A Population Model of Malaria Transmission According to Within-Host Parasite Dynamics

Ryan Bradley

Abstract

We present a theoretical study of the spread of multiple strains of malaria according to within-host parasite dynamics. The disease transmission mechanism is modeled in two parts. Transmission from the host to a mosquito depends upon the host's parasite density and a transmission-to-vector probability function, while transmission from mosquitoes to new hosts depends upon a general transmission parameter which describes the behavior of these vectors. Collaborators have provided data which describe the density of parasites in the blood for two different clones of rodent malaria over the course of a typical infection; these data have been modified to reflect characteristics of the human form of the disease. The transmission-to-vector probability function is based on a half-saturation parameter which affects the overall shape of the transmission probability curves. By stochastically simulating the model over a range of half-saturation and transmission parameters, we have found that there is large region of this parameter space in which coinfections dominate the host population at equilibrium.

Acknowledgments

I would like to thank my thesis adviser, Dr. Timothy Reluga, for providing the expertise, guidance, and encouragement which made this project possible, along with Silvie Huijben and Dr. Andrew Read for the experimental data which constituted the within-host dynamics portion of the model. They provided great insight into the development of this theory. I would also like to thank Professor Andrew Belmonte for providing helpful feedback. Additionally, I am very grateful for the undergraduate research opportunities which have been provided by Department of Mathematics and the Schreyer Honors College.

Contents

1 Introduction

Malaria is a vector-borne infectious disease which is caused by protozoan parasites. Symptoms are characterized by high fever, chills, flu-like symptoms, and in many cases, death. Malaria shares many characteristics with other protozoan parasites, which cause diseases such as African trypanosomiasis and visceral leishmaniasis [32]. However, malaria is by far the most prevalent of these diseases among humans. In 2002, it was estimated that 2.2 billion people were exposed to the threat of the most dangerous species, *Plasmodium falciparum* [35]. Researchers predict that this produced between 300 and 660 million clinical malaria attacks, most of them in Africa and Southeast Asia. In addition to impacting human health, malaria also negatively impacts economic growth, making it not only a result of poverty, but also a possible contributor to poverty [17]. Given the human and economic costs of this disease, there is a great need to better understand how it spreads through large populations of human hosts.

In the following thesis, I will first present a brief survey of malaria pathogenesis (section 1.1), familiarize the reader with the mathematics of the classic endemic model (section 1.2.1), describe some of the sophisticated general modeling methods available today (section 1.2.2), and survey some of the recent malaria-specific models (section 1.2.3). Next, I will outline a model for the transmission of malaria which is based explicitly on the within-host dynamics of the disease. I will develop the model for a single strain infection (section 3), expand it to include multistrain infections (section 4), and then prove that under reasonable conditions, a coinfection of two or more strains, must become extinct (section 4.3). Since coinfections are observed in nature, I will then develop a stochastic superinfection model, in which a host can be infected with additional strains on different days (section 5). The resulting numerical experiments support the survival of coinfections. Lastly, I will summarize the opportunities for further analysis (section 6), since the work presented here represents only a fraction of the possible hypotheses which may be tested with simulations of the stochastic model.

1.1 Malaria Pathogenesis

The life cycle of the malaria parasite is complex. The infection begins when a mosquito feeds on a human host's blood. Contact from the mosquito salivary glands causes an intravenous inoculation of *sporozoites*, the protozoan cells which are infect new hosts. They are quickly transported to *hepatocytes*, the majority constituent of the liver. They incubate and multiply in the liver for a median of eleven days [41], at which point the next phase of the parasite's life cycle, an asexual *merozoite* or *daughter cell*, ruptures the host's *hepatocytes* (liver cells) and enters the bloodstream. The merozoites invade *erythrocyte*s (red blood cells), where they begin the pathogenic phase of the disease. During this phase, they are amplified to high density in the host's blood [29].

During the pathogenic erythrocyte phase, *P. falciparum* is able to evade detection by manipulating the antigens which coat the surface of the cell. The immune system has difficulty recognizing the parasite's antigenic variations, and is therefore unable to quickly dispose of it [6]. During the erythrocyte stage of the disease, the erythrocyte may adhere to the *vascular endothelium*, the thin layer of cells on the inside of blood vessels. This is thought to be one of the factors which contributes to mortality, along with the release of a diverse set of toxins [5]. During reproduction and maturation in the erythrocytes, a small proportion of the asexual merozoites convert to sexual cells which serve to inoculate mosquitoes when they feed on the blood [28]. The anopheline mosquito serves as the vector which transmits the disease to a new human host. The merozoites reproduce in the mid-gut of the mosquito before they migrate to the salivary glands, thus completing the cycle [38]. Thus, the asexual parasites undergo a "puberty" stage in human hosts, where they mature into sexual parasites, which then infect a mosquito.

There are modern drug therapies for treating malaria. These generally fall into three categories, based on the core component of the drug: quinoline, antifolate, or artemisinin. Quinine, an alkaloid originally derived from tree bark, is the oldest effective malaria treat-

ment. It was discovered in cinchona trees found in Peru in the late seventeenth century [19]. Quinine would later be used as a the most common treatment until the 1960s, at which point other drugs such as chloroquinine became available. In recent years, multi-drug resistance has revived its usefulness, and it is still in widespread use, though in combination with other drugs [3]. Evidence suggests that the quinolines suppress malaria by inhibiting the polymerization of a toxic heme called *hemozoin* that is released by the parasite, thus slowing the process by which the asexual parasites mature, reproduce, and rupture the erythrocyte [37]. Artemisinin is also believed to inhibit hemozoin or possibly target other proteins in an infected cell, however its mechanism is completely different. Research suggests that artemisinin is activated by the high concentration of parasitic heme, causing it to release free radicals which prevent polymerization [27]. Despite the variety of drug treatments, however, the emergence of resistance determines the useful life of a particular treatment, and therefore impacts the overall costs of controlling malaria.

One major difficulty concerning drug resistance is the conflict between clinical and epidemiological outcomes. The mission of a doctor or clinician is to save the patient with any available tools, however, if this is done improperly, it may accelerate the spread of drug resistance. The application of drug pressure, which encourages the selection of resistant parasites, is a key factor in promoting the emergence of resistance. Therefore, antimalarial drugs must be administered under strict guidelines to minimize the factors which enhance resistance, including poor compliance when administering multi-drug regimens, the use of poor treatment or sub-therapeutic doses, or the overuse of treatments, especially when a diagnosis of malaria has not been confirmed in a lab. For these reasons, epidemiologists recommend restricted use of massive drug administration, heavy use of post-treatment follow-up, and synergistic use of partner drugs and alternative treatments [43].

Since it is difficult to create effective malaria treatment policies on a case-by-case basis, epidemiologists have urged the creation of centralized databases which will track clinical,

in vitro, and pharmacological data in order to create better malaria management policies [33]. An effective treatment strategy would provide for both the treatment of patients while preserving the efficacy of today's antimalaria drugs. However, this goal has been elusive. Today, the newest drug treatments, which are based on a compound called artemisinin, are being used in combination with older drugs such as quinoline and antifolate to lengthen the useful lifespan of artemisin [31].

It is clear that a fundamental understanding the mechanism by which malaria replicates and transmits is essential to maintaining or improving the current state of malaria control. By combining an understanding of the malaria mechanism with better surveillance and rational treatment policies, it may be possible to reduce the enormous human and economic costs of malaria.

1.2 A Review of Infectious Disease Models

Because of the potency of malaria, the emergence of drug-resistant strains, and the many factors which influence the transmission and progression of malaria endemics, infectious disease modeling provides a valuable tool for exploring new ways to control disease. However, because the individual processes which comprise an infectious disease – such as the dynamics of parasite replication in the host and the nature of contacts between hosts – occur over large time and length scales, it is difficult model the disease as a whole. Even the simplest infectious disease is influenced by a stunning array of factors. It is therefore useful to study infectious disease with the simplest relevant models in order to implement them easily, and moreover, to elucidate their fundamental processes. Let us now briefly survey of the field of infectious disease modeling, in order to better place the forthcoming malaria model in the proper context.

When surveying the collection of disease models, it is helpful to classify infection processes according to three components: biological, behavioral, and environmental [18]. The

biological component of infectiousness comprises the nature of the pathogen's life cycle, its dynamics in the body of the host, its interaction with the host immune system, potency and physical damage or mortality in the host, and susceptibility to drug treatments. The behavioral component is defined on a much larger length scale, and consists primarily of the contact patterns between hosts and vectors. Lastly, the environmental component of infectiousness includes those components which determine transmission between hosts, or if necessary, between hosts and vectors. For example, malaria requires a mosquito vector to transmit between human hosts. Since mosquitoes thrive in wet environments, their population is much larger during a rainy season. In some cases, pesticides influence the mosquito populations as well. For several decades following the 1950s, the pesticide dichloro-diphenyl-trichloroethane (DDT) was used as an effective means for controlling malaria by eradicating large populations of mosquitoes. However, in recent years, the environmental damage of DDT along with mosquito resistance has led to more flexible forms of disease management [42]. For example, malaria can be prevented by using a mosquito net or bed-curtain, as this reduces the number of interactions between a human host and potential mosquito vectors, though efficacy hovers around 17% [25]. In addition to the biological and behavioral components of the disease, these environmental components may also impact the population-level dynamics by influencing the transmission rate. To understand the propagation of infectious diseases, an understanding of each component is necessary.

The product of an infectious disease model will be a description of how the infectious disease propagates through a population, according to initial conditions and the parameters which describe the biological, behavioral, and environmental properties of the disease. By surveying the model predictions across different parameters and initial conditions, we may better understand the behavior of the model, and also evaluate its compatibility with our observations.

1.2.1 The Classic Endemic Model (SIR)

Many epidemiological models are based upon a compartmental model which distinguishes among individuals according to their disease state. We will now explore the classic endemic model, called the "SIR" model because it provides a useful starting point for understanding the theory of infectious diseases.

The SIR model is perhaps the simplest example of an epidemiological compartment model. The acronym *SIR* stands for the three classes of hosts, which include those who are susceptible, infected, and recovered from the disease. Hethcote provides a useful review of this model, which will now be briefly summarized [22]. We may describe the model with the following initial value problem.

$$\begin{array}{rclcrcl} \frac{dS}{dt} &=& \mu N - \mu S - \beta \frac{IS}{N} & & S(0) &=& S_0 \geq 0 \\ \frac{dI}{dt} &=& \beta \frac{IS}{N} - \gamma I - \mu I & & I(0) &=& I_0 \geq 0 \\ \frac{dR}{dt} &=& \gamma I - \mu R & & R(0) &=& R_0 \geq 0 \\ N &=& S + I + R & & & & \end{array} \tag{1}$$

The variables S, I, and R represent the populations of susceptible, infected, and recovered hosts. This model includes balanced birth and death rates given by the inflow of newborns into the susceptible class with a rate of μI and deaths in each class, at rates of μS, μI, and μR. The parameter μ represents the number of births and deaths per unit time per person, which gives a mean lifetime of $\frac{1}{\mu}$ time units. We have set $N = S+I+R$ in order to conserve population at a constant quantity, since this forces $\frac{dS}{dt}+\frac{dI}{dt}+\frac{dR}{dt} = \mu N - \mu(S+I+R) = 0$. Note that births and deaths enable this system to represent an *endemic* disease, in which the infection is able to sustain itself indefinitely. In this case, the persistance of infection is thanks to the supply of new susceptible individuals by birth. Without birth and death processes, eventually all hosts would contract and recover from the disease. This is called an *epidemic*, to distinguish it from an endemic. The assumption that population size is

constant is a strong one; a fluctuating population may have a profound effect on malaria transmission. For example, in a growing population, births may provide an ever-increasing pool of new susceptible hosts, while a shrinking population may force the disease to become extinct when the susceptible population becomes too small to sustain it.

The transmission parameter β is the contact rate, defined as the number of disease-transmitting contacts per unit time per infected host. The number of new infections per unit time is equal to the total number of disease-transmitting contacts βI multiplied by the probability that the recipient is susceptible, which must be $\frac{S}{N}$ in a well-mixed population. The recovery parameter γ represents the number of recoveries per unit time, per person. This means that $\frac{1}{\gamma}$ is the expected duration of the disease, implicitly assuming an exponential waiting time $e^{-\gamma t}$ for recovery. The SIR model therefore captures behavioral *and* biological components of the disease in β because it includes the rate at which hosts interact with each other, scaled by the probability that an infected host will transmit the disease to a susceptible one. Additionally, γ completes the simple biological picture of the disease by specifying its infectious duration. We have summarized the properties of our model, along with a collection of necessary assumptions in the following list.

- The population of N hosts is well-mixed, meaning that it is equally probable for any two hosts to come into contact.
- We assume that the population is sufficiently large that the size of each class can be treated as a continuous variable.
- The product of the rate of contacts between all hosts and the probability of disease transmission if one host is infected is β.
- Birth and death are exactly balanced, and happen at a rate μ – which implies an expected lifespan of $\frac{1}{\mu}$ – such that the exponential waiting time for a birth or death event is $e^{-\mu t}$.

- The expected duration of the disease is $\frac{1}{\gamma}$ time units.
- There is no additional migration into the system or out of the system.
- The disease is not fatal, and hence only natural death depletes the number of infected hosts.

The behavior of the classic SIR model can be classified in terms of the *basic reproduction ratio* R_0. The *basic reproduction ratio* is the average number of secondary infections created when a single infected host is placed in an entirely susceptible population. Intuition suggests that if $R_0 < 1$ then the infection will not be able to grow in our population. If we consider Equation 1 we might guess that $R_0 = \frac{\beta}{\gamma+\mu}$ because this is the average number of disease-transmitting contacts by a single individual over the death-adjusted duration of the disease. However, we can show that our intuition is correct by describing the equilibria and local stability of the system. We will show that whenever $R_0 = \frac{\beta}{\gamma+\mu} \leq 1$ the system will approach a disease-free state, and whenever $R_0 = \frac{\beta}{\gamma+\mu} > 1$ the infection will persist. In considering the local stability of the system, we are neglecting the possibilities of limit cycles which would require more elaborate solution methods.

First we normalize our ordinary differential equations according to $\bar{S} = S/N$, $\bar{I} = I/N$, and $N = 1$, which is equivalent to setting $\bar{S} + \bar{I} + \bar{R} = 1$, where $\bar{S}$, $\bar{I}$, and $\bar{R}$ represent proportions of the total population which are susceptible, infected, and recovered. We ignore the recovered class, which can be calculated from the susceptible and infected classes thanks to conservation of population. The normalized equations are as follows.

$$\begin{aligned} \frac{d\bar{S}}{dt} &= \mu - (\beta\bar{I} + \mu)\bar{S} \\ \frac{d\bar{I}}{dt} &= \beta\bar{S}\bar{I} - (\mu + \gamma)\bar{I} \end{aligned} \tag{2}$$

If we set these equal to zero, we may solve for two equilibria, (S_∞, I_∞).

$$\begin{aligned} \bar{I}_\infty &= 0 && \text{and} \quad \bar{S}_\infty = 1 \\ \bar{I}_\infty &= \tfrac{\mu}{\mu+\gamma}(1 - \tfrac{\mu+\gamma}{\beta}) && \text{and} \quad \bar{S}_\infty = \tfrac{\mu+\gamma}{\beta} \end{aligned} \tag{3}$$

Note that these equilibria are only positive, and hence physical, when $\mu + \gamma < \beta$. If we linearize the system about either of these equilibria, we may determine the asymptotic stability properties from the eigenvalues of the resulting coefficient matrix. We set $x = \bar{S} - \bar{S}_\infty$ and $y = \bar{I} - \bar{I}_\infty$. Recognizing that at equilibrium, $\frac{d\bar{S}}{dt} = \frac{d\bar{I}}{dt} = 0$ we may linearize our system by taking the Taylor series expansion of x and y. This yields the following linear system.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -\beta\bar{I}_\infty - \mu & -\beta\bar{S}_\infty \\ \beta\bar{I}_\infty & \beta\bar{S}_\infty - \mu - \gamma \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

If we substitute the disease-free equilibrium condition, $\bar{I}_\infty = 0$, we find that the eigenvalues are $\beta - \gamma - \mu$ and $-\mu$. For an asymptotically stable system, the solution to our linearization must tend to zero as $t \to \infty$; this happens if and only if the eigenvalues have a negative real part. Applying this condition allows us to conclude that $\frac{\beta}{\gamma+\mu} < 1$ implies that disease-free equilibrium will be stable, while it will be asymptotically unstable if $\frac{\beta}{\gamma+\mu} > 1$.

Instead of showing the rather lengthy eigenvalues for the *endemic equilibrium state* – the one with a nonzero number of diseased hosts – it is sufficient to show that the coefficient matrix has a positive determinant and a negative trace. The trace of this matrix is $-\frac{(\beta+2\gamma)}{\gamma+\mu}$ and the determinant is $\mu(\beta - \gamma - \mu)$, which must always be positive if we assume that the disease is shorter than the average lifespan. Therefore, the endemic equilibrium is asymptotically stable as long as $\beta > \mu + \gamma$.

Note that since $\bar{S}_\infty = \frac{\mu+\gamma}{\beta} < 1$ is necessary for an endemic equilibrium to be non-negative, and hence realistic, it follows that $\frac{\beta}{\gamma+\mu} > 1$ and the disease-free equilibrium must

be unstable. In this case, intuition suggests that if $\frac{\beta}{\gamma+\mu} > 1$ then the system will approach the endemic equilibrium. While a proof is beyond the scope of this report, it has been shown that this is true for the more general SEIR model, and in fact, the endemic equilibrium is globally asymptotically stable[39].

To summarize, we have surveyed the SIR model, and shown that the invasion of an infection in a disease-free population will occur when the basic reproduction number, $R_0 = \frac{\beta}{\gamma+\mu}$, is greater than unity. We have thus classified the two possible behaviors of this system in terms of its physical parameters. Even when we cannot estimate these parameters, there is much to learn from understanding the behavior of a disease which follows this model.

1.2.2 Sophisticated Model Design

A preponderance of biological evidence has led epidemiologists to conclude that most infectious are more complicated than the simple SIR model suggests. Rigorous statistical methods and careful observation of the progression of infectious diseases in nature has led to ever more nuanced infectious disease models, which capture the complexity of the biological, behavioral, and environmental components of infectiousness.

Adding compartments to the model is one way to better mimic the disease. For example, the SEIRS model incorporates an exposed state in which an individual host has contracted the disease but is not yet infectious. It also returns recovered hosts into the susceptible state after a certain time, emulating the temporary immunity which is observed in many diseases [21]. In models such as these, the time spent in exposed, infectious, and recovered states may have a profound effect on the dynamics of the model. We may further modify the compartmental model by including mortality which is caused by the disease, and must hamper its infectiousness by reducing the population of infected hosts. Some have also considered models with variable infectivity according to the population size [40]. This is critical, as it is closely related to the idea of herd immunity, which may play an extensive

role in any infectious disease.

