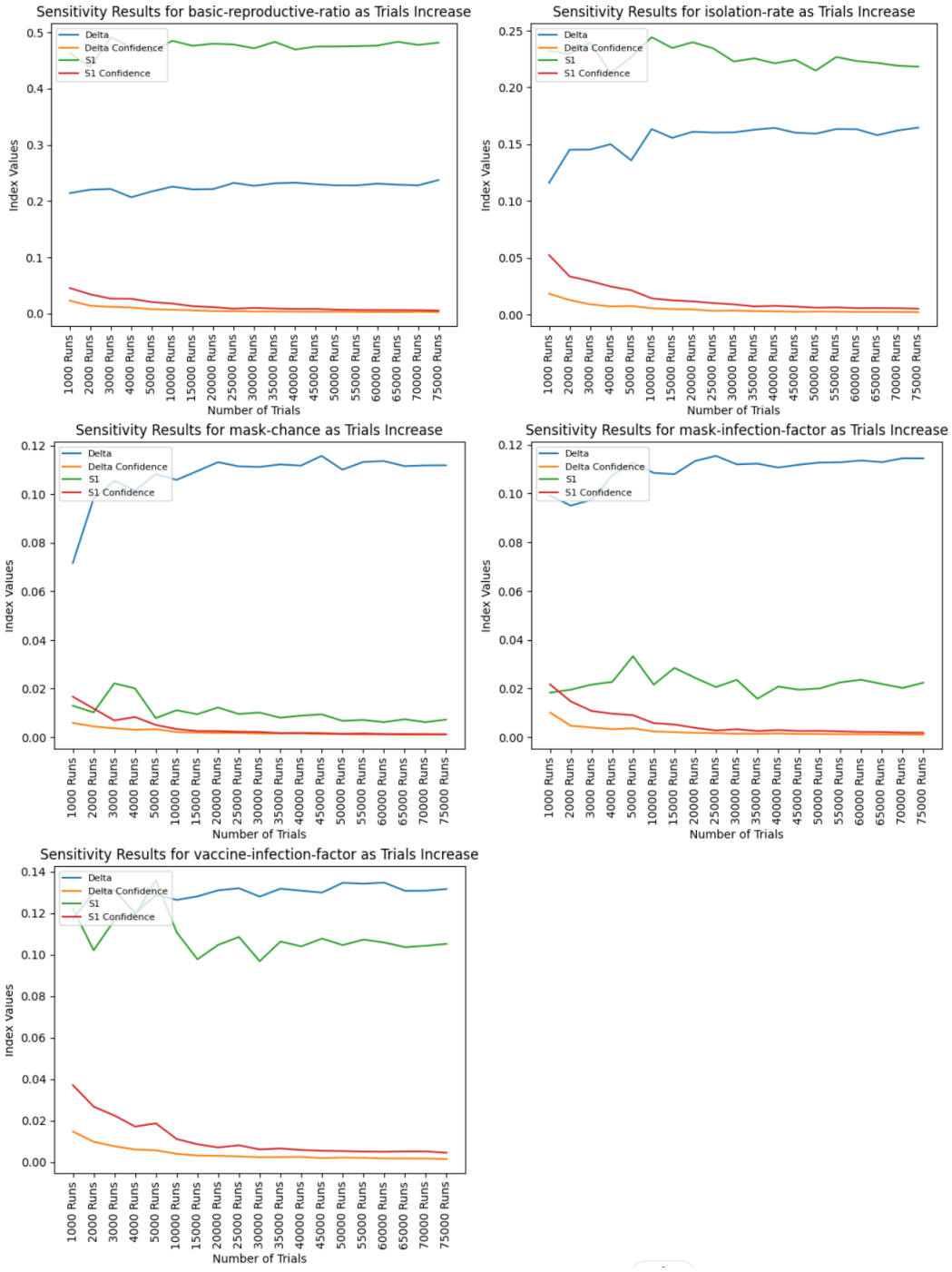

Figure C.6: Parameter sensitivity results varying trial count for the Delta method. In selecting stopping point, we examine the "Delta" value, shown in blue on each of the five graphs.

