Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Johnathan Jia April 15, 2015

Investigating heterogeneity in the dynamics of virus and immune response following a yellow fever vaccination.

by

Johnathan Jia

Dr. Rustom Antia
Adviser

Department of Biology

Dr. Rustom Antia

Adviser

Dr. Arri Eisen

Committee Member

Dr. Jacobus De Roode

Committee Member

Dr. Lance Waller

Committee Member

2015

Investigating heterogeneity in the dynamics of virus and immune response following a yellow fever vaccination.

By

Johnathan Jia

Dr. Rustom Antia

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2015

Abstract

Investigating heterogeneity in the dynamics of virus and immune response following yellow fever vaccination

By Johnathan Jia

A key feature of the dynamics of infection in humans is that different individuals form different responses to a given pathogen. This has been demonstrated in recent studies conducted by Akondy et al. (2015) following immunization of the yellow fever vaccine. Their results show that there is a 10^4 fold variation in the peak viral titer and 10^2 fold variation in the peak immune cell density among human participants. We used simple mathematical models of the within host dynamics of infection to investigate what gives rise to variation between individuals in the dynamics of infection and immunity. We found that changes in most parameters over biologically reasonable ranges resulted in greater changes in peak viral load compared to the peak immune cell density. Only changes in the killing rate of adaptive immunity resulted in similar changes in the peak viral load and peak immune cell density. Our simple model does not recapitulate the yellow fever vaccination data. This indicates that our model assumptions are incorrect. Since our model is the simplest case, we must have failed to consider an important process or factor that would be necessary to recapitulate the observed variation in the yellow fever vaccination data. However our model makes predictions which can be experimentally tested, and in doing so it sets the stage for a quantitative understanding of what gives rise to heterogeneity in the responses of different individuals and infection.

Investigating heterogeneity in the dynamics of virus and immune response following yellow fever vaccination.

By

Johnathan Jia

Dr. Rustom Antia

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2015

Acknowledgements

We thank Dr. Veronika Zarnitsyna and Dr. Philip Johnson for their help with the completion of the honor's thesis.

We also thank Dr. Eisen, Dr. Jacobus De Roode, and Dr. Lance Waller for the helpful comments on the thesis.

Lastly we thank the publishers at Elsevier for granting us copyright permission for the figure in our background.

Table of Contents

Figures and Tables

Figures

Tables

Investigating heterogeneity in the dynamics of virus and immune response following a yellow fever vaccination.

Johnathan Jia

April 21, 2015

1 Abstract

A key feature of the dynamics of infection in humans is that different individuals form different responses to a given pathogen. This has been demonstrated in recent studies conducted by Akondy et al. (2015) following immunization of the yellow fever vaccine. Their results show that there is a 10^4 fold variation in the peak viral titer and 10^2 fold variation in the peak immune cell density among human participants. We used simple mathematical models of the within host dynamics of infection to investigate what gives rise to variation between individuals in the dynamics of infection and immunity. We found that changes in most parameters over biologically reasonable ranges resulted in greater changes in peak viral load compared to the peak immune cell density. Only changes the mass action killing rate of adaptive immunity resulted in similar changes in the peak viral load and peak immune cell density. Our simple model does not recapitulate the yellow fever vaccination data. This indicates that our model assumptions are incorrect. Since our model is the simplest case, we must have failed to consider an important process or factor that would be necessary to recapitulate the observed variation in the yellow fever vaccination data. However our model makes predictions which can be experimentally tested, and in doing so it sets the stage for a quantitative understanding of what gives rise to heterogeneity in the responses of different individuals and infection.

2 Background

2.1 The Question and The Problem

The majority of the information we have about infections and immunity has been discovered through experimentation upon laboratory bred mice. The actual goal is to learn about the human immune system and how it interacts with foreign microbes after they have infected the body. We use mice as a model system because the mouse immune system is similar to the human immune system, and the model system can be utilized to learn more about the adaptive immune system's dynamics when a foreign pathogen is introduced into the host.

When using laboratory mice to discover the rules for the dynamics of infection and immunity, it is best to minimize variation to prevent any confounding factors from affecting the results. In order to minimize variation the experimental system controls for variation from genetics, age, gender, and environmental factors such as the previous exposure to other pathogens.