Many important infectious diseases – namely Malaria, but also yellow fever, the West Nile virus, and dengue – require transmission between hosts by a vector such as a mosquito. Mathematical models for these diseases must also consider both the biology and behavior of the vector as well as the host. It is oftentimes possible to treat the vector population with set of differential equations which resemble the same human host compartments [10].

Epidemiologists may also incorporate more complex behavioral and environmental components into their models. For example, they may tweak the incidence rates of births, deaths, and immigration [21]. Another effective modeling technique involves the stratification of hosts by age, since different age cohorts have different levels of infectiousness and mortality when infected with the same disease [23]. The mixing effects of a population can be explored by using spatially explicit models [24]. In many cases, such models predict a richer collection of different disease behaviors [30]. In the case of vector-borne diseases, seasonal forcing effects and climate changes may affect the vector, and change the overall disease dynamics [2, 36].

Given an apt model, it is possible to study the impact of treatments on the spread of the disease. This can lead modelers to propose optimal treatments, especially when the optimal treatment pattern is counter-intuitive. Research into influenza models provides a useful example. A large-scale epidemic simulation based on the basic reproduction number of past pandemics was used to describe the progression of after the initial outbreak in either the UK and US. This revealed that even intense border control would only delay the onset of the pandemic, while rapid treatment and household quarantine provide the greatest impact on transmission [11]. In the case of severe acute respiratory syndrome (SARS), researchers have shown that the success of disease control measures such as isolation and contact tracing depends on both the inherent transmissibility of the infectious agent and the proportion of transmission which occurs before the onset of symptoms. It is necessary to determine this

proportion in order to effectively manage the outbreak [12]. These two examples are only a small sample of the possible treatment strategies which may be elucidated from infectious disease models.

1.2.3 A Survey of Malaria Models

We have previously discussed infectious disease models generally, however, it is important to consider each disease specifically, since every infectious disease may have unique characteristics. Malaria is a distinct infectious disease for many reasons, namely its transmission via mosquitoes, two-stage parasite life-cycle, and the proliferation of several different strains. Because of its relative complexity modelers have explored the behavior of the malaria endemic in a number of ways. Some of the most recent models explore the more complicated aspects of the disease. Animal models and models from other diseases are also used to draw conclusions about malaria, since direct testing of humans would often be unethical.

A primary area of concern is immunity. In one example, modelers created age-stratified immunity functions which depend upon the frequency of infection [44]. These functions controlled the level of parasites in an infective host, thereby impacting transmission and changing the behavior of the model. Another model looks more closely at the immune system response, seeking to emulate the antigenic variation which the parasite uses during the erythrocyte phase [8]. Additionally, some modelers have created models which probe the effects of cross-immunity between different strains [15].

Finally, some modelers have attempted more comprehensive models, as seen in a recent model which simulates the parasite densities according to experiments in which neurosyphilis patients received therapeutic *P. falciparum* infections. This particular model has been designed to capture as many biologically relevant features as possible, including seasonality, age-stratified vector contact rates, and the effects of heterogeneous transmission rates [34].

2 Objectives and Hypothesis

We model a population of human hosts who contract and transmit malaria. We assume that transmission between hosts depends upon the density of malaria parasites in the blood, commonly called the *within-host dynamics* of the disease. This model is based on experimental data which describe the daily parasite densities for mice infected with up to two strains of malaria. We translate these data to match the features of the disease in a human host, and then simulate the spread of these strains in a population of human hosts. The model tests the hypothesis that the specific *shape* of the within-host dynamics, represented as the parasite density with respect to the age-of-infection, will have a profound effect on the dynamics of the disease. By surveying the spread of the disease across different shapes and levels transmission, we can draw conclusions about the way in which the spread of the disease depends on the shape of the within-host dynamics.

3 The Single-Strain Model

In order to model the transmission of malaria based on the within-host dynamics, we begin with the simplest possible model, which consists of two mechanisms. The host transmits a malaria parasite to a vector – in this case, a mosquito – and the vector transmits the disease to a new host. To ensure that we distinguish between diseased and recovered hosts, we assumed that the parasite density is zero after $(m+1)$ days, after which time the individual is permanently resistant to the disease. In practice, the parasite density may never reach zero; it may persist indefinitely in small amounts. In other cases, individuals may lose their resistance to the disease over time, effectively becoming naive again. The simple model discards these cases in favor of simplicity. We will choose m based upon the point at which the disease is at very low levels in the blood.

For each individual expressing the disease, the probability f of transmitting the disease to the vector in a single interaction, a mosquito bite, is a function which depends on the parasite density at day a, denoted $p(a)$. The transmission rate β is a factor which encompasses the remaining aspects which are implicated in transmission from the vector to a new host, including:

- the rate at which the vector interacts with the hosts
- the probability that the vector passes the disease to a new host in a single interaction
- the vector survival rate
- the relative number of humans and vectors

The model effectively separates the two processes which cause the disease to propagate: transmission to the vector and infection of a new host. The former depends upon experimental data which describe the parasite density and estimates of the transmission-to-vector probability function f, while the latter is subsumed into the vector-to-host transmission rate

β. We assume that β is independent of the age of infection, and that the population is well-mixed, or absent of any spatial heterogeneity which might make β vary. The transmission parameter β and the two parameters which we will use to define the transmission-to-vector probability function will be difficult to estimate. For that reason, we will later study the way the model behaves across a broad range of such parameter values, to consider all possible natural disease outcomes.

For the single-strain model there are three classes of hosts: those who are naive to the disease, those with an active infection, and those who have recovered. As the infection runs its course, there is a per-day survival rate of $s = S^{\frac{1}{(m+1)}}$ where S is the overall survival rate for the disease. This implicitly assumes that there is an equal probability of mortality for all days of the infection. Hosts in the naive and recovered states are also subject to a natural per-day survival rate of s_n. This assumes an expected lifespan of $1/s_n$ years. In order to conserve the population size, hosts who die from either the disease or natural causes are "recycled" to the naive class, thus emulating birth and death while assuming a constant population. This induces a natural turn-over effect which causes the disease to reach an equilibrium instead of becoming extinct. This is somewhat unrealistic, since natural populations grow and shrink. Our model assumes that there is no age-stratification, and that mortality and transmission do not vary throughout any groups within the host population. Many of our assumptions trade accuracy for simplicity in order to focus on our fundamental hypothesis, that the shape of the curve which describes the transmission-to-vector probability from parasite densities will have an effect on the steady-state outcomes of the disease.

3.1 The Single-Strain Equations

We will first consider a model with only one strain of malaria, outlined in Figure 1. The following equations represent such a model, according to the assumptions and definitions

given earlier in this section. At time t, the variables $y(a,t)$ represent the number of hosts with an a-day-old infection while $w(t)$ and $r(t)$ represent the naive and recovered populations, respectively.

$$\begin{aligned} w(t+1) &= w(t) - \beta\, w(t) \sum_{a=0}^{m} y(a,t)\, f[p(a)] \\ &\quad + (1-s_n)\, r(t) + (1-s) \sum_{a=0}^{m} y(a,t) \qquad (5) \\ y(0,t+1) &= \beta\, w(t) \sum_{a=0}^{m} y(a,t)\, f[p(a)] \qquad (6) \\ y(a,t+1) &= s\, y(a-1,t) \qquad \forall\, a = 1,\ldots,m \qquad (7) \\ r(t+1) &= s_n\, r(t) + s\, y(m,t) \qquad (8) \end{aligned}$$

Equation 6 represents the new, zero-day-old infections. New infections are the product of three factors. The number of individuals with an a-day-old infection multiplied by the per-person probability f of passing the disease on to the vectors gives the total number of vectors likely to have acquired the disease from those individuals. For example, if the per-person probability of passing the disease to a vector is 50%, and there are 10 individuals in the infected group, then there will be a population of 5 vectors carrying the disease. After summing these terms over all infected groups (each with varying days of infection), we multiply by the per-vector transmission rate β in order to find the number of infected hosts. If the rate is 0.2 then there will be 1 newly infected individual in our example. This scheme approximates the way in which vectors spread the disease, without explicitly modeling them.

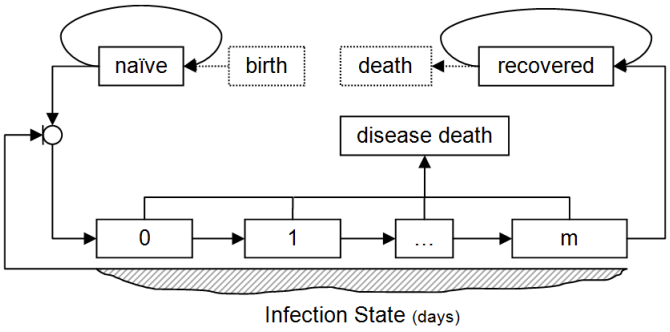

Figure 1: This diagram shows the migration of hosts from the naive state, through diseased states - enumerated by the number of days of infection - to a recovered state. Hosts who do not survive during the infection or after recovery are recycled to the naive state.

Equation 7 represents the course of infection, in which hosts with an a-day-old infection on day t have an $(a+1)$-day-old infection on day $t+1$ with probability s, the disease survival rate. The variables $y(a,t)$ for each a from 1 to m reflect transient states; hosts can only have an a-day-old infection for one day. The naive and recovered variables, however, retain hosts. Equation 8 defines $r(t)$, the recovered class, which receives hosts who survive the final day of infection and retains hosts from the previous time-step with a probability equal to the background survival probability, s_n. Finally, the naive class $w(t)$ changes according to Equation 5. It retains the naive hosts from the previous time-step and loses some to infection. The final two terms serve to balance the system and conserve the size of the population. The first term replaces those who were lost to the natural survival rate in the recovered class, while the second replaces those who were killed by the disease.

3.2 The Probability of Transmission to a Vector

The function $f(x)$ describes the probability that the host with a parasite density of x viruses per microliter of blood will transmit the disease to a vector in a single interaction, such as a mosquito bite. This function, given by equation 9, takes two parameters: q, the *shape*

parameter and c, the *half-saturation density*.

$$f(x) = \frac{x^q}{c^q + x^q} \tag{9}$$

The shape parameter determines the slope and "sharpness" of the probability function, while the half-saturation density is the parasite density which corresponds to a probability of $\frac{1}{2}$. Figure 2 shows an example of the range of possible probability functions which can be achieved with only two parameters. This function is vital to the behavior of the model because it translates the parasite densities $p(a)$ into the probabilities of infecting a mosquito during a single interaction during the a-th day of the infection. We have hypothesized that the shape of the parasite density curves with respect to time affects the dynamics of the disease. Therefore, $p(a)$ links our model to the biology of the infection. Section 5 and Appendix A describe the experimental data for $p(a)$, which were provided by collaborators.

Given the flexibility of this function, it has the power to reshape the transmission response in a number of ways. A cutoff which is small relative to the parasite density will ensure that transmission occurs in almost all cases, while a high cutoff ensures that transmission is incredibly rare. Since the fundamental purpose of this study is to draw conclusions about the spread of malaria in large populations from the *shape* of the function which tracks parasite density with time, it is important to choose these parameters wisely. Otherwise, the heavy-handedness of this function will obscure their fundamental shape, and the model will degenerate into a simpler one, in which the probability of transmission is near unity or zero for a certain duration of the course-of-infection.

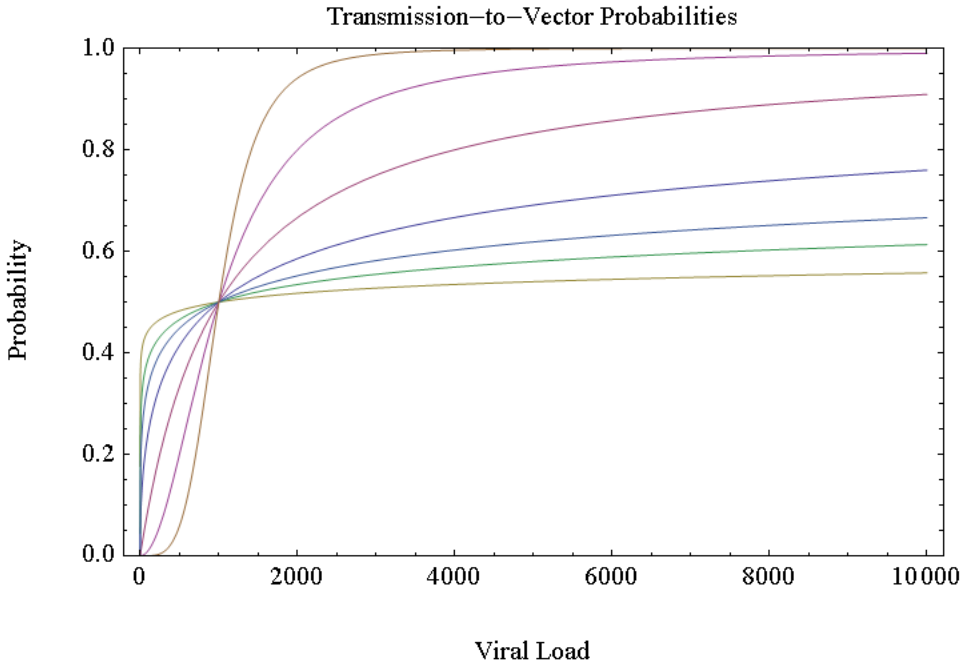

Figure 2: A depiction of different probability curves for the function f. Curves which approach 1 at very large parasite densities (the viral load) have power parameters (q) which are less than 1. Curves which approach 1 at lower parasite densities have shape parameters (q) which are greater than 1. The half-saturation (c) is 1000, the parasite density for which each function returns a probability of $\frac{1}{2}$.

3.3 Single-Strain Equilibrium

Now that we have outlined the single-strain equations, we would like to describe the disease patterns at equilibrium. We can do this by looking for stationary solutions to the equations. The following shows the re-written equations, with no reference to time.

$$\begin{aligned}
\bar{w} &= \bar{w} - \bar{y}(0) + \bar{r}(1 - s_n) + (1 - s)\sum_{a=0}^{m} \bar{y}(a) \\
\bar{y}(0) &= \beta\, \bar{w} \sum_{a=0}^{m} \bar{y}(a)\, f[p(a)] \\
\bar{y}(a) &= s\, \bar{y}(a-1) \qquad\qquad \forall\, a = 1, \ldots, m \\
\bar{r} &= s_n\, \bar{r} + s\, \bar{y}(m)
\end{aligned}$$

To solve the equations, we first eliminate all $\bar{y}(a)$ variables for $a > 0$. We may then solve for a trivial equilibrium by factoring, to get $0 = \bar{y}(0)(1 - \beta\bar{w}\sum_{a=0}^{m} s^a f[p(a)])$, the disease-free equilibrium. To solve for the remaining diseased equilibrium we will rewrite each of the above equations. Since the system conserves population, the total population size N is determined by the initial conditions.

$$\bar{w} = [\beta \sum_{a=0}^{m} s^a \, f[p(a)]]^{-1} \tag{10}$$

$$\bar{y}(a) = s^a \bar{y}(0) \tag{11}$$

$$\bar{r} = \frac{s^{(m+1)}}{(1 - s_n)} \bar{y}(0) \tag{12}$$

$$N = \bar{w} + \bar{y}(0)[\frac{s^{(m+1)}}{(1 - s_n)} + \sum_{a=0}^{m} s^a] \tag{13}$$

Since population is conserved, the number of infected and recovered hosts will depend on the population size. These three equations ensure that the system is properly constrained, and provided that $\bar{w} < N$, the system will have a unique positive stationary solution. If $\beta = 0$, then the system cannot create new infections, rendering it degenerate. In this case, Equations 10-13 will not apply due to division by zero. With no infection rate, the stationary solution would have no active infections, a naive population given by the initial conditions, and a recovered population equal to the number of initially infected hosts. This equilibrium is unstable; a single new infection is enough to force the system to gravitate towards a different steady-state.

4 The Multistrain Model

Given the single-strain model, we would like to expand it to include an arbitrary number of strains. This model will include both coinfections and strain immunity history. Hosts with a *coinfection* can be infected with a set of strains $z \subseteq Z$, instead of just a single strain. Each strain is distinguished by its parasite density, which is denoted $p_{i,z}(a)$ for the a^{th} day of infection with strain $i \in z \subseteq Z$ where Z is the set of all strains and z is a subset of strains which constitute a coinfection.[1] If there is only one strain in z then we call it a *single* infection. In this section we will assume that all coinfections are simultaneous; that is, a host can only have a coinfection if it has received both strains from another infecting host. It is important to distinguish between the parasite densities not only between single strains, but also among different coinfections, because it is likely that multiple strains will interact which each other within the host. [2]

We assume that after a host has recovered from a particular strain, it can no longer become re-infected with it. To include such immunity history, hosts who have recovered from a set $w \subseteq Z$ may carry active infections with another set $z \subseteq Z$, so long as these sets are disjoint (that is, $z \subseteq w^c$). We will use $y_{z,w}(a,t)$ to denote the population of hosts recovered from strain set w but infected for a days with strain set z at time t. We define $r_z(t)$ as the hosts who have recovered from strains $z \subseteq Z$. For example, if $Z = \{f,g,h,i\}$ the set of hosts naive to all strains is $r_\emptyset(t)$ and those who have recovered from strains $\{g,i\}$ but are still naive to the complement of this set $\{f,h\}$ are denoted by $r_{\{g,i\}}(t)$. This ensures that hosts who have recovered from some strains may still be infected by the others. In

[1] We will use set notation to describe the collection of strains which infect the members of each class of hosts. In set notation, $x \in Z$ means "x is an element of Z" and is used to describe individual strains in the set of all possible strains, Z. The set $z \backslash x$ represents the set of z minus those elements in x. The expression $z \subseteq Z$ means "z is a subset of Z" and represents any collection of strains in the set Z including the absence of any strains (the null set, $\emptyset$) or all possible strains, Z. The expression $z \subset Z$ is identical, but does not allow z to contain *every* strain in Z. If we take our entire space of strains to be Z, then the term "x complement" or x^c is equal to $Z \backslash x$.

[2] In section 4.3 we will show that coinfections will vanish if the viral loads and survivability are the same across all coinfections.

addition to these definitions for z and w, we will also denote a host's full complement of strains, of which the host may only transmit a subset, with the letter x where $z \subseteq x$.

We will distinguish between different survival rates for infection with each set of strains, with $s_z(a)$ representing the per-day survival rate for an individual with an a-day-old infection of strain set $z \subseteq Z$. Including dependence upon the age-of-infection in the survival rates will allow us to model the mortality of each course of infection more accurately. For simplicity, the transmission parameter β is independent of time. However, in future work it may be wise to extend the model by allowing β to fluctuate, specifically with seasonal fluctuations during which mosquito populations – and hence transmission – rise and fall.

The model is not fully general because it will only propagate coinfections which originate from the same host - two hosts with different strains cannot infect a new host with both. This means that the course of infection must begin with coinfected hosts, otherwise they will never emerge on their own. In some cases, this assumption may be reasonable, since the probability of the same vector acquiring the disease from separate hosts could be remote if the population of vectors is much larger than the population of hosts. This limitation also provides an opportunity to study the way in which different values for the transmission rate β affect the relative populations of hosts with single- and multiple-strain infections.