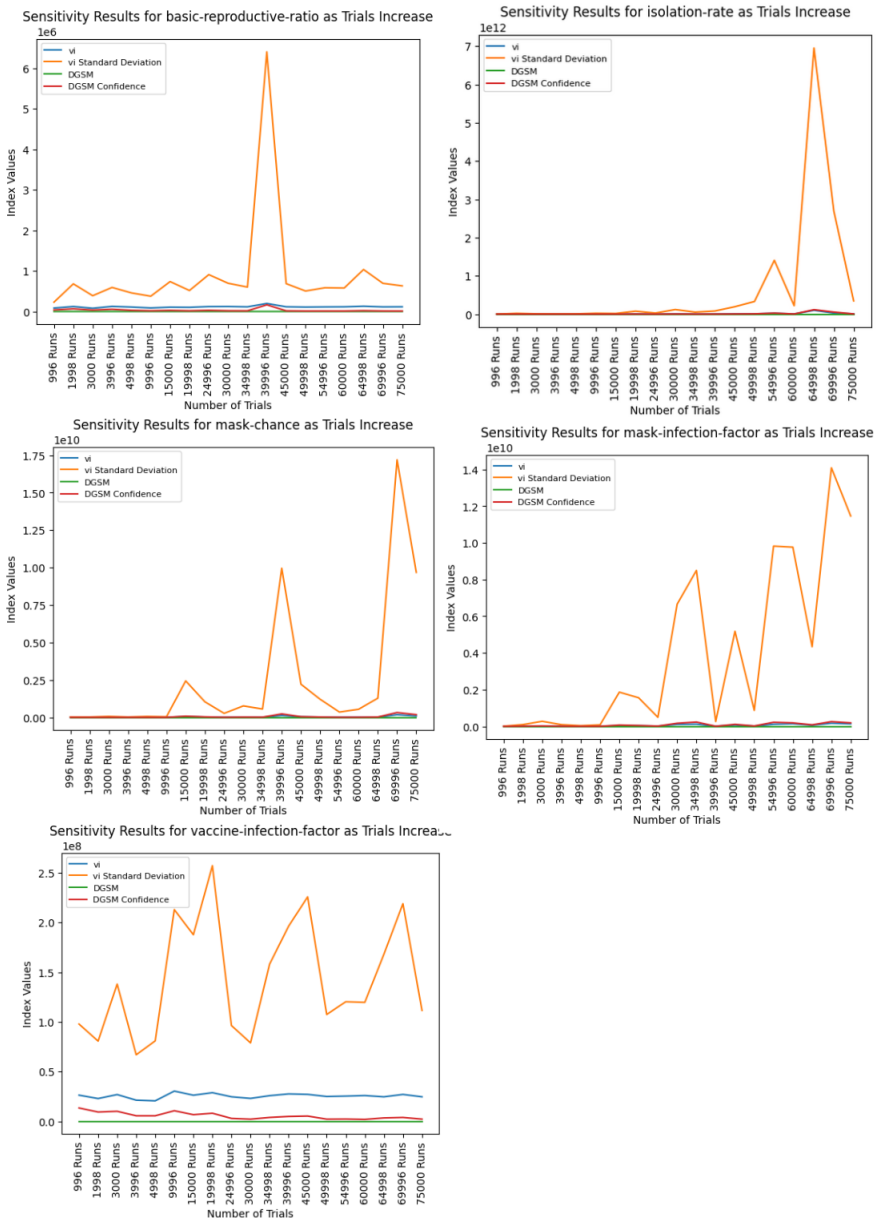

Figure C.7: Parameter sensitivity results varying trial count for the DGSM method. In selecting stopping point, we examine the "DGSM" value, shown in green on each of the five graphs.