Studies use laboratory mice that are inbred which results in a genetically homogenous group of mice. These mice also share the same living environment so that heterogeneity that may arise from the environment is minimized as well. Furthermore, in experiments conducted with mice, the mice are also controlled to have the same age and sex to further minimize any variation in the experiments. As a result, the data on the dynamics of infection in laboratory mice has low amounts of variation. This can be best seen in the small error bars present in Figure 1. Figure 1 shows the population dynamics of the epitope specific CD8 T cells over the course of an LCMV infection. Murali-Krishna et al. (1998) quantified the number of the CD8 T cells specific for two different structural protein epitopes over the time course of a Lymphocytic choriomeningitis virus (LCMV) infection in mice [1]. We know the relation between sample size (n), standard error (SE), and standard deviation (s). The equation is $SE = \frac{s}{\sqrt{n}}$. The sample sizes used at each time point varied between three to five mice. In Figure 1 the width of the standard error bars of the immune cell density at each point indicates that the standard error is low. Using the equation, given that $n = 3$ or $n = 5$, we find that the standard deviation is even lower. The standard deviation is even lower therefore the variation in the mice is low.

In contrast to laboratory mice, humans are usually not genetically inbred and do not live in a controlled environment which results in a heterogenous

human population. While the general rules of the immune response are the same for both mice and humans, the outcomes of infection in a human have greater variation compared to the outcomes of infection in a mouse.

There are data present on the pathogen and immune cell dynamics over the time course of infection in a mouse host, but there are less data present on the dynamics of infection within a human host. More specifically there is very little quantified data on the size of pathogen and immune cell populations over the time course of an acute infection within a human host. Since we lack access to such data, the biological causes of the heterogeneity in the dynamics of infection in the human population are currently unknown.

The yellow fever vaccine contains a live attenuated form of the virus. This form replicates and causes a mild acute infection in immunized individuals. It also generates a very robust and protective immune response. Indeed it is one of our best vaccines, having been distributed to approximately 1 billion individuals worldwide [2]. Recently Akondy et al. (2015) measured the dynamics of the yellow fever virus and the CD8+ T- cell response following immunization [3]. Interestingly even though the viral inoculum dose is controlled for immunizations, there is an enormous amount of variation present in the peak viral load (10^4) in different individuals as well as the magnitude of the immune responses elicited (10^2). This variation is best shown by Figure 2.

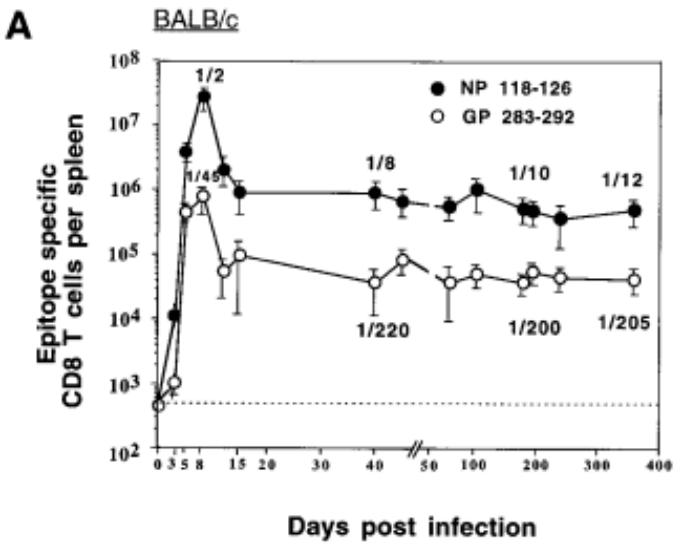

Figure 1: Dynamics of the CD8 T-Cell Response towards 2 different epitopes in mice after infection by Lymphocytic chroriomeningitis virus (LCMV). Murali-Krishna et al. (1998) infected BALB/c laboratory mice with LCMV to measure two CD8 T-cell population specific for the nucleoprotein (NP) and the glycoprotein (GP) epitopes after infection. Nucleoprotein and glycoprotein are two of the three major structural proteins of Lymphocytic choriomeningitis virus [4]. They found that the NP epitope elicited a higher peak CD8 T-cell density, and it can be seen from the figure that the differences in peak effector CD8 T cells per spleen specific for the two LCMV epitopes differed by less than 100 fold. It can be seen that the mice have relatively low variation because the standard error bars are narrow and the number of mice used for the experiments was low as well. As a result, the standard deviation and variance are low.