4.1 Equations for the Multistrain Model

The following equations describe the multistrain model with simultaneous-only coinfections and strain-specific immunity history, but no cross-immunity. The sets are x, w, and z are defined in the previous section. The v index is used to account for all hosts with active infections x, because they may have recovered from any $v \subseteq x^c$.

$$
\begin{aligned}
y_{z,w}(0,t+1) &= \beta\, r_w(t) \sum_{x \supseteq z} \sum_{v \subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\, \Phi_{x,z,w}(a) \quad \forall z \neq \emptyset,\, \forall w \subseteq z^c \qquad (14)\\
y_{z,w}(a,t+1) &= s_z(a)\, y_{z,w}(a-1,t) \qquad \forall\, a \in \{1,\ldots,m\} \quad \forall z \neq \emptyset,\, \forall w \subseteq z^c \qquad (15)\\
r_w(t+1) &= s_n\, r_w(t) + \sum_{z \subseteq w \setminus \emptyset} s_z(m+1)\, y_{z,w\setminus z}(m,t) \qquad \forall w \subseteq Z\\
&\quad -\beta\, r_w(t) \sum_{z \subseteq w^c} \sum_{x \supseteq z} \sum_{v \subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\, \Phi_{x,z,w}(a)\\
&\quad +\delta_{w\emptyset}[(1-s_n) \sum_{x \neq \emptyset} r_x(t) + \sum_{x \neq \emptyset} (1-s_x(t)) \sum_{v \subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)] \qquad (16)\\
\text{where } \Phi_{x,z,w}(a) &= \prod_{i \in z} f(p_{i,x}(a)) \prod_{j \in x \setminus (z \cup w)} [1 - f(p_{j,x}(a))]
\end{aligned}
$$

These equations are composed of the same elements as the single-strain equations. Equation 14 describes hosts with active infections z who have previously recovered from w, drawing from the corresponding recovered population, r_w. This class can become infected with z by any host with an active infection x which includes z ($x \supseteq z$), any immunity to set v such that $v \subseteq x^c$, and any age-of-infection $a \in 0, ..., m$, hence the triple summation. Equation 15 describes the course-of-infection for each active infection z in hosts recovered from w. Lastly, equation 16 describes the states of recovered individuals. This includes four terms, the last of which (preceded by $\delta_{w\emptyset}$) is described in section 4.1. The first term is used to retain recovered hosts from the previous step. The second term adds hosts who have survived infections with z and previously recovered from infections with $w \setminus z$. The third term subtracts hosts who have recovered from w, but received a new infection with z. The hosts in r_w can become infected with any $z \subseteq w^c$ by any host with active infection $x \supseteq z$, with any immunity to set $v \subseteq x^c$, and any age of infection $a \in 0, ..., m$.

The Probability Function Φ In the multistrain equations, $\Phi_{x,z,w}(a)$ represents the probability that an individual infected with strain set x transmits set z and fails to transmit the set $x\backslash(z \cup w)$ to a vector during a single interaction (note that $z \subseteq x$ by our formulation). This probability must be independent of set w because the function $\Phi_{x,z,w}(a)$ will always be used to transmit new infections to the recovered class r_w. Since this class has recovered from w and has permanent immunity, its probability of becoming infected with z does not depend upon whether w is transmitted. Therefore, w must be omitted from the strains which fail to transmit – $x\backslash(z \cup w)$ – and cannot be present in the set of strains z which are transmitted, because $z \subseteq w^c$. This function allows one to compute the probability that a group of hosts passes a subset of its active strains to new hosts who may already be immune to some of these strains.

Conservation of Population In order to conserve population, equation 16 includes the following term:

$$\delta_{w\emptyset}[(1-s_n)\sum_{x\neq\emptyset} r_x(t) + \sum_{x\neq\emptyset}(1-s_x(t))\sum_{v\subseteq x^c}\sum_{a=0}^{m} y_{x,v}(a,t)]$$

This term recycles hosts who have died from infection or natural causes into the naive class, $r_\emptyset$. The leading factor is a Kronecker delta, which is equal to 1 when $z = \emptyset$ and is equal to 0 when $z \neq \emptyset$. This allows one to include the "recycling term" for the equation which describes $r_\emptyset$.

Creating a Time Series From the Difference Equations The difference equations in 4.1 can be used to track the spread of the disease. There are some practical concerns which make them somewhat unwieldy, however. If we imagine a system in which transmission is high or the amplitude of population oscillations is large, then it is possible for the population of naive hosts to become negative. This nonphysical result is due to the fact that each

strain of the disease blindly draws new hosts from the naive population during each time step, without regard to infections by the other strains. Without exploring new methods for constraining the difference equations, the best way to

4.2 The Linear Multistrain Model

In this section, we will consider a simplified linear version of this system. Section 4.1 outlines the nonlinear system with a limited but conserved population. We can remove this nonlinearity if we investigate the behavior of the system in which a new set of infections has just been introduced to a naive population. We make the following assumptions.

- The number of infections is small relative to the overall population.
- The population of hosts is entirely naive, and the number of naive individuals is constant: $r_\emptyset = P$. It is not depleted by new infections.
- It follows that no recovered populations exist and all active infections have an empty infection history: $r_w = 0 \; \forall w \neq \emptyset$ and $y_{z,w}(0) = 0 \; \forall w \neq \emptyset$.

In this case, the new infections are linear in the number of current infections of each type. Arranging these equations into a matrix equation will furnish eigenvalues which will be equal to the *initial velocity* or initial growth rate of each strain. The equations are as follows.

$$\begin{aligned} y_{z,w}(0, t+1) &= \beta(t)\, r_w(t) \sum_{x \supseteq z} \sum_{v \subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\, \Phi_{x,z,w}(a) \quad \forall z \neq \emptyset, \forall w \subseteq z^c \\ y_{z,w}(a, t+1) &= s_z(a)\, y_{z,w}(a-1, t) \qquad \forall\, a \in \{1, \ldots, m\} \quad \forall z \neq \emptyset, \forall w \subseteq z^c \end{aligned}$$

If we rewrite these as a matrix equation, the resulting physical eigenvectors would represent the initial relative populations of each $y_{z,w}(a)$. The eigenvalues therefore correspond to the rate at which each type of new infection grows linearly.

4.2.1 A Simple Example

To illustrate the relationship between eigenvalues and initial velocity, consider a two-strain system with three days of infection. Written in a matrix, this system will have the following form: $Y(t+1) = M \times Y(t)$. For simplicity, there are no infection history indices, no variables referring to individuals who recovered from a previous infection, and a uniform mortality rate s. The strains are numbered *"1"* and *"2"* respectively. Note that this matrix can be simplified into block form. If we extract the dominant eigenvalue from the submatrices along the diagonal, we are left with the growth rate for the initially-infected hosts of that type. This growth rate will depend on β and will serve as a criteria for the invasion of the strain into a naive population. Any eigenvalue which is greater than one will grow in the population. Otherwise the strain will fail to invade.

$$M = \begin{bmatrix}
\Phi_{1,1}(0) & \Phi_{1,1}(1) & \Phi_{1,1}(2) & 0 & 0 & 0 & \Phi_{\{1,2\},1}(0) & \Phi_{\{1,2\},1}(1) & \Phi_{\{1,2\},1}(2) \\
s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & s & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \Phi_{2,2}(0) & \Phi_{2,2}(1) & \Phi_{2,2}(2) & \Phi_{\{1,2\},2}(0) & \Phi_{\{1,2\},2}(1) & \Phi_{\{1,2\},2}(2) \\
0 & 0 & 0 & s & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & s & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \Phi_{\{1,2\},\{1,2\}}(0) & \Phi_{\{1,2\},\{1,2\}}(1) & \Phi_{\{1,2\},\{1,2\}}(2) \\
0 & 0 & 0 & 0 & 0 & 0 & s & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & s & 0
\end{bmatrix}$$

$$Y(t) = \begin{bmatrix} y_1(0,t) & y_1(1,t) & y_1(2,t) & y_2(0,t) & y_2(1,t) & y_2(2,t) & y_{\{1,2\}}(0,t) & y_{\{1,2\}}(1,t) & y_{\{1,2\}}(2,t) \end{bmatrix}$$

4.2.2 When Is the System Linear?

This analysis will only apply as long as the naive population is significantly larger than the number of infected hosts. When this is true, new infections do not significantly reduce the naive population and therefore do not noticeably affect the rate of new infections. In this case, the size of the system is important because we can only introduce integer numbers of infected hosts for physical reasons. Moreover, the smaller the system, the smaller the

window of time in which this assumption is valid.

The higher the population ceiling, the longer the system will seem linear, since nonlinearities become most apparent when the number of infected and recovered hosts is a large proportion of the overall population. At the very least, we expect the system to be linear at the first instant of new infections. It will also seem linear during the initial phase of infection, when all infections are increasing in a mostly naive population.

4.3 Without Interactions, Coinfections Vanish

In this section, we will show that coinfections must eventually become extinct if we assume that the parasite densities are the same for every strain, regardless of whether or not it is a member of a coinfection, and that the survivability of a coinfection is not higher than infection by some single strain. That is, we define $p_i(a) = p_{i,z}(x)\ \forall z \subseteq Z$. Now consider the largest possible coinfection, which contains every strain in Z. We will treat transmission probability independent of time and simplify the equation further by including survival-rate terms.[3] New coinfections with this entire set will follow the following form.

$$\begin{aligned} y_{Z,\emptyset}(0,t+1) &= \beta\, r_\emptyset(t) \sum_{a=0}^{m} y_{Z,\emptyset}(a,t)\, \Phi_{Z,Z,\emptyset}(a) \\ y_{Z,\emptyset}(0,t+1) &= \beta\, r_\emptyset(t)\, y_{Z,\emptyset}(0,t) \sum_{a=0}^{m} s_z(a)\, \Phi_{Z,Z,\emptyset}(a) \end{aligned}$$

The equation for the largest coinfection is different than smaller coinfections and single infections because it has only one term. This is a consequence of the fact that the largest simultaneous coinfection can only arise in completely naive hosts. At steady-state, this

[3] Here we define $s_x(0) = 1$ in order to keep make these equations more readable; this allows us to remove references to $y_{z,w}(a)$ for $a > 0$. The survival rate is defined physically as $s_x(a)\ \forall a \in 1, ..., a_m + 1$, the probability of surviving from the a^{th} day of infection to the following day. It follows that $s_x(a_m + 1)$ is the probability of surviving the final day of infection, to become a recovered host.

equation gives a condition involving the steady state of $r_\emptyset$, shown below.

$$0 \;=\; y_{Z,\emptyset}(0)\,[1-\beta\, r_\emptyset \sum_{a=0}^{m} s_z(a)\,\Phi_{Z,Z,\emptyset}(a)]$$

This implies that either coinfections do not exist at steady state, that is $y_{Z,\emptyset}(0)=0$, or $r_\emptyset = [\beta \sum_{a=0}^{m} s_z(a)\,\Phi_{Z,Z,\emptyset}(a)]^{-1}$. This steady-state condition ensures that pool of completely naive hosts is large enough so that the coinfection will not decrease. Now consider the equation for one of the single-strain infections $z_1 \subseteq Z$ which were formerly naive to all strains. Thus z_1 is a singleton and $w=\emptyset$. In the following equations, we substitute our steady-state $r_\emptyset$ condition.

$$\begin{aligned}
y_{z_1,\emptyset}(0,t+1) &= \beta\, r_\emptyset(t) \sum_{x\supseteq z_1} \sum_{v\subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\,\Phi_{x,z_1,\emptyset}(a) \\
y_{z_1,\emptyset}(0,t+1) &= \frac{\sum_{x\supseteq z_1} \sum_{v\subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\,\Phi_{x,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} \\
y_{z_1,\emptyset}(0,t+1) &= \frac{\sum_{a=0}^{m} y_{z_1,\emptyset}(a,t)\,\Phi_{x,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} + \frac{\sum_{x\supseteq z_1, x\neq z_1} \sum_{v\subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\,\Phi_{x,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} \\
y_{z_1,\emptyset}(0,t+1) &= \frac{y_{z_1,\emptyset}(0,t)\sum_{a=0}^{m} s_{z_1}(a)\,\Phi_{z_1,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} \\
&\quad + \frac{\sum_{x\supseteq z_1, x\neq z_1} \sum_{v\subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\,\Phi_{x,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)}
\end{aligned} \tag{17}$$

Equation 17 expands the summation over x into two parts, when $x=z_1$ and all other cases, $x\neq z_1$. The first term represents the new z_1 infections which are caused by other z_1 infections. Coinfections are considered in the second term, which must be nonnegative because live infections cannot inhibit new infections in any way; moreover, each factor is nonnegative. Let us now assume that the coinfection with Z cannot be more survivable than coinfection with z_1. That is, $s_{z_1}(a) \leq s_Z(a)\ \forall a$. With each day of infection, individuals infected with every strain are not less likely to die than those with only one strain, z_1.

This is reasonable, as it assumes that the strains do not inhibit each other or reduce their individual potency.

Now consider the relationship between $\Phi_{z_1,z_1,\emptyset}$ and $\Phi_{Z,Z,\emptyset}$. Assume that $p_{z_1,Z}(a) = p_{z_1}(a)\ \forall z \in Z,\ \forall a$, that the viral load of $z_1$is the same if it is in the largest coinfection Z or as a singleton. This means we can write both probabilities as follows

$$\begin{aligned} \Phi_{z_1,z_1,\emptyset}(a) &= f(p_{z_1,}(a)) \\ \Phi_{Z,Z,\emptyset}(a) &= f(p_{z_1,}(a)) \prod_{i \in Z \backslash z_1} f(p_{i,Z}(a)) \qquad \forall a \end{aligned}$$

We can now see that $\Phi_{z_1,z_1,\emptyset} \geq \Phi_{Z,Z,\emptyset}$ since the extra probability terms in the latter cannot be greater than one. We know that the viral load for strains other than z_1 cannot be zero for all days because then the strain would not be a member of the largest coinfection. Thus, so long as the viral loads for the strains $z \in Z \backslash z_1$ are finite then their corresponding probabilities will be less than unity and we have a strict inequality.

$$\frac{\sum_{a=0}^{m} s_{z_1}(a)\,\Phi_{z_1,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} \geq \frac{\sum_{a=0}^{m} s_{z_1}(a)\,\Phi_{z_1,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_{z_1}(a)\,\Phi_{Z,Z,\emptyset}(a)} > \frac{\sum_{a=0}^{m} s_{z_1}(a)\,\Phi_{z_1,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_z(a)\,\Phi_{z_1,z_1,\emptyset}(a)} > 1$$

$$y_{z_1,\emptyset}(0,t+1) \quad > \quad y_{z_1,\emptyset}(0,t) + \frac{\sum_{x \supseteq z_1, x \neq z_1} \sum_{v \subseteq x^c} \sum_{a=0}^{m} y_{x,v}(a,t)\,\Phi_{x,z_1,\emptyset}(a)}{\sum_{a=0}^{m} s_Z(a)\,\Phi_{Z,Z,\emptyset}(a)} \tag{18}$$

Having applied a steady state condition on $r_\emptyset$ we can now see a contradiction. Equation 18 shows us that $y_{z_1,\emptyset}(0,t)$ must be increasing when this steady-state condition holds. This would reduce $r_\emptyset$ and contradict our claim of a steady state. Thus, the largest coinfection must be extinguished at steady state.

4.3.1 Approaching Steady State

The logic given above has made use of the threshold $r_\emptyset = [\beta \sum_{a=0}^{m} s_z(a)\, \Phi_{Z,Z,\emptyset}(a)]^{-1}$. When $r_\emptyset$ is larger than this threshold, there are enough naive hosts to cause the coinfection to grow. It cannot grow to engulf the entire population, because any coinfection creates a total number of new infections which is greater than the new coinfections - by virtue of the fact that coinfections can create singleton infections as well. Thus, increases in the number of coinfections are offset by still larger decreases in the naive population, forcing the naive population to the threshold near steady-state. If $r_\emptyset$ is below the threshold, then the population of the coinfection must necessarily shrink. The only way in which the coinfection can persist is if $r_\emptyset$ is exactly equal to the threshold. The previous section demonstrates why this is not possible. This also precludes any unstable equilibrium which is not zero. If the entire population has a coinfection, some portion of recycled naive hosts will have a single infection, deplete the naive population, and $r_\emptyset$ will eventually reach the threshold, then fall below it.

4.3.2 Any Coinfection Must Become Extinct

In this section we have shown that the largest coinfection, consisting of all strains in Z, must eventually extinguish itself. When this happens, any remaining coinfections will have one less strain than the master strain. However, since these strains will consist of the largest coinfection available to any individuals in the population, their time-series equations will have the same form as the master equation. By inducting over the logic given in this section, all smaller infections will become extinct. This leaves only singleton infections. This inductive argument forces our assumptions to apply to all singletons, and not just some arbitrary z_1.

4.3.3 Coinfections Vanish Under Certain Conditions

We have shown that coinfections vanish under the following two assumptions.

1. Each singleton infection causes an equal or lower per-day mortality rate when compared to any coinfection containing that single strain.

2. The parasite densities for any single strain are not smaller than densities for that strain when it is a member of a coinfection.

This assumes that there is no positive interaction among a coinfection's constituent strains. If such interaction has a *negative* effect on the coinfection - and parasite densities for strains in a coinfection are lower than the parasite densities for the singleton infection - then singletons will grow even faster relative to the coinfections at the threshold, and coinfections will still become extinct. While a positive interaction may make it possible to observe coinfections in this model, it is not likely, given that multiple strains must often compete for the same limited resources in the host.

5 The Superinfection Model

Section 4 presented a multistrain model in which coinfections propagate solely from other coinfections. This excludes the possibility that one host could become infected with two strains from different hosts, a process which we will call *superinfection*. We will include this in the *superinfection model*, named to distinguish it from the simplified *coinfection model*, in which all coinfections are simultaneous. Since we don't understand the dynamic importance of superinfections a priori, we would like to explore it using our model.

The Case For Superinfections. The hypothesis that superinfections occur is based on two arguments. First, coinfections are observed in nature. In fact, between 30% and 80% of all *P. falciparum* infections consist of more than one strain [7]. This is also supported by observations of the parasite's erythrocyte stage in which parasite strains vary the antigens that line the surface of the red blood cell [6]. This provides evidence that coinfections are common in malaria epidemics, leading researchers to create models that not only include multiple strains, but also seek to understand the way in which interactions between strains affect the transmission, diversity, and progression of the disease [16, 20].

The second argument, based on the theory demonstrated in Section 4.3, is that coinfections will become extinct in systems where they can propagate only from other coinfections, where they share the same parasite density curves as the single infections, and where the survivability of a coinfection is no better than its most potent single infection. Since *positive interactions* between strains – in which survivability and transmissibility are enhanced in a coinfection – are less likely than negative interactions, we can rule out the coinfection model. Given this argument, combined with the evidence that coinfections exist in nature, there is a strong justification for relaxing the assumptions of our model to include superinfection.