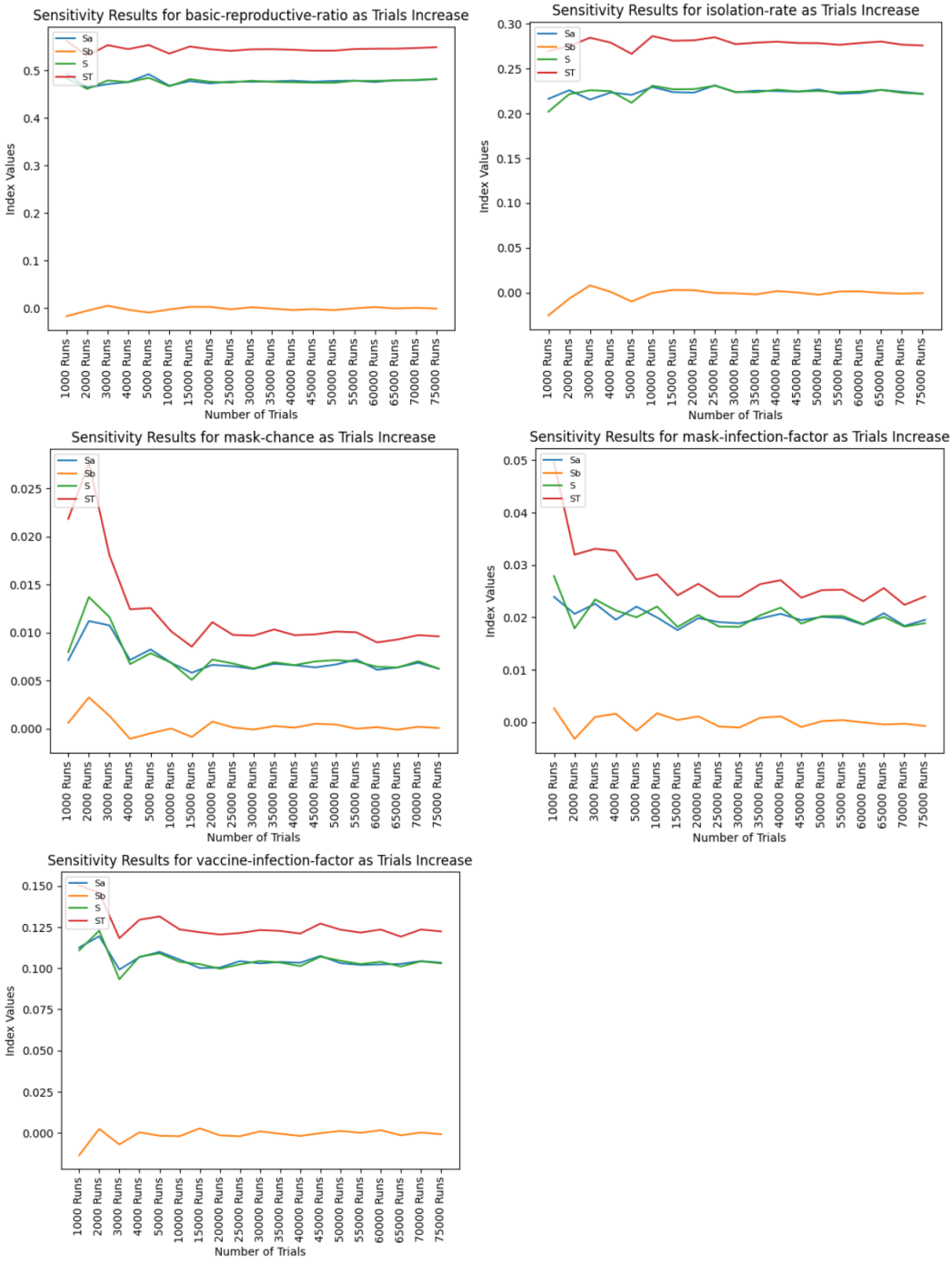

Figure C.8: Parameter sensitivity results varying trial count for the HDMR method. In selecting stopping point, we examine the "ST" value, shown in red on each of the five graphs.