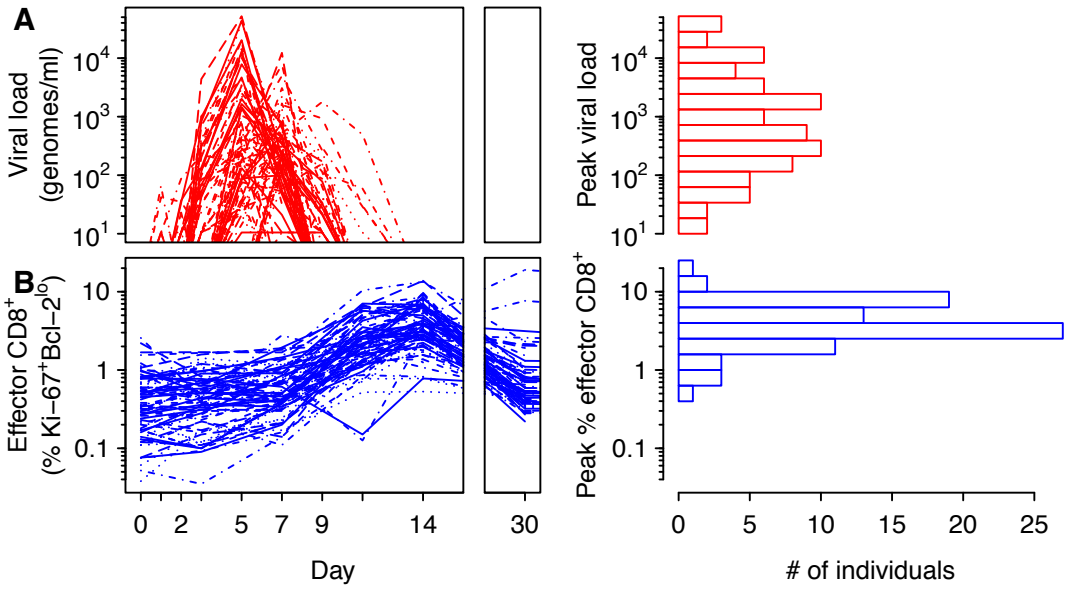

Figure 2: Dynamics of Infection and Immunity within humans after Immunization by the Yellow Fever Vaccine. Peak viral load and peak % effector CD8 T cell density along with the dynamics of the yellow fever virus and the YFV specific CD8 T cells were measured after immunization by the live-attenuated yellow fever virus. There were 80 participants and all were young adults between the ages of 18 and 40. Viral load was measured using quantitative reverse transcriptase polymerase chain reaction (qRT-PCR), and the activated CD8 T cells were measured using activation markers or tetramers that recognized specific CD8 T cells. It can be seen that there is variation present in the peak viral load (10^4) and a smaller amount of variation in the peak % effector CD8 T cell population (10^2).

2.2 Sources of Heterogeneity

Here are some of the sources of heterogeneity that have been studied and how they affect the dynamics of infection within a host. They have been categorized into larger categories which are then stratified into smaller subcategories to elucidate how certain genetic, environmental, pathogen, or other heterogenous characteristics affect the dynamics of infection.

1. Host and Pathogen Genetic Factors

   (a) Host Gender
   (b) MHC and Other Genes
   (c) Antigen Presentation
   (d) Pathogen Genetic Variation

2. Environmental Factors

   (a) Season, Temperature, Humidity
   (b) Spatial Distribution of Hosts
   (c) Host Nutrition
   (d) Host Exercise
   (e) Host Past Infectious Background

3. Other

   (a) Host Age

2.3 How Do Genetic Factors Impact the Within-host Dynamics of Infection

2.3.1 MHC Genes and Other Genes

It is known that there is a great amount of diversity in the genes that encode for the major histocompatibility complex (MHC) [5]. Human leukocyte antigen (HLA) genes are the human MHC genes. Individuals with homozygosity in the HLA group 2 alleles predicted treatment failure in chronic hepatitis C infection [6]. It has been shown that HLA variation in the T helper 2 gene cluster is associated with the ability to mount an immune response to mycobacterial epitopes [7]. Diversity in the MHC genes of a host could allow the host to mount different responses. It has been shown that macaques with diverse MHC genotypes will have better clinical outcomes following SIV infection compared to those that are homogenous for the MHC genes [8]. It has also been shown that variation in the genes that encode for the gamma-interferon receptor. Allelic variation of the vitamin D receptor has been linked with variation in susceptibility to some infectious diseases [9].

2.3.2 Host Gender

It has also been shown that vaccination during the morning forms an increased antibody response in men than vaccination during the afternoon [10], but it was found that acute stress prior to vaccination enhances the antibody response in women [11]. A number of immunological differences between men and women have been found, but gender specific differences in the dynamics of infection ultimately depends on the pathogen involved and the gender specific differences with the specific immune response [12].

2.3.3 Previous Antigenic History

Unlike mice, humans have different infection histories. Not every individual has been infected by the same pathogen at the same point in their lifetimes. Past antigenic history has the possibility of affecting or changing the density of immune cells specific for an epitope on the pathogen. If the host has had an infection with a similar pathogen before, provided they formed an adaptive immune response, the presence of memory cells upon secondary infection can greatly change the dynamics of infection as the response of memory cells to a given antigen results in faster and more aggressive proliferation of the immune cells to clear out the antigen [13].