The Superinfection Model Requires Stochastic Simulation. While it is possible to formulate deterministic equations for the superinfection model which are based upon those for the coinfection model, developed in section 4, solving these equations would be extremely time-consuming because they are significantly more complex. Therefore, we must rely on simulation to produce numerical results.

To understand the computational cost of solving such deterministic equations, consider the fact that the deterministic equations for the simpler coinfection model with 2 strains, each of which has a 30-day course of infection produces $5 \times 30 = 150$ distinct states, corresponding to a coupled differential equation. In contrast, the superinfection model would have at least $30 \times 30 = 900$ different states, each corresponding to the different ages of each strain in the coinfection. This is very computationally costly in practice, but not outside the reach of commonly used methods. However, the computational-time scaling is 30^n for n strains. Rather than deal with this complexity, we have used a stochastic model, via simulations.

5.1 Simulation Methods

A population of hosts where interactions transmit malaria is analogous to a system of reacting particles. The individual hosts behave as particles in which the transmission of a strain from an infected host to a naive host serves as a "reaction" which leaves the former untouched, and converts the latter into a newly-infected host. Using an algorithm, we can employ Monte Carlo simulation methods to project changes in malaria prevalence based on the probabilities of different reactions affecting each individual.

The Gillespie algorithm [14] provides a useful method for conducting these discrete stochastic simulations. The probability of each reaction in an infectious disease is defined by our model. In the model developed in this thesis, the probability of transmission is equal to the probability of infecting a vector, given by the Φ function, scaled by the transmission

rate parameter that incorporates the rate of contacts between people and vectors, and the probability of transmission per contact. Therefore, for each set of parameters, we may define all of the possible "reactions" and their probabilities.

The Gillespie algorithm formalizes these probabilities in an elegant way. Let us call the state of the system $X(t)$ at time t. This is sometimes called the state vector and is defined as $X(t) = (X_1(t), \ldots, X_M(t))$ where there are M different states and $X_i(t)$ is the population of state i at time t. In our system, the states represent populations of hosts carrying a particular set of strains. Each reaction denoted R_j has two parts. The state-change vector is defined as $v_j = (v_{1j}, \ldots, v_{Mj})$ where v_{ij} is the change in i-th state by the j-th reaction. The reaction causes the system to instantaneously change from $x = X(t)$ to $x + v_j$. The second component of the reaction is its *propensity* $a_j(x) = c_j p_j$ which is the the product of the probability of the j-th reaction and the number of combinations of possible reactants. The quantity $c_j dt$ is the probability of an infection event occurring on the time interval $[t, t + dt)$. Since every reaction in an infectious disease system follows the form $infected + naive \rightarrow infected + infected$, we find that h_j is always the product of the naive and infected populations for that reaction.

The Gillespie algorithm proceeds as follows. First, two uniformly-distributed random numbers $r_1$and $r_2$are drawn from the unit interval. Then we define $a_0(x) = \sum a_j(x)$ which is used to calculate the exponential waiting time to the next reaction, $\tau = \frac{1}{a_0(x)} ln(\frac{1}{r_1})$. The second random number is used to select a particular reaction according to their relative propensities. The next reaction is the j-th reaction, where j is the smallest integer satisfying the condition $\sum_{i=1}^{j} a_i(x) > r_2 a_0(x)$.

While this method can accurately simulate our system stochastically, it is somewhat computationally inefficient. For that reason, we have implemented a coarse-graining method called the binomial τ-leaping method, which decreases the computational cost of the simulation while ensuring conservation of population. In this method, larger time-steps are

used, and each reaction "fires" a number of times which is given by a binomial distribution [4]. The simulation results in the following sections were checked against the exact Gillespie algorithm to confirm that the τ-leaping method did not introduce any unusual effects.

The simulation was written in C++ and embedded Python for usability. It is designed to track every host in the system in a literal way. With the firing of each new reaction, the reactant hosts are chosen at random from the population. Because the system allows reactions in which a host with one strain may be infected with another, it effectively simulates the superinfection model.

5.2 Simulation Design

Experiments conducted in mice provide the parasite density curves for coinfections. The experimental data were provided by Silvie Huijben, a member of Professor Andrew Read's group in the Center for Infectious Disease Dynamics at the Pennsylvania State University. Even though it is possible to extend these simulations to include an arbitrary number of strains, only limited by computational resources, we will consider only two strains of the disease, with interactions which are based upon the experimental data. These data describe the parasite densities for one drug-resistant and one drug-susceptible strain, which are primarily distinguished by the parasite density of the second peak of the infection, shown in Figure 3. We have chosen these particular strains in order to eventually develop optimal treatment strategies which extinguish the resistant strain, and hence extend the useful lifetime of each treatment. Treatment strategies are not considered in this thesis, however, the interactions between the two strains in individual hosts may determine the way in which they propagate. Appendix A provides a detailed explanation for how these data were modified to represent the course of the disease in humans.

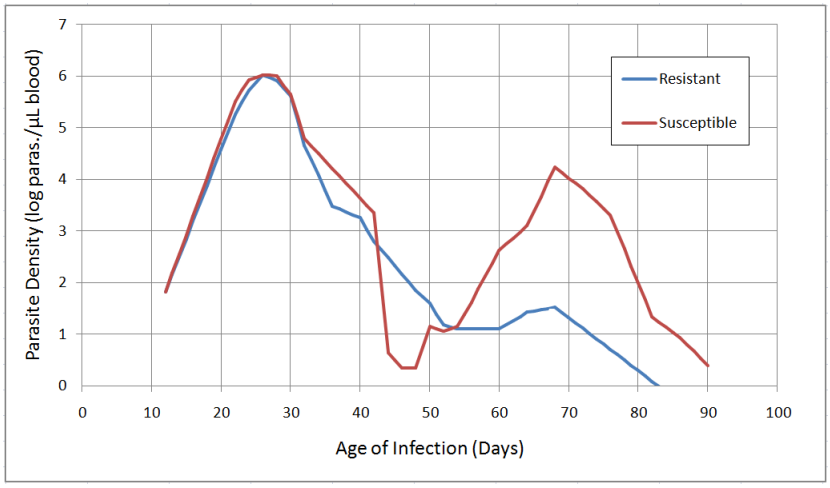

Figure 3: Parasite densities for the resistant and susceptible strains, in single infections. Note that the resistant strain (in blue) is characterized by a smaller second peak parasite density.

Negative Interactions in Superinfections. Instead of allowing coinfections which have the same properties as their respective single-strain infections, we have chosen to make three modifications which will improve the realism of the simulations and create negative interactions between the coinfection strains.

- Superinfection is only possible during the first ten days of an infection. After this point, there is a immunity window which extends to the trough which follows the initial parasite peak, in which we assume the immune system is activated enough to prevent a new infection.

- Superinfections in which the resistant strain infects the host *after* infection by the susceptible one, and also occur during the initial 10-day period, have attenuated parasite densities. Superinfections in which the resistant strain infects the host first will follow the same parasite densities as their respective single-strain infections. We justify this by noting that the resistant strain normally carries a competitive disadvantage, which makes it resistant to drug treatment. This will become important in future studies of

the effects of drug treatment, which are not part of this thesis.

- For the remainder of the infection, superinfection is possible, and the superinfecting strain will obey its single-strain parasite densities.
- There is a 12-day incubation period in which the host is no longer naive, but does not have parasite densities which are high enough to transmit to other hosts. This corresponds to the period in which the parasites must incubate in the liver.

The justification for these changes is provided in Appendix A, which also explains how these data were extracted from experiments involving mice. The effect of these changes is to attenuate the parasite densities of the resistant strain in a coinfection with the susceptible one, compared to a single infection. Additionally, we have introduced an immunity window for the duration of the initial peak parasite densities, which mimics the immune response, during which time a superinfection may be unlikely. These effects are called *negative interactions*, because they serve to decrease the potency of a coinfection compared to an infection with both strains that follow the parasite densities of their respective single-strain infections. This is not only biologically feasible, it also helps us test the parameters under which coinfections may be too weak to sustain themselves. That is, we expect that stronger negative interactions will result in fewer coinfections. In future studies, it would be useful to test interactions further, by varying the immunity window and relaxing the assumptions in this section.

Parameter Choices. Given the experimental data, the simulations have six parameters along with the initial conditions. These include the half-saturation and shape parameters of the transmission-to-vector probability function, the population size N, the transmission parameter β, the natural death rate, and the vector of death probabilities for the course of the infection.

The natural death rate is chosen to be commensurate with the expected lifespan of an individual who lives in a malaria stricken region. For the simulations, I have assumed an average lifespan of 50 years. This is significantly higher than many countries in sub-Saharan Africa, in which malaria is most prevalent; for example, Tanzania has a high incidence of malaria and an expected lifespan of 36 years [26]. However, diseases like malaria contribute significantly to this low life expectancy, and the population of these countries is overwhelmingly skewed towards children. Therefore, it is reasonable to assume that the background life expectancy would be higher.

Second, it is necessary to define the probability of mortality due to malaria. I have assumed that this 6% for both strains of the disease. This is primarily based on a literature review conducted by Alles et. al. which demonstrated that the fatality rate for *P. falciparum* malaria is at least 2% and as high as 20% [1]. I have applied this 6% mortality rate evenly across the 60 days during the course of the 91-day course of infection which correspond to heightened parasite densities, under the assumption that mortality is much more likely to die during this period. This is a bold assumption, and may have affected the overall outcomes of the disease, therefore it may be useful to test it in the future. Though full-scale simulations were not conducted with different natural and disease death rates, changing the background death rate to correspond to an average lifespan anywhere from 25 to 75 years did not change the results at all; increasing disease mortality to 20% did not change the qualitative results.

The transmission rate β and population size N are vital to the dynamics of the model and must therefore must be varied across a wide range in order to understand the behavior of the model. In this case, we only need to vary β because it scales every population term in which new infections are created. The population size must be large enough that random effects do not extinguish the disease in our simulations, but small enough that the simulations run on limited computational resources. For that reason, I have chosen a population size of 50,000,

which roughly mimics a small city or town. I have varied β across a wide range of values in order to understand how transmission affects the overall outcomes of the disease.

Perhaps the greatest unknowns in these simulations are the parameters which translate the parasite densities into probabilities of infection, given in the f-function and hence also the Φ function. It is difficult to estimate these parameters because there is insufficient understanding of the distribution, timing, and exact mechanism of the transfer to mosquitoes to conclude that a single parameter value is appropriate. Since the half-saturation point has a profound effect on the overall shape of the probability curve, I have studied the model across a wide range of half-saturation densities. Figure 4 provides an example of these probability functions, with different half-saturation points. Since high half-saturation values prevent the disease from reaching full probability, relatively higher transmission rates are required to provide an equivalent comparison of "shape". This will be discussed further in the following sections.

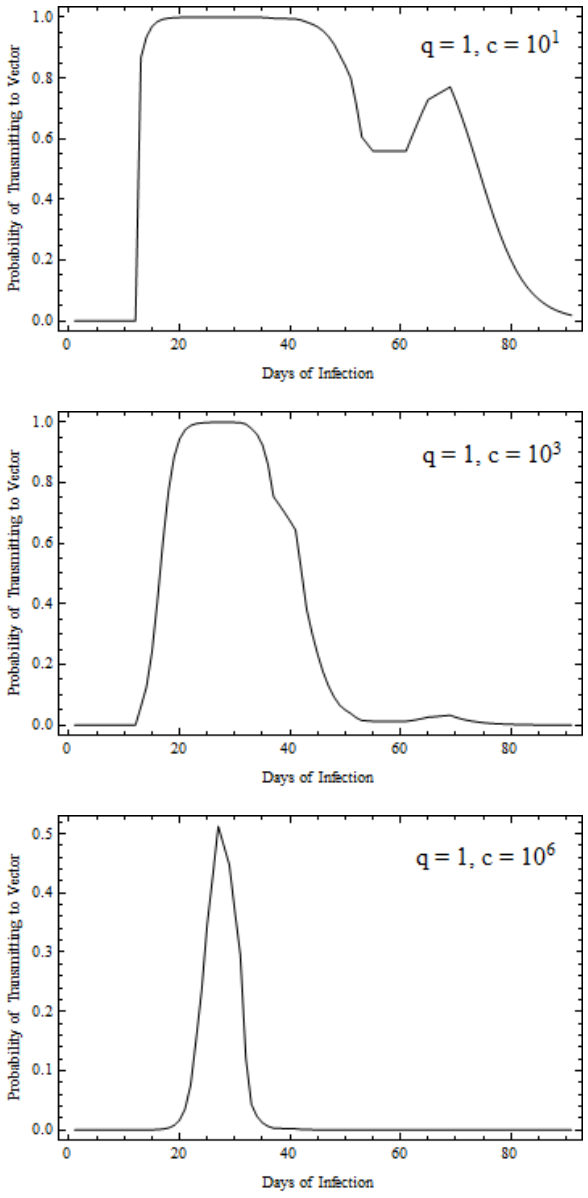

Figure 4: A depiction of the probability of infection $f(p(a))$ with the resistant strain. The half-saturation densities are 10^1, 10^3, and 10^6 parasites per microliter of blood. Notice that at high half-saturation points, the probability is never unity. The shape parameter is $q = 1$.

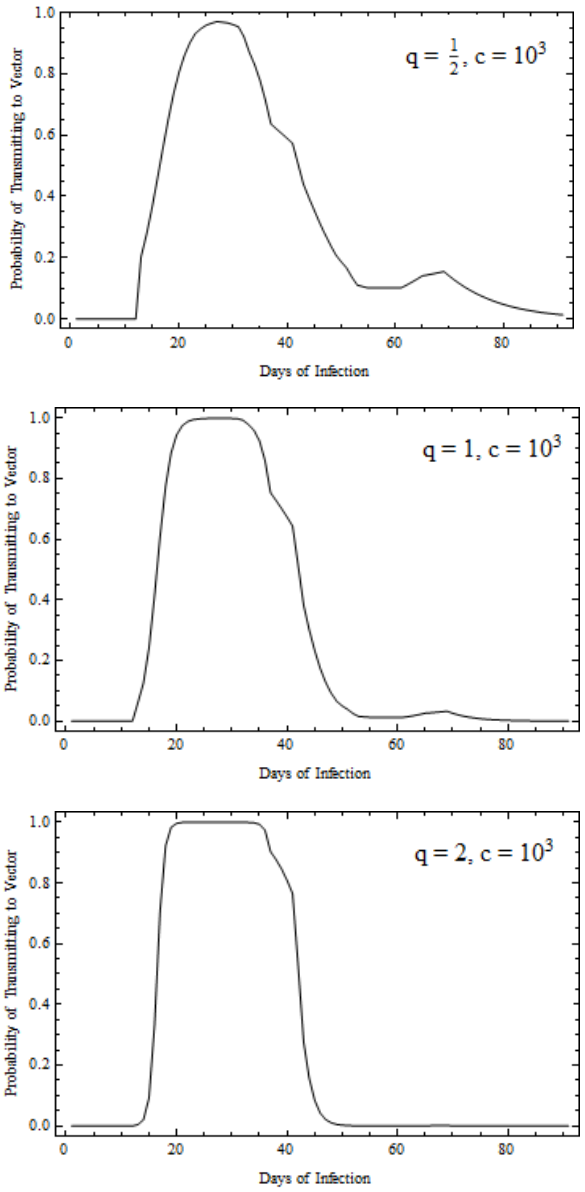

Figure 5: A depiction of the probability of infection $f(p(a))$ with the resistant strain. In this case, we can see that the overall shape changes very little between shape parameters of $\frac{1}{2}$, 1, and 2. The half-saturation density is 10^3 parasites per microliter of blood.

Lastly, we must choose a shape parameter for the transmission-to-vector probability function. I have selected unity, because the resulting parasite density curves do not significantly change shape with higher and lower shape parameters. The shape parameter may have a slight impact on the periodic effects of the infected populations, since it may change the number of days in which a host is maximally infective. However, the half-saturation parameter will have a more dramatic effect on the shape of these probability curves. This is pictured in Figure 5.

Summary of Parameter Choices. The following list summarizes the fixed parameter choices. Widely variable transmission rates and half-saturation densities will be used to evaluate the behavior of the model.

- The background death rate corresponds to an expected lifespan of 50 years.
- The overall disease mortality rate is 6%, over the course of 60 days, which correspond to the highest parasite densities.
- The shape parameter is set to 1, because it does not change the shape of the probability curves as significantly as the half-saturation point.
- Population size of 50,000 because it is large enough to minimize noise in the simulations, and small enough to be computationally efficient.

5.3 Results

Having fixed all but two key parameters: the transmission rate and the half-saturation density, I have analyzed the state of the system after 2,000 days, with two groups of 1,000 hosts infected with each strain at the beginning of each simulation. After an initial survey of the simulation results across a very wide range of parameters, I have restricted my analysis to transmission values between 10^{-6} and 10^{-3} because the infections always become extinct

at lower transmissions and the entire population is infected at higher transmission rates. In fact, we will see that this *infection saturation* occurs in much of this parameter range as well. I have also restricted my study of the half-saturation point to a range between 1 and 10^9 because this covers a large range of probability curve shapes. Moreover, half-saturation below 10^2 ensures that there is very nearly a 100% probability of transmission to the mosquito on each day of infection, and half-saturation densities which are higher than 10^9 result in extinction. This is because they reduce the maximum probability of transmission.

Each simulation lasted at least 2,000 days or about 5.5 years. This was almost always sufficient for the system to converge to a state independent of the initial conditions. Figure 6 shows the proportion of hosts which are infected at equilibrium and Figure 7 depicts the amount coinfections in a density plot. The distinguishing feature of these plots is the transition from unsustainable transmission to a population that is entirely infected with coinfections. This happens in both directions, as transmission increases, and as the half-saturation density decreases, since lower half-saturation lowers the maximum of the probability curves. If $\beta < 10^{-6}$, then transmission is not self-sustaining even with full probability of transmitting to a vector (that is, $f(x) = 1$). In this case, the disease becomes extinct. The ceiling on the half-saturation density (c) is weaker. We may say that $\forall c$, $\exists \beta*$ such that $\beta > \beta*(c)$ implies that transmission is sustained. We leave an exploration of the shape of $\beta*(c)$ as $c \to \infty$ as an open problem which merits further attention, because it will describe the half-saturation density ceiling above which self-sustained transmission is possible.