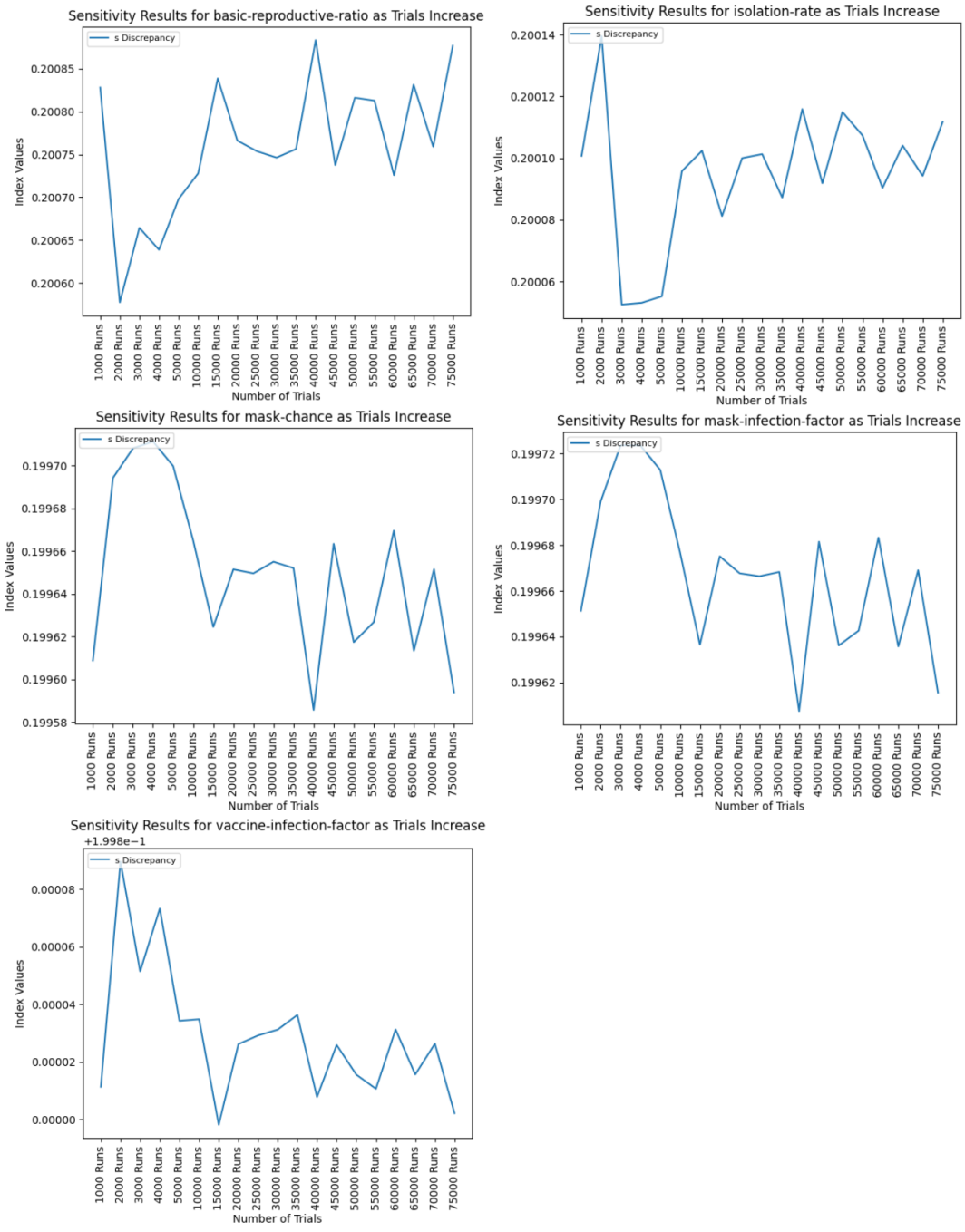

Figure C.9: Parameter sensitivity results varying trial count for the Discrepancy method. In selecting stopping point, we examine the "S Discrepancy" value, shown in blue on each of the five graphs.

Appendix D

Framework Descriptions

This appendix serves to minimally elaborate on the four agent-based modeling frameworks with respect to the complexity and capabilities related to generic model creation.

D.1 Mesa

Creating a basic Mesa model requires very little setup as far as model initialization goes - one must simply create a "Model" class, passing in "mesa.Model" as an argument, which allows for functionalities like agent scheduling to be taken advantage of. The only two pieces of Mesa-specific "setup" are performed in the class initialization method of the disease model are as follows:

```
self.schedule = mesa.time.RandomActivation(self)
self.running = True
```

All this does is set the agents to be stepped through randomly (though it does not matter in the case of this disease model, since all agents have their states updated simultaneously, which we will see in the next chapter), and set the model to start

running. It is also in the initialization method that agents are initialized, though this is entirely in the realm of traditional Python programming - agents are classes that inherit from the main "Model" class and can have their own functions per user specification. The only requirements are that once agent objects are created, they must be added to the schedule defined in the above code chunk, they must have "mesa.Agent" as an argument, and they must have a "step()" method that is automatically called on a given time step (which is activated via calling "self.schedule.step"). Apart from these mesa-specific behaviours that handle basic scheduling, the model functionality and design is entirely up to the user with minimal upkeep (in comparison to other models such as Repast, which we will see in a moment).