2.4 How Do Environmental Factors Impact the Dynamics of Infection Within A Host?

2.4.1 Seasonality/Humidity/Temperature

Certain seasons could make it more or less likely for the host to form a robust immune response. It has been shown that the season can affect host susceptibility to anthrax infections due to seasonal gastrointestinal helminth infections [14]. There is also evidence that winter facilitates pathogen survival, and the host immune system is relatively weakened during winter as well. Both of these factors lead to increased host susceptibility to viral infections during winter [15]. It has been shown that activation of adaptive immune cells is temperature dependent, but effector cell activity is nearly temperature independent [16]. The season could affect the dynamics of response by changing the temperature and therefore affecting activation of adaptive immune cells. For example it has been shown that temperatures above room

temperature, tropical phages of tropical bacterial pathogens undergo a lytic cycle and at room temperature undergo a temperate cycle [17].

2.4.2 Host Lifestyle/Exercise

The effects of exercise upon the immune system have been studied by other researchers. The experiments found that after exhaustive exercise, young male mice had significantly decreased immune cell activity and memory cell formation. On the other hand old male mice did not have a change in the immune cell activity, and their levels of CD8 specific immune cells did not change after intensive/exhaustive exercise [18]. Exercise could change the growth rate of immunity or killing rate of immunity, and as a result change the dynamics of infection between different individuals.

2.4.3 Host Nutrition

There is also evidence that nutrition, specifically protein nutrition, is linked to increased likelihood of infection. It was shown that protein malnutrition can cause a 2-fold decrease in the number of specific CD8 memory T cells, and that proliferation of the CD8 memory T cells along with responsiveness had significantly decreased in the mice with protein malnutrition [19]. There is also evidence that intrauterine growth-retardation due to malnutrition could disrupt the development of the immune system and cause permanent impairments [20]

2.5 How does Pathogen Heterogeneity Affect the Immune Response

Pathogen heterogeneity can generate large amounts of variation in the dynamics of infection. The pathogen could grow faster or slower because the pathogen has inherent variation in the dynamics of growth once it has invaded the host. Adaptive immune cells could also be specific for different epitopes on a given pathogen, and the more immunogenic epitopes could elicit a more robust immune response as seen in Figure 1. It is clear in Figure 1 that nucleoprotein (NP) is more immunogenic than glycoprotein (GP), and the peak immune cell density specific for NP is greater than the density of adaptive immune cells specific for GP. Given pathogens can also vary their lifestyles in response to different stimuli such as increased temperature, which causes

some phages to undergo a lytic cycle as opposed to a temperate cycle [17]. Herpesviruses can also epigenetically activate or deactivate genes required to switch from a lytic to a latent cycle [21]. It has been hypothesized that the switching of life cycles is an adaptive response to increase fitness when there is predation from the immune system [22]. Variation in a given pathogen's ability to switch between differing life cycles could cause variation in the dynamics of infection as well.

2.6 How do other heterogenous factors influence the dynamics of an acute infection?

2.6.1 Host Age

The effects of age on the dynamics of immunity and microorganisms within the host have also been studied. It has been shown that as mice get older their ability to generate memory T cells decreases, but the homeostatic mechanisms that maintain the population of memory cells remain intact with age [23]. Furthermore young mice have a suppression of specific primary T-cell response after intense physical exercise while old mice do not [18].

2.7 Our Approach

Each of these genetic, environmental, pathogen-specific, and other factors have specific influences and effects upon the dynamics of infection within a host. Past antigenic history could result in a higher initial immunity towards the same pathogen, closer spatial locality of hosts could result in a higher initial infectious inoculum, and increasing age could result in a host with a weaker immune system. The biology of how these factors affect the interplay between the immune system and the invading pathogen is best repesented by the parameters in a model.

We will generate a simple model of the dynamics of infection within a host. We will then ask several questions using the model. What are the key parameters that influence the dynamics of infection? Which parameters do we expect to vary? Why is the variation in the peak viral load greater than the variation in the peak effector CD8 T cell number? What is the effect of variation in different parameters? Does it recapitulate the yellow fever virus immunization data?

We will attempt to tease out the major biological determinants in generating the observed heterogeneity in the population. Using our simple models, we will first introduce heterogeneity by varying a single parameter within a certain biological range (See Model section). We will observe how varying our model parameters over the biological range of values changes the magnitude of peak virus and peak immunity, and we will see how the variation in peak viremia and peak immunity is affected by heterogeneity in our model parameters.