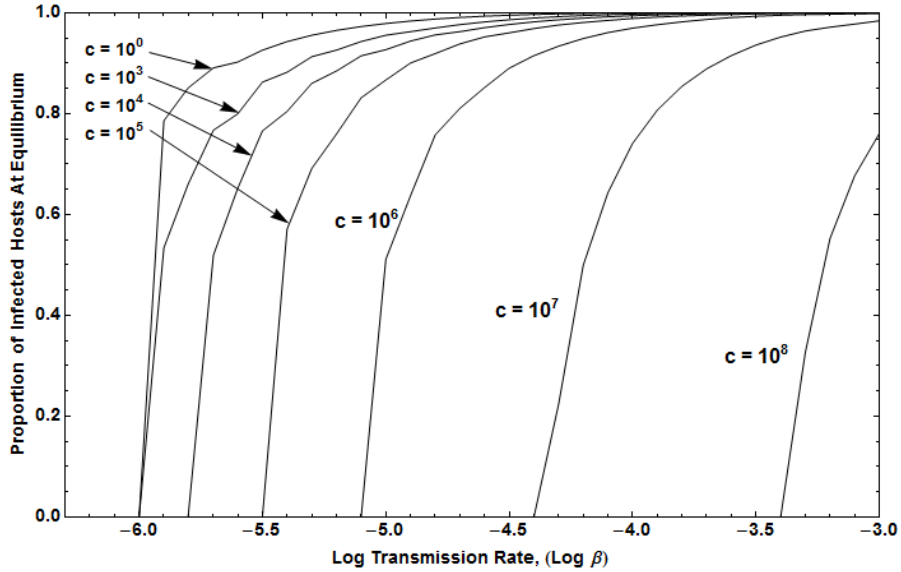

Figure 6: This plot shows the proportion of infected hosts, including both single infections and coinfections. Each curve represents a different half-saturation point, with lower half-saturation points producing higher proportions of infected hosts. The populations were averaged across the final 500 days of simulation to minimize noise.

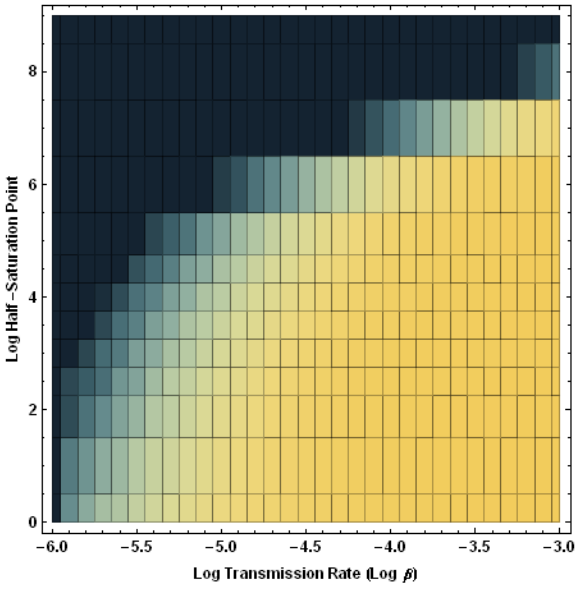

Figure 7: This density plot is shaded according to the number of coinfections. Dark blue represents the absence of infections, while bright yellow parts of the plot represent a high – in this case saturated – population of coinfected hosts, which have crowded out any single infections.

These plots highlight several distinct features of the model. At low transmission rates and high half-saturation densities, there are no infections present at equilibrium. A system with low transmission may prevent one infection from creating enough secondary infections to sustain itself, while a system with high half-saturation produces a ceiling, depicted in Figure 4, on the maximal probability of transmission to the vector, which has the same effect as reducing transmission.

Figure 8 shows the results from four different time-series of the superinfection model simulation, each with a different overall outcome. These were all taken from the same half-saturation density of 10^4 so that they can easily be associated with a single row in Figure 9, which plots the outcomes across the parameter space of different transmission rates and half-saturation densities.

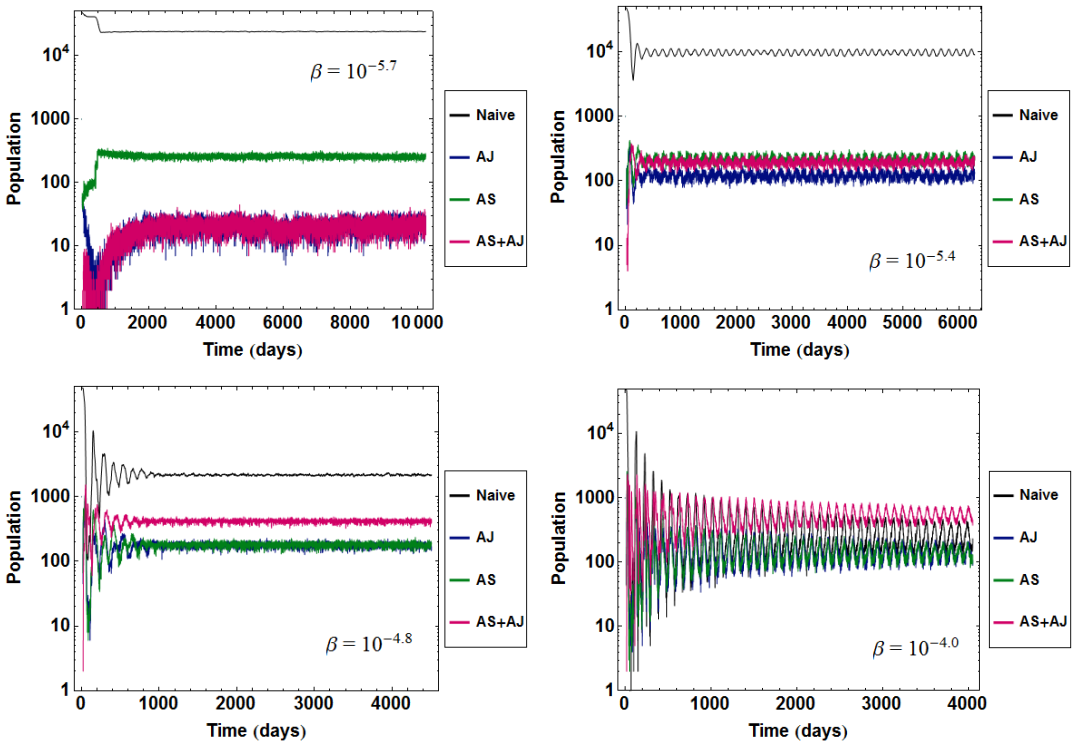

Figure 8: Time series for four representative simulations. These correspond to different transmission rates along a single horizontal row ($c = 4.0$) on the parameter space map shown in Figure 9. Lower populations show noise which is deceptively large due to the logarithmic plots. Note that for simplicity, I have plotted only the 0-day-old single-strain infections and coinfections where at least one of the infections has begun within 1 day. For this reason, the total number of infected hosts for a particular strain is often higher than the number of naive hosts, even though it appears lower on the plot. Coinfections are depicted by the curve "AS+AJ".

Figure 9 shows a color-coded map across the two-dimensional parameter space, in which each color corresponds to a different ranking of infected populations. Aside from the region in which both strains become extinct, the plot has two important features. Most of the parameter space is consumed by three large regions. In the upper-left corner lies the "extinct zone" in which no infections are possible. In the center and lower-right part of the plot is the "high-transmissibility" zone in which β is large or the half-saturation point is low, meaning that the probability of transmission to the vector is high – this somewhat opposes the drop

in transmissibility when the half-saturation point is high, and the probability of transmitting to a vector is never zero. In the high-transmissibility zone, coinfections dominate, but it is possible for either the resistant (denoted AJ) or susceptible (denoted AS) strain to have the second-largest infected population. In the middle of the plot lies the roughly crescent-shaped "intermediate zone" in which there is disease at equilibrium, but overall transmissibility is not high enough to cause coinfections to dominate. Instead, the susceptible strain (AS) has the largest infected population. Figure 8 provides samples of the time series for each major type of equilibrium denoted in Figure 9, across a single half-saturation density, pictured horizontally on the parameter space plot.

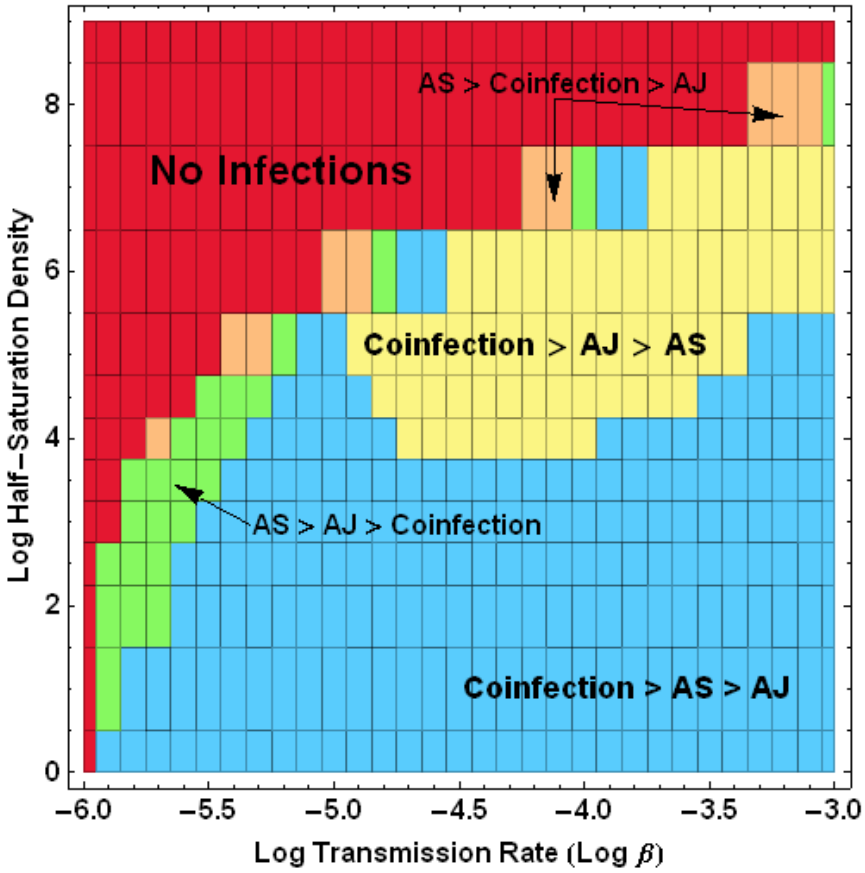

Figure 9: A color-coded representation of the largest to smallest infected populations at equilibrium. In the blue and yellow regions (the high transmissibility zone), coinfections dominate both single strains. Along the roughly crescent-shaped boundary between the disease-free state and the diseased states, there are some simulations in which coinfections were not dominant (this is the intermediate zone).

When classifying the equilibria of the stochastic superinfection system, we are primarily concerned with the relative populations of each strain combination and the types of oscillation and noise. The plots in this section have demonstrated that there are three zones in the transmissibility and half-saturation parameter space: an extinct zone, a zone of high transmissibility in which coinfections dominate, and an intermediate zone in which the susceptible strain dominates the coinfections. The noise was generally constant, and relatively

proportional to the population size. Pronounced oscillations appeared in several places, but were more likely to be found in areas with higher transmissibility. The oscillation period was invariant, and was always roughly 90 days, the duration of the disease.

6 Conclusions and Future Work

In the preceding sections, we have developed a theory for the spread of multiple strains of malaria based on malaria's within-host dynamics. In our theory, the parasite density for an infection of a particular age – given by experimental data – determines the probability of transmitting the strains from a human host to a mosquito, according to transmission-to-vector function. This function is characterized by the half-saturation density, which is the parasite density at which there is a 50% chance of transmission to a vector during a single interaction. In concert with the transmission probability, I have shown that these parameters have a profound effect on the steady-state outcomes of the disease. Specifically, it is clear that coinfections dominate the population in regions with high transmission. However, the crescent-shaped boundary (pictured in Figure 9) between regions of this parameter space with no infection, and those with dominant coinfections also shows areas where the susceptible strain dominates the population, but coinfections still exist in smaller quantities. These results suggest that superinfection can explain the maintenance of coinfections in a population, regardless of the parameters, since coinfections were present in almost all of the diseased equilibria. Moreover, the clear distinctions between disease-free, single-strain dominant, and coinfection-dominant parts of our parameter space also indicates that that the shape of the transmission-to-vector probability curve influences the equilibrium outcomes of the disease.

These results represent only a fraction of the power of this model and there are many opportunities test alternative hypotheses. For example, it would be useful to run simulations to determine the transmission and half-saturation parameters under which the resistant strain is able to "invade" or reach significant prevalence in a population that has already reached equilibrium with the susceptible strain. It would also be useful to create even stronger negative interactions between the strains in a coinfection in order to determine just how strong coinfections can be, since it is possible that coinfections exist in our model

thanks to the relatively weak competitive exclusion between strains. If we compare these results to simulations with no coinfections, we may also find that coinfections provide an advantage which allows the weaker strain to persist under low transmission rates and heavy competition from another strain, where it might not otherwise survive. Finally, we have access to experiments which describe the attenuation of both resistant and susceptible strains under drug treatments. In future research we will use these to study the effects of treatment on the survivability of both strains, in order to design optimal treatment strategies.

Even in comparison to more detailed models, this model impacts future efforts in studying infectious disease. By showing that the shape of the probability curve may have a profound effect on both the persistence of malaria and the relative proportion of coinfections, we have demonstrated the importance of within-host dynamics. These results also suggest an important role for superinfection in supporting the persistence of multiple-strain infections. We may conclude that future efforts to characterize and control malaria – and also other infectious diseases – require the consideration of both superinfections with multiple strains and more generally, the effect of the within-host dynamics on the transmission of the disease.

A Appendix: Parasite Densities

In order to analyze the effectiveness of this malaria model, we must first extract reasonable parasite density curves from the experimental data. This section will summarize the way in which the data were modified. We have used two sets of experimental data, both of which were provided by Silvie Huijben, a member of Professor Andrew Read's group in the Center for Infectious Disease Dynamics at the Pennsylvania State University. Professor Andrew Read is Professor of Biology and Entomology. The experiments have not yet been published.

A.1 Parasite Densities are Given By Experiments in Mice

The first set of data consists of a series of experiments which tracked parasite densities for mice infected with rodent malaria. Two clones of the rodent malaria *Plasmodium chabaudi* were used: a treatment-resistant strain (AJ) and a treatment-susceptible strain (AS). Since Malaria has a two-stage reproductive cycle, the experimental data contain measurements of the number of both asexual and sexual parasites per microliter of blood. While the sexual stage is responsible for transmission to mosquitoes, the asexual parasite densities are higher, and thus less noisy. Since the shape of both asexual and sexual parasites is roughly the same, we have opted to use the asexual parasites. The probability of infection is influenced by the f-function, which has flexible half-saturation and shape parameters described in Section 3.2. Since these may be tuned to adjust the way in which the parasite densities are translated into probabilities, using the asexual parasite counts will not affect the role of the parasite densities in our model.

The first experiment provides the density data for single infections and coinfections, but not superinfections. The original experiment data for single infections are shown in Figure 10 whereas coinfections are depicted in Figure 11. For superinfections, we have turned to another set of experiments.

The second set of experiments was designed to describe the parasite densities in a coinfection (AS+AJ) in which the initial inoculum of the resistant clone AJ was lower than the inoculum of AS. This gave the AS clone a competitive advantage, which led to attenuated levels of the AJ clone. The data are plotted along with the interpolated data in Figure 14. We will use these data to infer the parasite densities for a coinfection in which there is a several-days delay before the subject is inoculated with the AJ clone, since a time-delay between infections is physically similar to observing an infection with a lower inoculum of one strain.

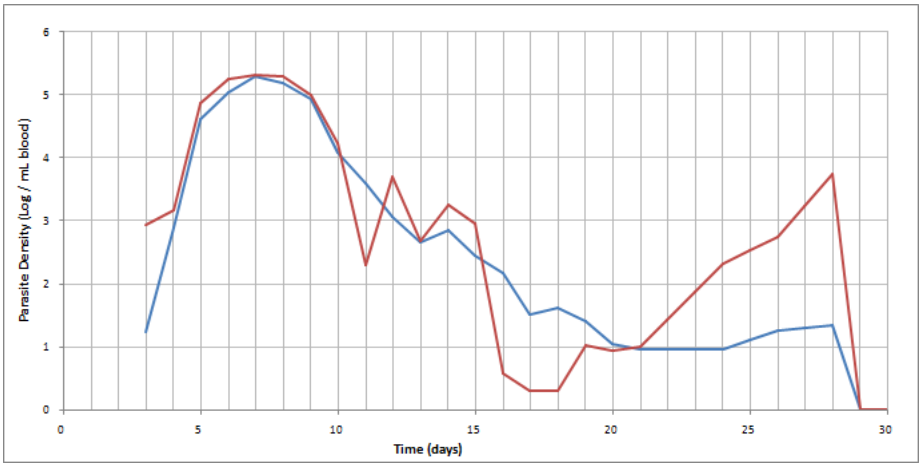

Figure 10: Experimental parasite densities for single infections with the AJ clone (blue) or the AS clone (red) in mice.

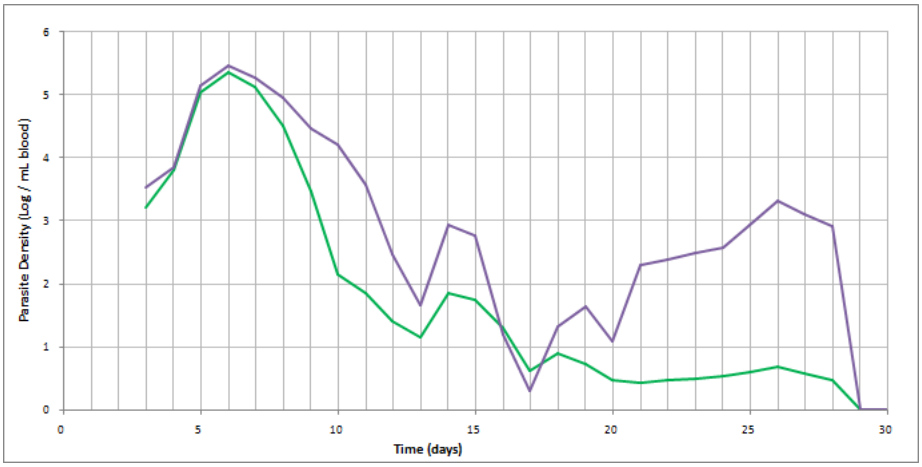

Figure 11: Experimental parasite densities for AJ (green) and AS (purple) in a simultaneous coinfection in mice.

A.2 Differences Between Malaria in Mice and Humans

In order to use the parasite density data from experiments in rodent malaria in a model for human malaria, it is necessary to translate these data so that they coincide with known features of human malaria. First, rodent malaria parasites reproduce on a 24-hour cycle, while human malaria - *Plasmodium falciparum* - reproduces on a 48-hour cycle [13]. Since the difference is an integer multiple of 24 hours, we simply scale the experimental data by a factor of 2 with respect to time. Thus, each day of the simulation on a rodent malaria time scale will correspond to 2 days of human malaria. In the following plots, we have pictured the parasite densities with the human time scale.