D.2 Repast

Creating a Repast model is significantly more involved than creating a Mesa model. The following is a code chunk at the beginning of the model initialization step:

```
#Retrieves CPU rank from space to handle per the mpirun definition
self.comm = comm
self.context = ctx.SharedContext(comm)
self.rank = self.comm.Get_rank()

#Creates the runner along with its step pattern and stopping point
self.runner = schedule.init_schedule_runner(comm)
self.runner.schedule_repeating_event(1, 1, self.step)
self.runner.schedule_stop(params['stop.at'])
self.runner.schedule_end_event(self.at_end)
```

```
#Defines the bounding box for the sim
box = space.BoundingBox(params['min.longitude'],
                        params['longitude.extent'],
                        params['min.latitude'],
                        params['latitude.extent'], 0, 0)

#Defines the space that exists within the bounding box
self.space = space.SharedCSpace(name='space',
                                bounds=box,
                                borders=BorderType.Sticky,
                                occupancy=OccupancyType.Multiple,
                                buffer_size=2,
                                comm=comm,
                                tree_threshold=100)

#Actually adds the space to the context
self.context.add_projection(self.space)

world_size = comm.Get_size()
```

These are the basic steps that are run in the disease model presented in this work that facilitate the actual running of the model, the creation of the geographical space, and the downstream agent initialization. The first three lines serve to retrieve the CPU rank that we are on. Since repast models are run using a variety of CPU

cores that come together to create one "master" model, all of the CPU cores must go through model creation locally and also create agents locally. The "rank" of the CPU is used in later agent creation in unique ID assignment, alongside the "world_size" variable defined at the bottom of the code chunk, which is equal to the number of ranks in total. The "runner" is then given some characteristics in order to specify how the model is actually stepped through from start to finish. It is set so that it begins on time step (or "tick" in the case of Repast) 1, increments by 1, and calls the user-defined "step" method that governs agent functionality. It is set to stop after a user-defined number of steps, and calls the user-defined "at_end" method at the end of the simulation. The geography must also be created, which is done by creating a "bounding box" and a "shared continuous space" that the CPUs will cover (and that agents will exist in). These are just the basic steps that get the model started. There are also other involved processes that will not be covered here, but require more specific knowledge of Repast in comparison to simply Python. For instance, agent movement between CPU ranks requires deletion and re-creation (restoration), and involves creating temporary ghost agents that may interact with other agents as they are in transition between ranks. There are also logging requirements, which control how the data is recorded and stored. These are the main functionalities that are leveraged in this disease model, but the capabilities of Repast encompass much more than what is mentioned here.

D.3 RDDL

RDDL models do not have a particularly complicated setup, but must use a unique programming language (RDDL). Shown below is a sample snippet from the disease

model created in this work:

```
    masked'(?a) =
        if (time_step == 0)
            then Bernoulli(MASK_CHANCE)
        else
            masked(?a)
;
```

This is a very simple segment of code, which governs whether or not an agent is said to be "masked" for the duration of a simulation. If the model is on the first time step, then "masked" is set to be true or false for each individual agent based on some pre-defined probability. If the model is *not* on the first time step, then "masked" is simply held at what it was in the previous time step. In RDDL, all non-static variables (i.e. ones that change over the course of a simulation) are called "fluents". This "masked" fluent is an example of a "state fluent" in contrast to "intermediate fluents". The difference between these two is that "state fluents" are calculated *on* each time step, and can refer to themselves in calculated (like how the "masked" fluent takes its own state on all time steps past the first), whereas "intermediate fluents" cannot do such a thing and are calculated *between* time steps (i.e. all intermediate fluents are calculated before any state fluents are updated on a given time step. By adding large numbers of fluents that update on every time step and can refer to each other, one can begin to create complex models fairly easily.

D.4 NetLogo

NetLogo is an application dedicated solely to agent-based modeling, and so the setup is essentially non-existent compared to the way it is with agent-based modeling frameworks developed for use in more traditional programming languages. However, in NetLogo, users must learn and use the built-in NetLogo language. Unfortunately, this language does not have the depth associated with more popular coding languages like Python or Java, and as such, users may find themselves unable to generate certain model behaviours that they wish to. An example of this is the lack of general "loops" in the way they exist in other languages. Everything must be in terms of objects (primarily agents/turtles and locations/patches), and so loops can *only* iterate over them directly. Another consequence of using a unique language is that in order for NetLogo to be run (and for the results to be analysed) using a more common language, libraries or packages must be used (should they exist) that establish this link. While this is certainly an achievable task, it does introduce a level of complexity and machine configuration that many of the other agent-based modeling tools developed such that they can be created directly using common programming languages do not have.