3 The Model

We have generated a simple model of the dynamics of an acute infection [24]. Our model incorporates the effects of pathogen growth, growth of adaptive immunity in response to the presence of a foreign microbe, and the killing and clearance of the pathogen by the adaptive immune system. We are modeling the dynamics of an acute infection. Our model schematic can be seen in Figure 3.

3.1 Model Schematic

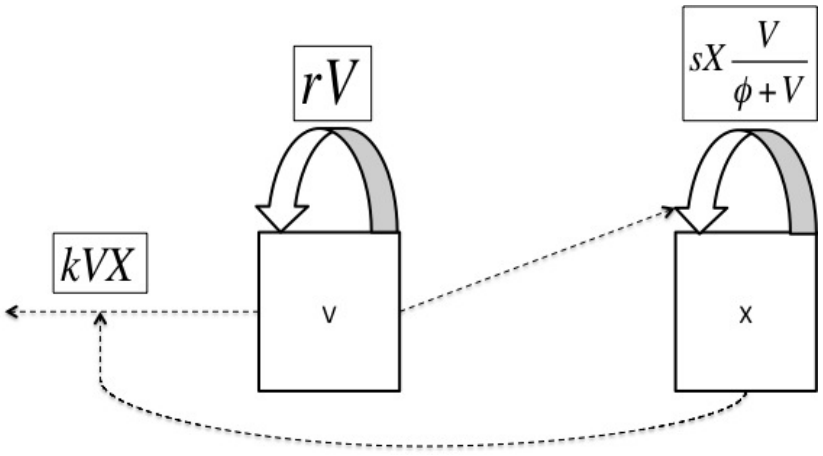

Figure 3: Model Schematic

3.2 Assumptions

1. Pathogen Growth
   (a) The pathogen grows exponentially when immunity is not present.
   (b) The pathogen density required for host mortality is extremely high, and so host death due to pathology is assumed to not occur.
2. Immune Growth
   (a) Immunity grows in a manner dependent on the pathogen density given by $\frac{V}{\phi+V}$
   (b) The immune system kills the pathogen by mass action
   (c) The immune system grows and saturates at a high density until the pathogen has been cleared.
   (d) A high density of immune cells is needed to clear the pathogen
   (e) The effects of adaptive immune cell exhaustion is negligible

3.3 Model Equations

The assumptions listed above generate the simplest possible model of the within host dynamics of a host infected by a pathogen. The equations are given below where V is the number of virus and X is the number of adaptive immune cells specific for the given pathogen:

$$\text{(Virus)} \qquad \frac{dV}{dt} = rV - kVX \tag{1}$$

$$\text{(Adaptive Immunity)} \qquad \frac{dX}{dt} = sX\frac{V}{\phi + V} \tag{2}$$

3.4 Parameterization

Table 1: Parameter units, definitions, mean values, and simulation ranges

Parameter, units	Definition	Mean Value	Range
r, $days^{-1}$	Virus Growth Rate	2	1-3
s, $days^{-1}$	Max Immune Cell Growth Rate	1	0.9-1.1
ϕ, V	Pathogen Density Needed For Half Max Immune Growth	10^3	10^3-10^6
k, $(X * days)^{-1}$	Killing rate by adaptive immunity	10^{-3}	10^{-6}-10^{-3}
$V(0)$, V	Initial Viral Inoculum	1	1-10^3
$X(0)$, X	Initial Immunity Present	1	1-10^2

3.4.1 Pathogen Growth Rate: r

The range chosen for the pathogen growth rate $(1 - 10)$ corresponds to a doubling time from a couple hours to a few days. This range reflects a biologically meaningful range for the growth rate of the pathogen, and the range used in the simulations was from $1 - 3$. This was chosen because the distribution of values used for r were centered and symmetrical about a mean of 2. In order to increase the range to 10, using the same mean value and distribution we were using before, r would have negative values which do not hold any biological significance at all.

3.4.2 Maximum Rate of Immune Cell Growth: s

Our parameter s represents the maximum rate at which the adaptive immune cells grow. The biological range of the maximum growth rate for immunity is small because the same cell type should grow similarly even in different hosts. We can also see that even in humans, the dynamics of the immune response are similar enough to suggest that the rate at which immunity grows does not vary greatly in the human population.

3.4.3 Pathogen density required for Half the Maximum Rate of Immune Cell Growth: ϕ

Our parameter ϕ represents the pathogen density required for half of the maximum adaptve immune growth rate the pathogen density at which immunity responds. The mean value chosen for ϕ is biologically relevant because after the pathogen enters the host, it requires time for the pathogen to grow. It is only after the pathogen reaches a certain density that it begins to either cause damage or provide enough epitopes/antigen to cause an immune response.