Second, human malaria is asymptomatic for the first 6 to 14 days, with a median of 11 days [41]. This period may roughly correspond to the incubation time in the human liver before the parasites enter the bloodstream, at which point the disease becomes transmissible. In the experiments, the parasite densities were not measured until the third day of the mice experiments. To replace the missing data, we have extrapolated the data to approximately $10^{1.8}$ parasites per microliter of blood at the beginning of the experiment, using the first

available slope. Since we expect the liver to inoculate the blood with roughly the same amount of parasites regardless of the type of infection, we will force each infection to match this density for the day corresponding to the start of the experiment. Even though this will produce slightly different initial growth rates for some of the curves, the difference will not noticeably change our model.

Without experimental data to describe the amount of initial bloodstream inoculum or the exact time of incubation in the liver, we must infer these parameters. Let us assume that symptoms arise at the midpoint of the reported 8-30 day delay, or 19 days, which we round to 18 so that we can evenly convert it into mouse-malaria time. Then let us assume that these symptoms arise at roughly half the number of logarithms between the initial inoculum and the peak. This would correspond to a 12-day human incubation time in humans, which is pictured in Figure 14. It is possible that the initial bloodstream inoculum is lower than $10^{1.8}$ parasites per microliter, however at such low densities, this will have almost no effect on transmission. Nevertheless, this solution meets the requirements of our model by preserving the shape of the parasite density curves, and roughly aligning the time until symptoms arise with the steepest part of the initial growth curve.

Third, it is necessary to consider the limits of our experimental data. We expect the parasite density to vary among each host. Moreover, there many be differences among rodent and human malaria parasites, despite their similar physiology and mechanisms. Therefore, in order to capture the general shape of the parasite density curves in our simulation, without amplifying the effects of any noise from our experimental data, we have smoothed - via interpolation - any data points which show one-day peaks or troughs. We have also scaled the AJ data slightly, so that they peak at the same parasite density, and hence have the same maximum transmission probability. This preserves the natural shape of the curves while making them more general, and hence applicable to a large number of simulated hosts.

A.3 Description of Parasite Densities for Human Malaria Simulations

The data which represent the parasite densities for human malaria are summarized in the following section. We will consider four cases: single strains, simultaneous coinfections, and superinfections with delayed inoculation of either the AS or AJ clone.

Single-Strain Infections. Data from the first set of experiments are summarized below, including the parasite densities of the single-strain infections and equal-inoculum coinfections. Even though we have access to similar data from the second set of experiments, these experiments did not include the parasite density curves for AJ alone. To maintain consistency, we are using the first set of experiments to define both single-strain and coinfection curves, leaving the second set of experiments to describe only the superinfections. Both single-strain courses-of-infection are pictured in Figure A.3. There is a 12-day incubation period in the liver, after which time the parasites appear in the bloodstream.

To maintain consistency among single-strain infections and superinfections, we have used the same 90-day course-of-infection as the second set of experiments, which involved a longer duration of data collection to include the second peak of parasites in the susceptible clone. In addition to smoothing one-day noise by interpolation, we have extrapolated the tail of the AS curve to complete the second peak. The second set of experiments provides a curve for AS which extends to 90 days on the 24-hour cycle. These data have been appended to the tail of this curve, and scaled to the height of the final measurement on the shorter curve, to avoid discontinuity. This allows the second peak, a major feature of the single AS clone, to be included in the curve.

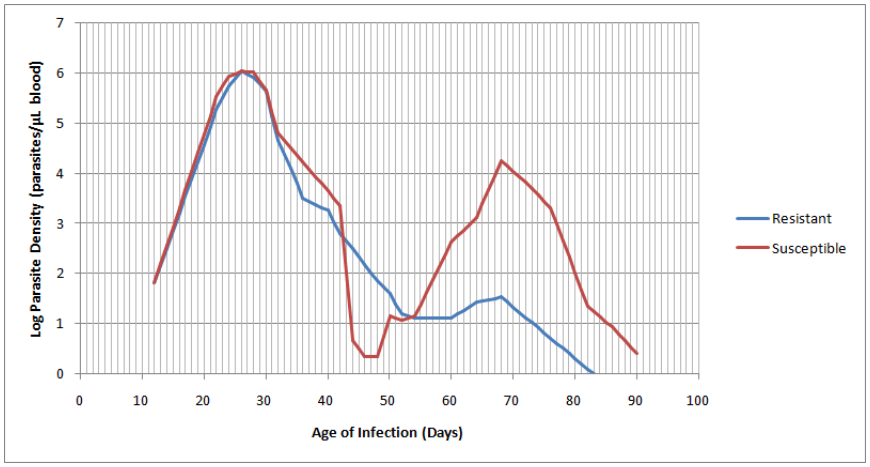

Figure 12: Parasite densities for the resistant strain (blue) and susceptible strain (red) in single infections.

Simultaneous Coinfection. The parasite densities throughout the course-of-infection for coinfections which begin on the same day - whether they are transmitted by a single host, or different hosts - will adhere to the experimental data in which the mice were inoculated with the same number of parasites. The parasite densities are pictured in Figure 13, which shows an extra peak immediately after the initial peak. These curves were subject to the same procedures mentioned above. That is, they were extended to match the extra data provided by the second set of experiments without discontinuity, they were extrapolated to the universal bloodstream inoculum density, and one-day noise was removed with interpolation.

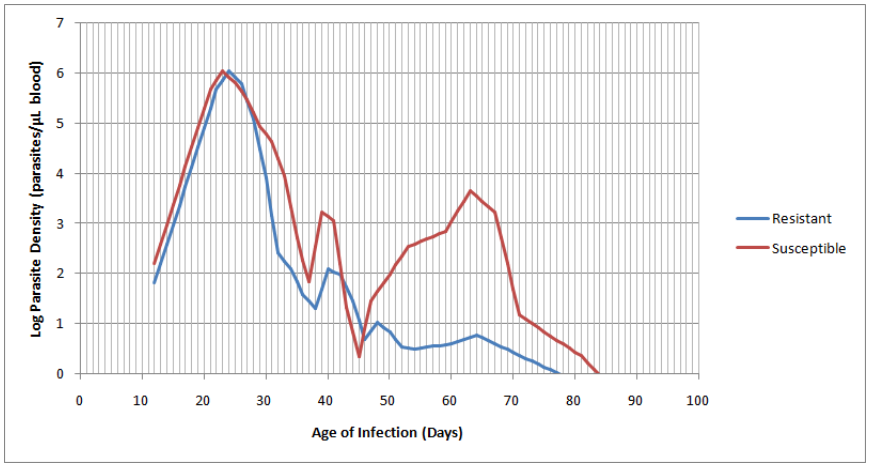

Figure 13: This chart describes the parasite densities for simultaneous coinfection, which includes the susceptible, AJ clone (red) and the resistant, AS clone (blue). The coinfection curves show an intermediate peak that is not present for the single infections.

Resistant Strain Infects a Host With the Susceptible Strain. When a resistant strain infects a host which already has the susceptible strain, the immune response and competition with the susceptible strain will attenuate the resistant infection. It cannot realize the same densities as a single resistant infection. In this case, we will use coinfection experiments with a lower resistant strain inoculum to approximate a delayed infection. This method is supported by evidence that the greater the time between an AS and subsequent AJ infection, the weaker the AJ infection will be [9].

To make this approximation, however, we must assume that both infections have the same density of parasites at some point in their respective courses-of-infection. We assume that this occurs at the initial inoculation of the bloodstream, from parasites which were incubated in the liver. This allows us to interpolate between parasite densities which correspond to an attenuated initial inoculum of the AJ clone so that each delayed infection with AJ still starts at $10^{1.8}$ parasites per microliter. This effectively describes a delayed infection while agreeing with our assumptions about the incubation period and bloodstream inoculum. The data are summarized in Figure 14. As with all of our data, we have also

removed any one-day noise with interpolation.

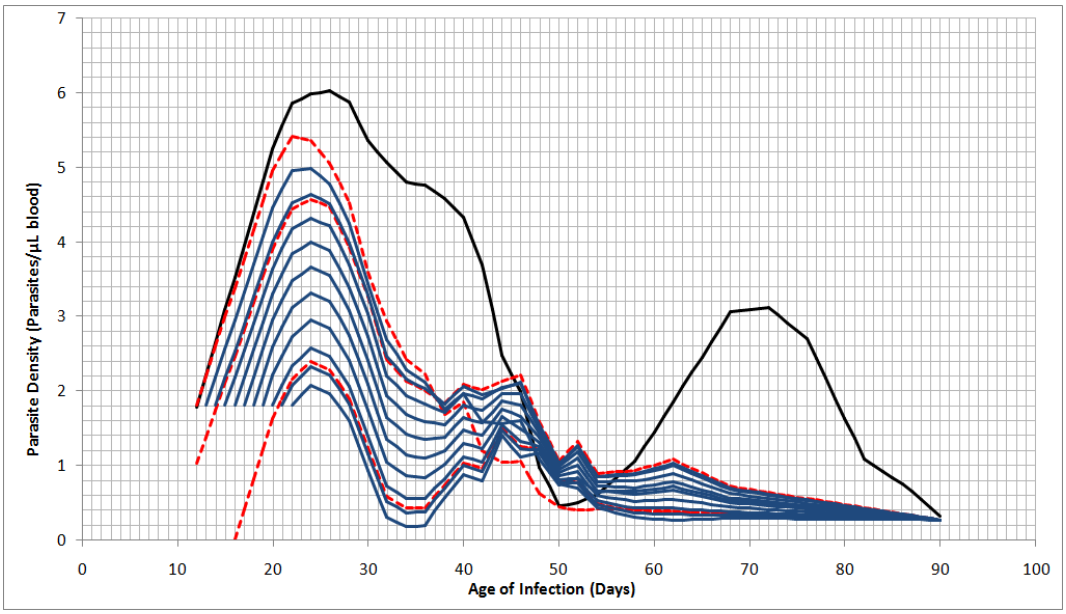

Figure 14: This chart describes the parasite densities for superinfections with a resistant, AS infection (blue) following a resistant AJ infection (black). The AJ data were interpolated from the experimental data (dashed, red), in which the AJ inoculum was smaller than the AS inoculum.

Susceptible Strain Infects a Host With the Resistant Strain. We have no experimental data regarding the reverse situation, in which a susceptible infection follows a resistant one. While there is evidence that a susceptible infection which follows a resistant one will be attenuated, [9] we also know that there are often biological costs associated with resistance to drug treatments. Therefore, the susceptible strain will be attenuated less than the resistant strain in the reverse case. It is reasonable, then, to assume that susceptible infection may not be attenuated, and follows the same course-of-infection as its single infection, given in Section A.3. Without more experimental data, we cannot relax this assumption. It is useful, however, because it is the simplest assumption for the dynamics of a superinfection with a delayed susceptible strain.

B Appendix: Simulation Code

The following sections outline the simulation code which was used in this thesis. The C++ code was compiled using GNU C++ version 3.4.6, and embedded a Python package called "mis" (short of Malaria Infection Simulator), which was compiled with Python 2.6. The library links against the GNU Scientific Library (GSL) in order to generate random numbers by a binomial distribution. At the time this thesis was printed, the code was still under development in order to include an arbitrary number of strains. As such, some of the features of the program will accomodate many strains, while others are only designed for the two-strain model studied in this thesis. The code employs the binomial τ-leaping method to simulate the model, and randomly selects recipient hosts of the desired type whenever a reaction occurs.

The simulation code uses a binary system to represent the state of the host. For the 2-strain model, I have used a 4-bit number. The two leftmost bits correspond to the strains which the host has recovered from, while the remaining two bits represent the active infections. For example, if a host is recovered from the first strain, but is infected with the second strain, then the host state will be $0110 = 6$ (we read the digits in each half of our 4-bit number from right to left). Some states are not possible, such as $0101 = 5$ which implies that the host is infected with the first strain (the resistant strain), but is also recovered from it, which is a contradiction because I have assumed permanent immunity. In order to represent the days of both infections in a single number, I have also employed the following formula for encoding two days of infection x, and y in a single number $k(x, y) = x + D \times y$ where D is the duration of the infection. This is used throughout the code.

The code requires 3 data files: a single/coinfection file, a superinfection file, and a survivability file. The values are not printed with this thesis because they rely on experimental data which have not yet been published. If our infection lasts D days, then the required single/coinfection tab-delimited text file consists of a $6 \times D$ matrix of log (base 10) parasite

densities, where the first two rows correspond to the parasite densities for each strain in a single infection with one strain (one row will be blank), the next two represent a single infection with the other strain (the alternate strain will be blank), and the final two rows represent both strains in a coinfection.

The superinfection file consists of a $(2 \times D) \times D$ matrix which captures only the parasite loads for delayed infections in which strain 2 (the susceptible one) infects the host first. Recall that when the resistant strain infects the host first, we have assumed that the parasite densities will be the same as a coinfection. The i-th and $D + i$-th rows of this matrix correspond to the parasite loads for a delayed resistant strain superinfection which is delayed by i days. Lastly, the survivability file contains a $2 \times D$ matrix holding the probability that a host will survive the infection on the day corresponding to the column, where the first row corresponds to the resistant strain, and the second corresponds to the susceptible one. These survival probabilities were the same for both strains in this thesis.

B.1 Python Script: "script.py"

```
import mis
import time

def init():
        print "Loading data."
        mis.load_data(filename='asex_noootreatment_55.txt',
                super_filename='asex_noootreatment_super21.txt')
        mis.load_survivability_data('survivability.txt')

def run():
        ## Name Output Files
        mis.out_names(pops="expt_4_%012.8f" % script_B,
                report="expt_4_REPORT.dat")
        ## Pass Parameters
        print "Passing parameters."
        mis.model_parameters(B=script_B, N=script_N,
                phi_cutoff=script_phi_cutoff,
                phi_power=script_phi_power,
                disease_death_rate=script_disease_death_rate,
                natural_death_rate=script_natural_death_rate,
                super_infection=1,print_rates=0)
        ## Prepare Simulation
        print "Generating reaction probabilities."
        start = time.clock()
```

```
        mis.generate_probabilities()
        end = time.clock()
        mis.initialize_counters()
        ## Seed Infection
        print "Seeding infection."
        for i in range(0,3):
                if script_seeds[i] != 0:
                        mis.seed_infection(1,script_seeds[i])
        ## Run Simulation
        print "Running simulation."
        start = time.clock()
        for i in range(1,20000):
                mis.step_BD(100)
                #if ((i%1000)==0):
                #       print "Internal counter is: ", i, "."
        end = time.clock()
        ## Cleanup
        mis.reset_counters()

## Fixed Parameters
script_phi_cutoff=10**4
script_phi_power=1
script_disease_death_rate=0
script_natural_death_rate=0.000055
script_seeds = [0, 0, 5000]
script_N = 50000

init()
betas = [0.00001]
for script_B in betas:
        run()
```

B.2 Definitions and Interface with Python: "mis.cpp"

```
#include <Python.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <math.h>
#include "gsl/gsl_rng.h"
#include "gsl/gsl_randist.h"
using namespace std;

// VIRAL DENSITY VARIABLES
vector<float> * data;           // Single-Strain Infection Data
vector<float> * super21_1;      // Superinfection Data, strain 1, then 2
vector<float> * super21_2;      // Superinfection Data, strain 2, then 1
vector<float> * survivability;  // Daily disease survival probs.
int COI;                        // The duration of the infection
int kernels;                    // Lines in "data" (for 2 strains: 6)

// SIMULATION PARAMETERS
int N = -1;                     // Population size
float t;                        // Time (days)
```

```
int day;                          // The next integer day.
int rxns;                         // Total possible reactions
int step_num;                     // Step counter
int duration;                     // Total number of steps to simulate
float time_step;                  // Time until next reaction

// MODEL PARAMETERS
float B = -1;                     // Transmission parameter
float phi_cutoff = -1;            // Half-saturation density
float phi_power = -1;             // Shape parameter
float disease_death_rate = -1;    // Unused (see survivability)
float natural_death_rate = -1;    // Background death rate
int SUPER = -1;

// COUNTERS
int M;                            // Number of strains (fixed at 2)
int *toc;                         // Binary lookup table, valid strains
int *toc_coinfect;                // Binary lookup table, number of strains
int L;                            // Index for days in recipient
int K;                            // Index for days in transmitting host
int J;                            // Binary state index in recipient
int I;                            // Binary state index in transmitting host
int H;                            // Index of strain set (1-3)
int Q;                            // 2^M
int** pop;                        // Array of population counts, detailed
int* popC;                        // Total population counts for each state
int** states;                     // For nonrandom selection of recipients
double a0;                        // Sum of propensities
int* rates;                       // Used to observe rates of infection

// REACTION TABLES
float* tableP;  // Reaction probabilities
float* tableM;  // Master reaction table
int* tableRi;   // Transmitter state, particular reaction
int* tableRj;   // Receiver state, particular reaction
int* tableRk;   // Transmitter encoded days, particular reaction
int* tableRl;   // Receiver encoded days, particular reaction
int* tableRh;   // Transmitted strain for a particular reaction
int A;          // Length of tableM, fixed
float AA;       // Number of rxn with nonzero propensities, updated
int* ktable;    // Used for decoding k with different "day" indices
int* ltable;    // Used for decoding l with different "day" indices
int* ktableref; // Used for decoding k, l with different "day" indices
int* ktableref2;// Used for decoding k, l with different "day" indices

// Output Files
ofstream fp_pop;        // Outputs population counts
ofstream fp_report;     // Debugging tool
int print_rates = 0;    // Set to zero to suppress output of rates

#include "host.h"

gsl_rng * random_number;        // Used to generate random number
host* pool;                     // Array of host objects

void population_output(int increment);
void update_master();

#include "init.h"
#include "step.h"
```

```
#include "output.h"

static PyMethodDef mis_methods_py[] = {
        { "load_data", (PyCFunction)load_data, METH_VARARGS |
                METH_KEYWORDS, "Loads kernel data."},
        { "load_survivability_data", load_survivability_data,
                METH_VARARGS, "Loads survivability data from
                specified file."},
        { "load_immune_response_data", load_immune_response_data,
                METH_VARARGS, "Loads immune response curves."},
        { "model_parameters", (PyCFunction)model_parameters,
                METH_VARARGS | METH_KEYWORDS, "Sets the model parameters."},
        { "out_names", (PyCFunction)out_names, METH_VARARGS |
                METH_KEYWORDS, "Specify output names for a
                particular experiment."},
        { "generate_probabilities", generate_probabilities,
                METH_VARARGS, "Generate the reaction probability table."},
        { "initialize_counters", initialize_counters, METH_VARARGS,
                "Creates counting tables."},
        { "reset_counters", reset_counters, METH_VARARGS,
                "Resets counting tables for another experiment."},
        { "seed_infection", seed_infection, METH_VARARGS,
                "Start the infection with a particular strain."},
        { "step", step, METH_VARARGS, "Orginal stepping method."},
        { "step_BD", step_BD, METH_VARARGS, "Binomial tau-leap
                stepping method."},
        { NULL, NULL, 0, NULL}
};

PyMODINIT_FUNC initmis(void)
{
    (void) Py_InitModule("mis", mis_methods_py);
}
```