3.4.4 Rate of Killing by Adaptive Immunity: k

The rate of killing by the immune system is represented by the parameter k, and the value chosen for k is much lower than the initial immunity present ($X(0) = 1$) since we assumed that a high level of pathogen-specific immune cells is required to clear the pathogen.

For $\dot{V} = 0$, $rV = kVX$. This gives the result $k = \frac{r}{X}$ and since X_{max} is approximately 10^3, the mean value of k is 10^{-3} as indicated in Table 1.

3.4.5 Initial Viral Inoculum : $V(0)$

We are also interested in how differences in the infectious dose of a given pathogen can affect the variation of the dynamics of infection within a host. We assume that the initial infectious dose is very small, $V(0) = 1$. The range of $V(0)$ is exponential because the virus grows exponentially, and so changes in the initial viral inoculum also reflect this growth. Linear changes in $V(0)$ have little to no effect upon changes in the dynamics of infection.

3.4.6 Initial Immunity Present : $X(0)$

We also assume that the host has no previous experience with the given pathogen. A naive host has a very small population of immune cells specific to the pathogen, $X(0) = 1$. Variance in the initial immunity reflect different antigenic history. Individuals with a higher density of immune cells specific to a given pathogen may have had a previous infection, or they could have previously been immunized against the pathogen. Our range of interest extends from what would be a naive host to a host who has previous antigenic history, but not enough memory to generate sterilizing immunity. Sterilizing immunity describes the situation where the host has a high enough density

of memory cells specific to the pathogen such that the pathogen population immediately goes to extinction after subsequent infections by the same pathogen.

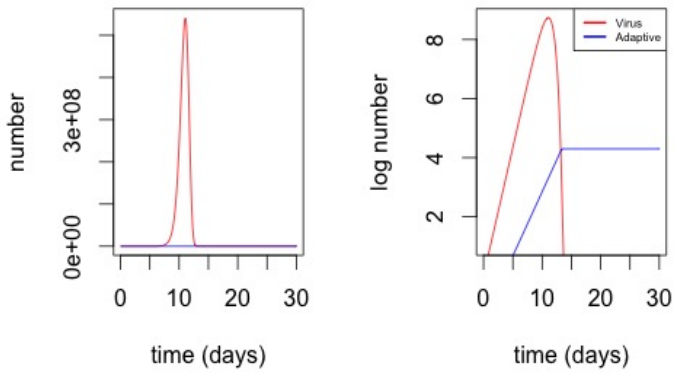

Figure 4: Model Simulation with All Parameters Equal to Mean Values. This is a single simulation of our model, linear scale on the left and $\log_{10}$ scale on the right, with all parameters equal to their respective mean values. Similar simulations will be run with heterogeneity introduced into our model parameters one at a time while keeping all others constant, and from each simulation we will measure V_{max} and X_{max} which can be seen in this figure as the peak of the curve for virus and the maximum of the curve for immunity.

4 Results

From our model simulations we are interested in the peak viral load, the maximum amount of immunity generated at end of the infection, the variation in these outputs, and the mean value of these outputs. We are also interested in the coefficient of variation of the peak viral load and the coefficient of variation of the maximum amount of immunity generated. From these outputs we hope to determine which parameters cause the most variation in the dynamics of infection and how much variation is generated when heterogeneity is introduced into the model parameters.

1. Parameter (denoted by x)

(a) Mean of the parameter: μ_x

(b) Variance of the parameter: σ_x^2

(c) Coefficient of Variation: $C_v < x >= \frac{\sigma_x^2}{\mu<x>}$

2. Pathogen

(a) The $\log_{10}$ of peak viremia during an infection: $\log_{10} V_{max}$

(b) The variance of $\log_{10} V_{max}$: $\sigma^2_{\log_{10} V_{max}}$

(c) The mean of $\log_{10} V_{max}$: $\mu_{\log_{10} V_{max}}$

(d) $C_v < \log_{10} V_{max} >= \frac{\sigma^2_{\log_{10} V_{max}}}{\mu<\log_{10} V_{max}>}$

3. Immunity

(a) The $\log_{10}$ of maximum immunity generated during an infection: $\log_{10} X_{max}$

(b) The variance of $\log_{10} X_{max}$: $\sigma^2_{\log_{10} X_{max}}$

(c) The mean of $\log_{10} X_{max}$: $\mu_{\log_{10} X_{max}}$

(d) $C_v < \log_{10} X_{max} >= \frac{\sigma^2_{\log_{10} X_{max}}}{\mu<\log_{10} X_{max}>}$