B.3 The Initialization Header File: “init.h”

```
// ************************************************************
// Loads the Parasite Density Data and initializes variables

static PyObject*
load_data(PyObject *self, PyObject *args, PyObject *keywds)
{
        const char* filename;
        const char* super_filename;
        static char *kwlist[] = {"filename","super_filename",NULL};
        if (!PyArg_ParseTupleAndKeywords(args, keywds,"|ss", kwlist,
                &filename, &super_filename)) return NULL;
        fp_report << "\tLoading data from " << filename << ".\n";
        fp_report << "\tLoading super21 data from " << super_filename
                << ".\n";

        // First count the lines in the file (6 for the 2 strain model)
        ifstream fp_in_count(filename, ios::in);
        string line;
        int k = 0;
```

```
while (getline(fp_in_count,line)) {
        kernels++;
}
fp_in_count.close();
ifstream fp_in_count2(filename, ios::in);
getline(fp_in_count2,line);
while(line[k] != '\0') {
        if(line[k] == '\t') { COI++; }
        k++;
}
COI++;
fp_in_count2.close();

// Calculate limits for most parameters
M = 0;
while (2*(pow(2,M)-1) != kernels)
        M++;
fp_report << "\tThe viral load data allow for " << M
        << " strains in this system.\n";
fp_report << "\tThis gives " << I
        << " states (some impossible).\n";
I = (int)pow(pow(2,M),2);
K = (int)pow(COI+1,M);
J = I;
Q = (int)pow(2,M);
H = (int)(pow(2,M)-1);

// Initialize the population counters
pop = new int* [I];
for (int i=0; i < I; i++)
        pop[i] = new int [(COI+1)*(COI+1)];
popC = new int [I];
rates = new int[I*H];
for (int l=0; l < I*H; l++)
        rates[l] = 0;

// Read the parasite density data
ifstream fp_in(filename, ios::in);
data = new vector<float>[COI];
float in_value = -1;
for (int i = 0; i < kernels; i++)
{
        for(int j = 0; j < COI; j++)
        {
                fp_in >> in_value;
                data[i].push_back(in_value);
        }
}
fp_in.close();

// Define random number variables
const gsl_rng_type* random_type;
gsl_rng_env_setup();
random_type = gsl_rng_default;
random_number = gsl_rng_alloc(random_type);

// Create the tables used for encoding 2 age of infection
// indices in 1 number
ktable = new int [M];
ltable = new int [M];
```

```
        for (int m=0; m < M; m++) {
                ktable[m] = -1;
                ltable[m] = -1;
        }
        ktableref = new int [M+1];
        ktableref[0] = 1;
        for (int m=0; m < M; m++)
                ktableref[m+1] = COI+1;
        ktableref2 = new int [M+1];
        ktableref2[0] = 1;
        for (int m=0; m < M; m++)
                ktableref2[m+1] = ktableref[m+1]*ktableref2[m];
        fp_report << "\tThis is the k-value encoding table:\n\t\t";
        for (int m=0; m < M+1; m++)
                fp_report << ktableref2[m] << " ";
        fp_report << endl;

        // Load superinfection data from the necessary files
        if (super_filename != "") {
                super21_1 = new vector<float>[COI];
                super21_2 = new vector<float>[COI];
                ifstream fp_in_super(super_filename, ios::in);
                for (int i = 0; i < COI; i++)
                {
                        for(int j = 0; j < COI; j++)
                        {
                                fp_in_super >> in_value;
                                super21_1[i].push_back(in_value);
                        }
                }
                for (int i = 0; i < COI; i++)
                {
                        for(int j = 0; j < COI; j++)
                        {
                                fp_in_super >> in_value;
                                super21_2[i].push_back(in_value);
                        }
                }
                fp_in_super.close();
        }

        return Py_BuildValue("");
}

// ************************************************************
// Loads the Survivability Data

static PyObject*
load_survivability_data(PyObject *self, PyObject *args)
{
        const char* filename;
        if (!PyArg_ParseTuple(args, "s",&filename)) return NULL;
        fp_report << "\tLoading survivability data from "
                << filename << ".\n";
        ifstream fp_in(filename, ios::in);
        survivability = new vector<float>[M];
        float in_value;
        for (int m = 0; m < M; m++)
        {
                for(int k = 0; k < COI; k++)
```

```
                {
                        fp_in >> in_value;
                        survivability[m].push_back(in_value);
                }
        }
        fp_in.close();
        return Py_BuildValue("");
}

// ************************************************************
// Receives model parameters from Python script

static PyObject*
model_parameters(PyObject *self, PyObject *args, PyObject *keywds)
{
        float temp_B = B;
        int temp_N = N;
        float temp_phi_cutoff = phi_cutoff;
        float temp_phi_power = phi_power;
        float temp_disease_death_rate = disease_death_rate;
        float temp_natural_death_rate = natural_death_rate;
        int temp_SUPER = SUPER;
        int temp_print_rates = print_rates;

        static char *kwlist[] = {"B","N","phi_cutoff","phi_power",
                "disease_death_rate", "natural_death_rate",
                "super_infection","print_rates",NULL};
        if (!PyArg_ParseTupleAndKeywords(args, keywds, "|fiffffii", kwlist,
                &temp_B, &temp_N, &temp_phi_cutoff, &temp_phi_power,
                &temp_disease_death_rate, &temp_natural_death_rate,
                &temp_SUPER,&temp_print_rates))
                return NULL;

        if (temp_B != -1) {
                B = temp_B;
                fp_report << "\tB\t\t\t=\t" << B << ".\n";
        }
        if (temp_N != -1) {
                N = temp_N;
                fp_report << "\tN\t\t\t=\t" << N << ".\n";
        }
        if (temp_phi_cutoff != -1) {
                phi_cutoff = temp_phi_cutoff;
                fp_report << "\tphi_cutoff\t\t=\t" << phi_cutoff << ".\n";
        }
        if (temp_phi_power != -1) {
                phi_power = temp_phi_power;
                fp_report << "\tphi_power\t\t=\t" << phi_power << ".\n";
        }
        if (temp_disease_death_rate != -1) {
                disease_death_rate = temp_disease_death_rate;
                fp_report << "\tdisease_death_rate\t=\t"
                        << disease_death_rate << ".\n";
        }
        if (temp_natural_death_rate != -1) {
                natural_death_rate = temp_natural_death_rate;
                fp_report << "\tnatural_death_rate\t=\t"
                        << natural_death_rate << ".\n";
        }
        if (temp_SUPER != -1) {
```

```
                SUPER = temp_SUPER;
                fp_report << "\tsuper\t\t\t=\t" << SUPER << ".\n";
        }
        if (temp_print_rates != print_rates) {
                print_rates = temp_print_rates;
                fp_report << "\tprint_rates\t=\t" << print_rates << ".\n";
        }

        Py_INCREF(Py_None);
        return Py_None;
}


// ***********************************************************
// Receives output file names from python script

static PyObject*
out_names(PyObject *self, PyObject *args, PyObject *keywds)
{
        const char* pops_filename = "";
        const char* report_filename = "";
        static char *kwlist[] = {"pops","report",NULL};
        if (!PyArg_ParseTupleAndKeywords(args, keywds,"|ss", kwlist,
                &pops_filename, &report_filename)) return NULL;

        fp_pop.open(pops_filename,ios::app);
        fp_report.open(report_filename, ios::app);

        return Py_BuildValue("");
}


// ***********************************************************
// Generates a table of contents array whose indices are
// assigned 1 for valid staes, and 2 for invalid states
// according to the binary encoding scheme
// A second array lists the number of coinfections for those
// states which correspond to coinfections

void generate_toc()
{
        toc = new int[I];
        int flag;
        for (int i=0; i < I; i++) {
                flag = 1;
                for (int j=0; j < M; j++) {
                        if ( (((i>>(M+j))%2)&((i>>j)%2)) ) {
                                flag = 0;
                        }
                }
                if (flag) {
                        toc[i] = 1;
                } else { toc[i] = 0; }
        }
        fp_report << "\tFinished generating the table of "
                << "contents, given below.\n\t\t";
        for (int i=0; i < I; i++)
                fp_report << toc[i] << " ";
        fp_report << "\n";
```

```
        fp_report << "\tFinished generating the table of "
                << " coinfections, given below.\n\t\t";
        toc_coinfect = new int[I];
        int sum;
        for (int w=0; w < I; w++) {
                if (toc[w] == 0) {
                        toc_coinfect[w] = -1;
                        fp_report << toc_coinfect[w] << " ";
                }
                else if ((w%(int)pow(2,M)) != 0) {
                        sum = 0;
                        for (int x=0; x < M; x++) {
                                if (((w>>x)%2)==1) sum++;
                        }
                        if (sum == 1) toc_coinfect[w] = 0;
                        else toc_coinfect[w] = w;
                        fp_report << toc_coinfect[w] << " ";
                }
                else {
                        toc_coinfect[w] = 0;
                        fp_report << toc_coinfect[w] << " ";
                }
        }
        fp_report << "\n";

        return;
}


// ************************************************************
// Sets all population counters to zero,
// except for the naive class

static PyObject*
initialize_counters(PyObject *self, PyObject *args)
{
        if (!PyArg_ParseTuple(args, "")) return NULL;

        states = new int* [I];
        for (int i=0; i < I; i++)
                states[i] = new int [N];
        for (int i=0; i < I; i++) {
                for (int k=0; k < K; k++) {
                        pop[i][k] = 0;
                }
        }
        for (int i=0; i < I; i++) {
                popC[i] = 0;
        }
        for (int i=0; i < I; i++) {
                for (int d = 0; d < N; d++) {
                        states[i][d] = 0;
                }
        }
        pool = new host[N];
        for (int i=0; i < N; i++) {
                pool[i].index = i;
        }
        for (int i=0; i < N; i++) {
                states[0][i] = 1;
```

```
        }
        fp_report << "\tCreated new counters and initialized "
                << N << " hosts.\n";
        update_master();
        t = 0;
        day = 1;

        return Py_BuildValue("");
}

// ************************************************************
// Resets counters to begin a new simulation

static PyObject*
reset_counters(PyObject *self, PyObject *args)
{
        if (!PyArg_ParseTuple(args, "")) return NULL;

        delete [] states;
        delete [] tableP;
        delete [] tableM;
        delete [] tableRi;
        delete [] tableRj;
        delete [] tableRk;
        delete [] tableRl;
        delete [] tableRh;

        fp_pop.close();
        fp_report.close();

        return Py_BuildValue("");
}

// ************************************************************
// Adds a number of infected hosts with a particular strain
// at time zero given by the python script

static PyObject*
seed_infection(PyObject *self, PyObject *args)
{
        int new_infection;
        int targets;
        if (!PyArg_ParseTuple(args, "ii", &new_infection, &targets))
                return NULL;
        int temp = N - popC[0];
        for (int i = temp; i < (targets+temp); i++)
                pool[i].infect(new_infection);
        population_output(1);
        fp_report << "\tInfected " << targets
                << " hosts with strain set "
                << new_infection << ".\n";
        return Py_BuildValue("");
}


// ************************************************************
// f-Function defined by this model

float f_function(float load)
{
```

```
        float ans;
        ans = pow(pow(10,load),phi_power) /
                (pow(pow(10,load),phi_power) + pow(phi_cutoff,phi_power));
        return ans;
}

// ************************************************************
// PHI Function defined by this model

float phi(int x, int z, int w)
{

        // The following control statements include the conditions which a
        // are specific to our implementation of the model
        // they provide an ad hoc method for implementing
        // the infection window and the attenuated strains
        // however, it is recommended that this code be updated
        // for more general specifications on the immunity window
        // and parasite densities for superinfection

        float result = 1;
        if ((z==1) && ((ltable[0] == COI) && (ltable[1] != COI))
                && (ltable[1] > 11) && (ltable[1] <= 49)) {
                return 0.0;
        }
        else if ((z==2) && ((ltable[1] == COI) && (ltable[0] != COI))
                && (ltable[0] > 11) && (ltable[0] <= 49)) {
                return 0.0;
        }
        else if (((x%Q)==3) && (ktable[1] - ktable[0] > 0)
                && (ktable[1] - ktable[0] <= 11) && (SUPER == 1)) {
                if (((z>>0)%2)) {
                        result *=
                        f_function(super21_1[ktable[1]-ktable[0]][ktable[0]]);
                }
                if (((z>>1)%2)) {
                        result *=
                        f_function(super21_2[ktable[1]-ktable[0]][ktable[1]]);
                }
                if (((x>>0)%2) && !((z>>0)%2) && !((w>>0)%2)) {
                        result *=
                        (1-f_function(super21_1[ktable[1]-ktable[0]][ktable[0]]));
                }
                if (((x>>1)%2) && !((z>>1)%2) && !((w>>1)%2)) {
                        result *=
                        (1-f_function(super21_2[ktable[1]-ktable[0]][ktable[1]]));
                }
        }
        else if (((x%Q)==3) && (ktable[0] - ktable[1] > 0)
                && (ktable[0] - ktable[1] <= 11) && (SUPER == 1)) {
                for (int i=0; i < M; i++) {
                        if (((z>>i)%2) && (ktable[i] != COI)) {
                                result *=
                                f_function(data[(x-1)*M+i][ktable[i]]);
                        }
                }
                for (int i=0; i < M; i++) {
                        if (((x>>i)%2) && !((z>>i)%2) && !((w>>i)%2)
                        && (ktable[i] != COI)) {
                                result *=
```

```
                                (1-f_function(data[(x-1)*M+i][ktable[i]]));
                        }
                }
        }
        else if (((x%Q)==3) && ((ktable[0] - ktable[1] > 49)
                || (ktable[1] - ktable[0] > 49)) && (SUPER == 1)) {
                for (int i=0; i < M; i++) {
                        if (((z>>i)%2) && (ktable[i] != COI)) {
                                result *=
                                f_function(data[(x-1)*M+i][ktable[i]]);
                        }
                }
                for (int i=0; i < M; i++) {
                        if (((x>>i)%2) && !((z>>i)%2) && !((w>>i)%2)
                        && (ktable[i] != COI)) {
                                result *=
                                (1-f_function(data[(x-1)*M+i][ktable[i]]));
                        }
                }
        }
        else if (((x%Q)!=3) || (((x%Q)==3) && (ktable[0]==ktable[1]))) {
                for (int i=0; i < M; i++) {
                        if (((z>>i)%2) && (ktable[i] != COI)) {
                                result *=
                                f_function(data[(x-1)*M+i][ktable[i]]);
                        }
                }
                for (int i=0; i < M; i++) {
                        if (((x>>i)%2) && !((z>>i)%2) && !((w>>i)%2)
                        && (ktable[i] != COI)) {
                                result *=
                                (1-f_function(data[(x-1)*M+i][ktable[i]]));
                        }
                }
        }
        else {
                return 0.0;
        }

        return result;
}


// ************************************************************
// Updates a global table which gives the decoded number of
// days for each strain, from a single integer

void update_ktable(int k)
{
        int temp;
        for(int m=0; m < M; m++) {
                k = k/ktableref[m];
                temp = k % ktableref[m+1];
                ktable[m] = temp;
                k -= temp;
        }
        return;
}
void update_ltable(int l)
{
```

```
        int temp;
        for(int m=0; m < M; m++) {
                l = l/ktableref[m];
                temp = l % ktableref[m+1];
                ltable[m] = temp;
                l -= temp;
        }
        return;
}

// ************************************************************
// Generates the master probability table

static PyObject*
generate_probabilities(PyObject *self, PyObject *args)
{
        if (!PyArg_ParseTuple(args, "")) return NULL;
        generate_toc();
        int pass;
        int could_pass;
        int lkvalid;
        A = 0;
        for(int i=0; i < I; i++) {
        for(int j=0; j < J; j++) {
        could_pass = (((i%Q)^(j%Q))^(j%Q))&(((i%Q)^(j%Q)));
        pass = (((could_pass)^(j>>M))^(j>>M))& (((could_pass)^(j>>M)));
        if (SUPER==0) {
        if ((j%Q) != 0) pass = 0; }
        if ((pass != 0) && ((toc[i]*toc[j])%2) ) {
        for (int h=1; h <= H; h++) {
        if ((pass&h) == h) {
        for(int k=0; k < (int)pow ((COI+1),M); k++) {
        for(int l=0; l < (int)pow((COI+1),M); l++) {
                update_ktable(k);
                update_ltable(l);
                lkvalid = 1;
                for (int m=0; m < M; m++) {
                        if ((ktable[m] == COI) && ((((i%Q)>>m)%2)==1))
                                lkvalid = 0;
                        if ((ktable[m] != COI) && ((((i%Q)>>m)%2)==0))
                                lkvalid = 0;
                        if ((ltable[m] == COI) && ((((j%Q)>>m)%2)==1))
                                lkvalid = 0;
                        if ((ltable[m] != COI) && ((((j%Q)>>m)%2)==0))
                                lkvalid = 0;
                        if ((ltable[m] != COI) && (((pass>>m)%2)==1))
                                lkvalid = 0;
                }
        if (((phi(i%Q,h,j>>M)) != 0) && (lkvalid == 1)) {
                A++;
        }}}}}}}}

        tableP = new float [A];
        tableM = new float [A];
        tableRi = new int [A];
        tableRj = new int [A];
        tableRk = new int [A];
        tableRl = new int [A];
        tableRh = new int [A];
        A=0;
```

```
        // Having counted the number of possible reactions,
        // Calculate their raw propensities in the following loops
        // and add them to the tables
        for(int i=0; i < I; i++) {
        for(int j=0; j < J; j++) {
        could_pass = (((i%Q)^(j%Q))^(j%Q))&(((i%Q)^(j%Q)));
        pass = (((could_pass)^(j>>M))^(j>>M))& (((could_pass)^(j>>M)));
        if (SUPER==0) {
        if ((j%Q) != 0) pass = 0; }
        if ((pass != 0) && ((toc[i]*toc[j])%2) ) {
        for (int h=1; h <= H; h++) {
        if ((pass&h) == h) {
        for(int k=0; k < (int)pow ((COI+1),M); k++) {
        for(int l=0; l < (int)pow((COI+1),M); l++) {
                update_ktable(k);
                update_ltable(l);
                lkvalid = 1;
                for (int m=0; m < M; m++) {
                        if ((ktable[m] == COI) && ((((i%Q)>>m)%2)==1))
                                lkvalid = 0;
                        if ((ktable[m] != COI) && ((((i%Q)>>m)%2)==0))
                                lkvalid = 0;
                        if ((ltable[m] == COI) && ((((j%Q)>>m)%2)==1))
                                lkvalid = 0;
                        if ((ltable[m] != COI) && ((((j%Q)>>m)%2)==0))
                                lkvalid = 0;
                        if ((ltable[m] != COI) && (((pass>>m)%2)==1))
                                lkvalid = 0;
                }
                if (((phi(i%Q,h,j>>M)) != 0) && (lkvalid == 1)) {
                        tableP[A] = B*phi(i%Q,h,j>>M);
                        tableRk[A] = k;
                        tableRh[A] = h;
                        tableRj[A] = j;
                        tableRi[A] = i;
                        tableRl[A] = l;
                        A++;
                }

        }}}}}}}
        fp_report << "\tFinished building probabilities table.\n";
        return Py_BuildValue("");
}
```

B.4 The Definition of the Host Class: "host.h"

```
class host
{
public:
        int* age;
                // an array which holds the ages of each infection
        int all_age;
                // a number which encodes infection ages for all strains
        int infection;
```

```
                // binary number where 1 represents infection of the
                // strain corresponding to that binary digit
        int history;
                // binary number representing the strains which
                // the host has recovered from
        int index;
                // a unique identifier for the host
public:
        void encode_age();
        host();
        void infect(int new_infection);
        void recover(int which_strain);
        void increment();
        void print();
        void death();
};

void host::encode_age()
{
        // Updates all_age whenever the individual ages of infection change

        all_age = 0;
        for (int m=0; m < M; m++)
                all_age += age[m]*(int)pow(COI+1,m);
}

host::host()
{
        // Initializes the host and registers it in the pop
        // and popC counters

        age = new int [M];
        for (int i=0; i < M; i++) {
                age[i] = COI;
        }
        history = 0;
        infection = 0;
        popC[((history<<M)+(infection))]++;
        encode_age();
        pop[((history<<M)+(infection))][all_age]++;
}

void host::infect(int new_infection)
{
        // Gives the host a new infection, updates its ages of infection
        // and registers these changes in the overall population counters

        if (print_rates == 1)
                rates[(new_infection-1)+H*((history<<M)+(infection))]++;
        states[((history<<M)+(infection))][index] = 0;
        pop[((history<<M)+(infection))][all_age]--;
        popC[((history<<M)+(infection))]--;
        infection = infection | new_infection;
        for (int s=0; s < M; s++) {
                if ((new_infection>>s)%2 == 1)
                        age[s] = 0;
        }
        encode_age();
        states[((history<<M)+(infection))][index] = 1;
        pop[((history<<M)+(infection))][all_age]++;
```

```
        popC[((history<<M)+(infection))]++;
}

void host::increment()
{
        // Increments the host to the next day
        // and also kills the host according to background noise
        // if the host has no infection, or kills the host with
        // probability according to the disease survivability
        // if the host has an infection

        float random = (float)rand()/RAND_MAX;
        int recover_flag = 0;
        if ((infection == 0) && (random <= natural_death_rate)) {
                death();
                return;
        }
        if (random > (survivability[0][age[0]] <
                survivability[0][age[1]] ?
                survivability[0][age[1]] : survivability[0][age[0]])) {
                death();
                return;
        }
        for (int s=0; s < M; s++) {
                if (age[s] == COI-1) {
                        recover_flag = recover_flag | (1<<s);
                }
        }
        if (recover_flag != 0) {
                recover(recover_flag);
                return;
        }
        pop[((history<<M)+(infection))][all_age]--;
        for (int s=0; s < M; s++) {
                if (age[s] < COI)
                        age[s]++;
        }
        encode_age();
        pop[((history<<M)+(infection))][all_age]++;
}

void host::recover(int which_strains)
{
        // Changes the host's state and updates counters
        // whenever the host recovers from a particular strain

        states[((history<<M)+(infection))][index] = 0;
        popC[((history<<M)+(infection))]--;
        pop[((history<<M)+(infection))][all_age]--;
        history = ((which_strains)|(history));
        infection = infection^which_strains;
        for (int s=0; s < M; s++) {
                if (((which_strains>>s)%2) == 1) {
                        age[s] = COI;
                }
                else if ((((infection>>s)%2) == 1) && (age[s] < COI)) {
                        age[s]++;
                }
        }
        encode_age();
```

```
        states[((history<<M)+(infection))][index] = 1;
        popC[((history<<M)+(infection))]++;
        pop[((history<<M)+(infection))][all_age]++;
}

void host::death()
{
        // Changes the host's state and updates counters
        // whenever the host dies, and is recycled into the naive state

        pop[((history<<M)+(infection))][all_age]--;
        for (int s=0; s < M; s++) {
                age[s] = COI;
        }
        encode_age();
        popC[((history<<M)+(infection))]--;
        states[((history<<M)+(infection))][index] = 0;
        infection = 0;
        history = 0;
        states[((history<<M)+(infection))][index] = 1;
        popC[((history<<M)+(infection))]++;
        pop[((history<<M)+(infection))][all_age]++;
}
```

B.5 The Simulation Functions: "step.h"

```
void step();
float calc_a0();
float gen_tau();
float gen_tau_BD(float f);
void increment_all();
void update_master();
void which_reaction();
void random_infect(int i, int k, int j, int h);
void random_infect_hybrid_specific(int i, int k, int j, int h, int l);

// ************************************************************
// Explicit Method
// one step is equivalent to one reaction (slow)


static PyObject*
step(PyObject *self, PyObject *args)
{
        if (!PyArg_ParseTuple(args, "")) return NULL;

        update_master();
        a0 = calc_a0();
        time_step = gen_tau();
        if ((a0 == 0) || (t+time_step >= day)) {
                population_output(1);
                increment_all();
                t = day;
                day++;
        }
```

```
        else {
                t += time_step;
                which_reaction();
                cout << t << "\n";
        }

        return Py_BuildValue("");
}

// ************************************************************
// Binomial tau-leap method
// scales the time-step by a coarse-graining factor

static PyObject*
step_BD(PyObject *self, PyObject *args)
{
        float coarse;
        if (!PyArg_ParseTuple(args, "f", &coarse)) return NULL;

        int i;
        int j;
        int h;
        int k;
        int kmax;
        int ksample;
        float p;
        update_master();
        a0 = calc_a0();
        time_step = gen_tau_BD(coarse);
        if ((a0 == 0) || (t+time_step >= day)) {
                population_output(1);
                increment_all();
                t = day;
                day++;
                cout << "\n" << t << "\n";
        }
        else {
                for (int a=A-1; a >= 0; a--) {
                        kmax = pop[tableRj[a]][tableRl[a]];
                        p = tableM[a]*time_step/kmax;
                        ksample = gsl_ran_binomial(random_number, p, kmax);
                        while (ksample--) {
                                random_infect_hybrid_specific(tableRi[a],
                                tableRk[a],tableRj[a],tableRh[a],tableRl[a]);
                        }
                }
                t += time_step;
                cout << t << "  ";
        }

        return Py_BuildValue("");
}

// ************************************************************
// Calculate the Propensity Function

float calc_a0()
{
        double result = 0;
        for (int a=0; a < A; a++) {
```

```
                result += tableM[a];
        }
        return result;
}

// ************************************************************
// Calculate the Time-Step for the Explicit Method

float gen_tau()
{
        float tau;
        float r1 = (float)((rand()+1.0)/(RAND_MAX+1.0));
        tau = (1/a0)*log(1/r1);
        return tau;
}

// ************************************************************
// Calculate the Time-Step for the Tau-Leaping Method

float gen_tau_BD(float f)
{
        float tau;
        tau = f / a0;
        return tau;
}

// ************************************************************
// Increment each Host's Infection clock by one day
// this function is called whenever the current time + time
// until the next reaction exceeds the next integer day

void increment_all()
{
        for (int i = 0; i < N; i++) {
                pool[i].increment();
        }
}

// ************************************************************
// Update the Master Table of Propensities
// Multiplies the probability of each reaction by the product of
// the number of each reactant:
// one infecting host and one receiving host

void update_master()
{
        for (int a=0; a < A; a++) {
                tableM[a] = tableP[a]*pop[tableRj[a]][tableRl[a]]
                                *pop[tableRi[a]][tableRk[a]];
        }
        return;
}

// ************************************************************
// Determine the Next Reaction

void which_reaction()
{
        float r2 = (float)rand()/RAND_MAX;
        float running_sum = 0;
```

```
        int i;
        int j;
        int h;
        int k;
        for (int a=0; a < A; a++) {
                running_sum += tableM[a];
                if (running_sum > a0*r2) {
                        random_infect_hybrid(tableRi[a],
                        tableRk[a],tableRj[a],tableRh[a]);
                        return;
                }
        }
}

// ************************************************************
// Determine the Next Reaction (explicit method)

void random_infect(int i, int k, int j, int h)
{
        int r = (rand() % popC[j]);
        for (int n = 0; n < N; n++) {
                if (states[j][n] == 1) {
                        if (r == 0) {
                                pool[n].infect(h);
                                return;
                        }
                        r--;
                }
        }
}

// ************************************************************
// Randomly infect a new host of a particular type
// which is specified by k (its state) and
// l (the number which encodes both its ages of infection)
// where h specifies which strains are being transmitted

void random_infect_hybrid_specific(int i, int k, int j, int h, int l)
{
        int r;
        if (!(pop[j][l] < N/1000)) {
                while (1) {
                        r = (rand() % N);
                        if ((pool[r].all_age == l) &&
                        (((pool[r].history<<M)+(pool[r].infection)) == j)) {
                                pool[r].infect(h);
                                return;
                        }
                }
        }
        else {
                r = (rand() % pop[j][l]);
                for (int n = 0; n < N; n++) {
                        if ((pool[n].all_age == l) &&
                        (((pool[n].history<<M)+(pool[n].infection)) == j)) {
                                if (r == 0) {
                                        pool[n].infect(h);
                                        return;
                                }
                                r--;
```

```
                        }
                }
        }
}
```

B.6 The Output Code: "output.h"

```
void population_output(int increment)
{
        int sum;
        if((step_num%increment) == 0) {
        fp_pop << t << "\t";
        for (int i=0; i < 16; i++)
        {
        if (((i%4)==0) && (toc[i]==1))
        {
                fp_pop << popC[i] << "\t";
        }
        else if (toc_coinfect[i] == 0)
        {
                if ((i%4) == 1)
                {
                        fp_pop << pop[i][0+COI*(COI+1)] << "\t";
                }
                if ((i%4) == 2)
                {
                        fp_pop << pop[i][COI+0*(COI+1)] << "\t";
                }

        else if (toc_coinfect[i] != -1 )
        {
        sum = 0;
        for (int k=0; k < (COI+1)*(COI+1); k++)
        {
                if ((k%(COI+1) == 0) || (floor(k/(COI+1)) == 0))
                        sum += pop[i][k];
        }
        fp_pop << sum << "\t";
        }
        }
        fp_pop << endl;

        if (print_rates == 1) {
        for (int j=0; j < J; j++) {
        if (toc[j] == 1) {
                fp_report << "";
                for (int h = 0; h < H; h++) {
                        fp_report << rates[h+H*j] << ",";
                }
                fp_report << "\t";
        }
        }
        fp_report << endl;
        for (int i=0; i < H*L; i++)
                rates[i] = 0;
```

```
            }
            }
}
```

References

[1] H. K. Alles, K. N. Mendis, and R. Carter, *Malaria mortality rates in south asia and in africa: Implications for malaria control*, Parasitology Today, 14 (1998), pp. 369–375.

[2] S. Altizer, A. Dobson, P. Hosseini, P. Hudson, M. Pascual, and P. Rohani, *Seasonality and the dynamics of infectious diseases*, Ecology Letters, 9 (2006), pp. 467–484.

[3] C. P. Babalola, O. O. Bolaji, F. A. Ogunbona, A. Sowunmi, and O. Walker, *Pharmacokinetics of quinine in african patients with acute falciparum malaria.*, Pharmacy World & Science, 20 (1998), pp. 118–122.

[4] A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis, *Binomial distribution based tau-leap accelerated stochastic simulation*, The Journal of Chemical Physics, 122 (2005), pp. 024112–7.

[5] Q. Chen, M. Schlichtherle, and M. Wahlgren, *Molecular aspects of severe malaria*, Clinical Microbiology Review, 13 (2000), pp. 439–450.

[6] A. Craig and A. Scherf, *Molecules on the surface of the plasmodium falciparum infected erythrocyte and their role in malaria pathogenesis and immune evasion*, Molecular and Biochemical Parasitology, 115 (2001), pp. 129–143.

[7] K. P. DAY, J. C. KOELLA, S. NEE, S. GUPTA, and A. F. READ, *Population-genetics and dynamics of plasmodium-falciparum - an ecological view*, Parasitology, 104 (1992), pp. S35–S52.

[8] P. De Leenheer and S. S. Pilyugin, *Immune response to a malaria infection: properties of a mathematical model*, Journal of Biological Dynamics, 2 (2008), pp. 102–120.

[9] J. de Roode, M. Helinski, M. Anwar, and A. Read, *Dynamics of multiple infection and within-host competition in genetically diverse malaria infections*, The American Naturalist, 166 (2005), pp. 531–542.

[10] M. Derouich, A. Boutayeb, and E. Twizell, *A model of dengue fever*, BioMedical Engineering OnLine, 2 (2003), p. 4.

[11] N. M. Ferguson, D. A. T. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke, *Strategies for mitigating an influenza pandemic*, Nature, 442 (2006), pp. 448–452.

[12] C. Fraser, S. Riley, R. M. Anderson, and N. M. Ferguson, *Factors that make an infectious disease outbreak controllable*, Proceedings of the National Academy of Sciences of the United States of America, 101 (2004), pp. 6146–6151.

[13] C. R. S. Garcia, R. P. Markus, and L. Madeira, *Tertian and quartan fevers: Temporal regulation in malarial infection*, The Journal of Biological Rhythms, 16 (2001), pp. 436–443.

[14] D. T. Gillespie, *Exact stochastic simulation of coupled chemical reactions*, The Journal of Physical Chemistry, 81 (1977), pp. 2340–2361.

[15] J. Gog and J. Swinton, *A status-based approach to multiple strain dynamics*, Journal of Mathematical Biology, 44 (2002), pp. 169–184.

[16] J. R. Gog and B. T. Grenfell, *Dynamics and selection of many-strain pathogens*, Proceedings of the National Academy of Sciences of the United States of America, 99 (2002), pp. 17209–17214.

[17] D. Gollin and C. Zimmermann, *Malaria: Disease impacts and long-run income differences*, tech. report, August 2007. IZA Discussion Paper No. 2997. Available at SSRN: http://ssrn.com/abstract=1012564.

[18] N. C. Grassly and C. Fraser, *Mathematical models of infectious disease transmission*, Nature Review Microbiology, 6 (2008), pp. 477–487.

[19] D. Greenwood, *The quinine connection*, The Journal of Antimicrobial Chemotherapy, 30 (1992), pp. 417–427.

[20] B. Hellriegel, *Modelling the immune response to malaria with ecological concepts: Short-term behaviour against long-term equilibrium*, Proceedings: Biological Sciences, 250 (1992), pp. 249–256.

[21] H. Hethcote, *A thousand and one epidemic models*, vol. 100 of Lecture Notes in Biomathematics, Springer-Verlag, Berlin, 1994, pp. 504–515.

[22] H. W. Hethcote, *The mathematics of infectious diseases*, SIAM Review, 42 (2000), pp. 599–653.

[23] H. Inaba, *Threshold and stability results for an age-structured epidemic model*, Journal of Mathematical Biology, 28 (1990), pp. 411–434.

[24] J. Kelsall and J. Wakefield, *Modeling spatial variation in disease risk*, Journal of the American Statistical Association, 97 (2002), pp. 692–701.

[25] C. Lengeler, *Insecticide-treated bed nets and curtains for preventing malaria*, Cochrane Database of Systematic Reviews, 2 (2009), p. CD000363.

[26] C. D. Mathers, R. Sadana, J. A. Salomon, C. J. Murray, and A. D. Lopez, *Healthy life expectancy in 191 countries, 1999*, The Lancet, 357 (2001), pp. 1685–1691.

[27] S. R. Meshnick, *Artemisinin: mechanisms of action, resistance and toxicity*, International Journal for Parasitology, 32 (2002), pp. 1655–1660.

[28] L. H. Miller, D. I. Baruch, K. Marsh, and O. K. Doumbo, *The pathogenic basis of malaria*, Nature, 415 (2002), pp. 673–679.

[29] L. H. Miller, M. F. Good, and G. Milon, *Malaria pathogenesis*, Science, 264 (1994), pp. 1878–1883.

[30] D. Mollison, *Spatial contact models for ecological and epidemic spread*, Journal of the Royal Statistical Society. Series B (Methodological), 39 (1977), pp. 283–326.

[31] P. Olliaro and W. Taylor, *Developing artemisinin based drug combinations for the treatment of drug resistant falciparum malaria: A review*, Journal of Postgraduate Medicine, 50 (2004), pp. 40–44.

[32] D. Sacks and A. Sher, *Evasion of innate immunity by parasitic protozoa*, Nature Immunology, 3 (2002), pp. 1041–1047.

[33] C. Sibley, K. Barnes, and C. Plowe, *The rationale and plan for creating a world antimalarial resistance network (warn)*, Malaria Journal, 6 (2007), p. 118.

[34] T. SMITH, N. MAIRE, A. ROSS, M. PENNY, N. CHITNIS, A. SCHAPIRA, A. STUDER, B. GENTON, C. LENGELER, F. TEDIOSI, D. DE SAVIGNY, and M. TANNER, *Towards a comprehensive simulation model of malaria epidemiology and control*, Parasitology, 135 (2008), pp. 1507–1516.

[35] R. W. Snow, C. A. Guerra, A. M. Noor, H. Y. Myint, and S. I. Hay, *The global distribution of clinical episodes of plasmodium falciparum malaria*, Nature, 434 (2005), pp. 214–217.

[36] J. R. Songer, *Influence of relative humidity on the survival of some airborne viruses*, Appl. Environ. Microbiol., 15 (1967), pp. 35–42.

[37] D. J. Sullivan, I. Y. Gluzman, D. G. Russell, and D. E. Goldberg, *On the molecular mechanism of chloroquine's antimalarial action*, Proceedings of the National Academy of Sciences of the United States of America, 93 (1996), pp. 11865–11870.

[38] A. Talman, O. Domarle, F. McKenzie, F. Ariey, and V. Robert, *Gametocytogenesis : the puberty of plasmodium falciparum*, Malaria Journal, 3 (2004), p. 24.

[39] H. R. Thieme, *Mathematics in population biology*, Princeton University Press, 2003, pp. 332–338.

[40] H. R. Thieme and C. Castillo-Chavez, *On the role of variable infectivity in the dynamics of the human immunodeficiency virus epidemic*, (1989), pp. 157–176.

[41] A. Trampuz, M. Jereb, I. Muzlovic, and R. Prabhu, *Clinical review: Severe malaria*, Critical Care, 7 (2003), pp. 315–323.

[42] P. Trigg and A. Kondrachine, *Commentary: Malaria control in the 1990s*, World Health Organization. Bulletin of the World Health Organization., 76 (1998), pp. 11–14.

[43] W. H. Wernsdorfer, *Epidemiology of drug resistance in malaria*, Acta Tropica, 56 (1994), pp. 143–156.

[44] S. Yeung, W. Pongtavornpinyo, I. M. Hastings, A. J. Mills, and N. J. White, *Antimalarial drug resistance, artemisinin-based combination therapy, and the contribution of modeling to elucidating policy choices*, American Journal Of Tropical Medicine And Hygiene, 71 (2004), pp. 179–186.