4.1 Variation in Viral Growth Rate: r

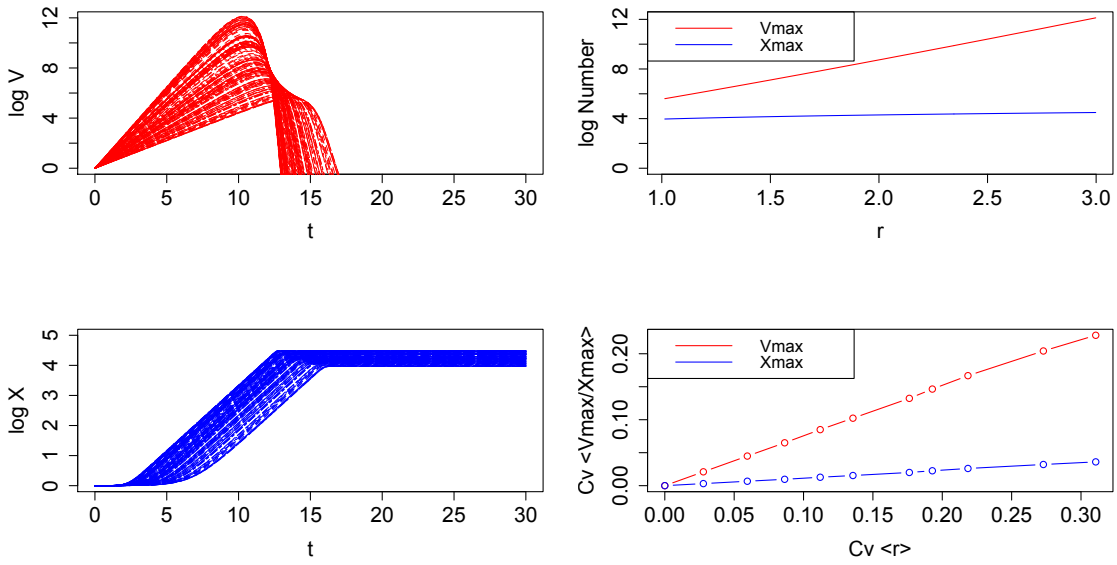

Figure 5: The effects of heterogeneity in the pathogen growth rate on the dynamics of infection in immunity. Model simulations where the parameter r was varied over the range indicated in Table 1, and $\log_{10} V_{max}$ and $\log_{10} X_{max}$ were calculated from each simulation. It can be seen that increasing r increases both V_{max} and X_{max}, and increasing $C_v < r >$ increases both $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$. However $C_v < \log_{10} V_{max} >$ has a higher magnitude than $C_v < \log_{10} X_{max} >$.

We see that as r increases both $\log_{10} X_{max}$ and $\log_{10} V_{max}$ increase as well. The magnitude of increase for $\log_{10} X_{max}$ is smaller compared to the increase in V_{max} which goes from approximately 10^6 to 10^{12}. We can also see that as $C_v < r >$ increases $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase, but the magnitude of increase for $C_v < \log_{10} V_{max} >$ is much greater than $C_v < \log_{10} X_{max} >$. $C_v < \log_{10} V_{max} >$ increases to 0.20 and $C_v < \log_{10} X_{max} >$ increases to 0.030.

4.2 Variation in Immune Growth Rate: s

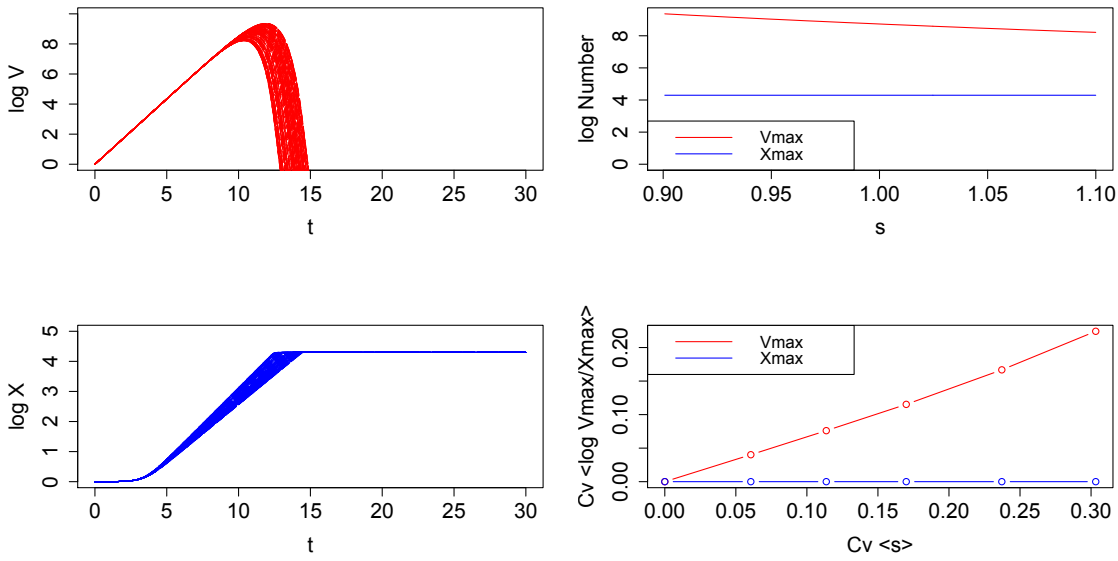

Figure 6: Heterogeneity in the maximum immune cell growth rate and how it impacts the dynamics of infection and immunity. Model simulations were run for the specified range in Table 1, and model outputs were calculated from each simulation. It can be seen that heterogeneity in s causes significant variation in $\log_{10} V_{max}$ but no changes in $\log_{10} X_{max}$. This is further supported by the differences in magnitude of $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$.

The results indicate that as s increases, $\log_{10} V_{max}$ decreases very little in magnitude and $\log_{10} X_{max}$ does not change. As $C_v < s >$ increases $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase. The magnitude of $C_v < \log_{10} V_{max} >$ increases to 0.20 and the magnitude of $C_v < \log_{10} X_{max} >$ increases to $1.5 * 10^{-6}$ which is extremely small.

4.3 Variation in the Pathogen Density at which Immunity Responds: ϕ

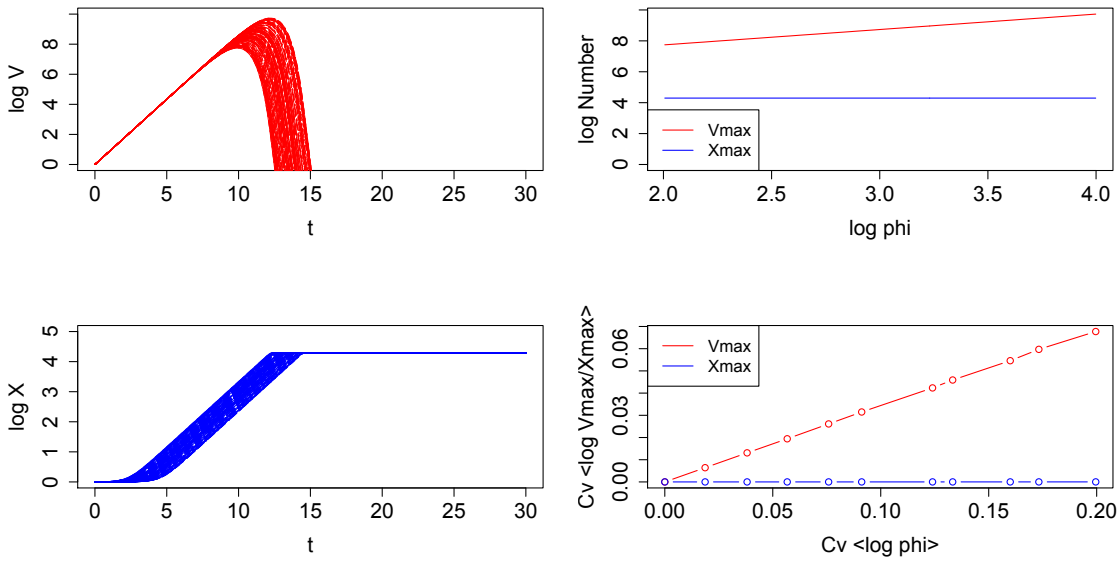

Figure 7: Heterogeneity in density required for half the maximum rate of immune cell growth and the effects of this variation upon the dynamics of infection and immunity within the host. Variation in ϕ only increases $\log_{10} V_{max}$ and does not change $\log_{10} X_{max}$. Increasing $C_v < \log_{10} \phi >$ generates very little variation in $\log_{10} V_{max}$ and yields no variation in $\log_{10} X_{max}$. This can be seen in the magnitude of $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ which are approximately 0.06 and 0 respectively.

The results show that as $log_{10}\phi$ increases $\log_{10} V_{max}$ increases and $\log_{10} X_{max}$ does not change. $\log_{10} V_{max}$ increases by 100 fold. We can also see that as $C_v < \log_{10} \phi >$ increases both $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase. The magnitude of change in $C_v < \log_{10} V_{max} >$ is 0.06. This change is greater than the change in $C_v < \log_{10} X_{max} >$, which barely increases to $1.5 * 10^{-6}$.

4.4 Variation in the Rate of Killing of Pathogen by Immunity: k

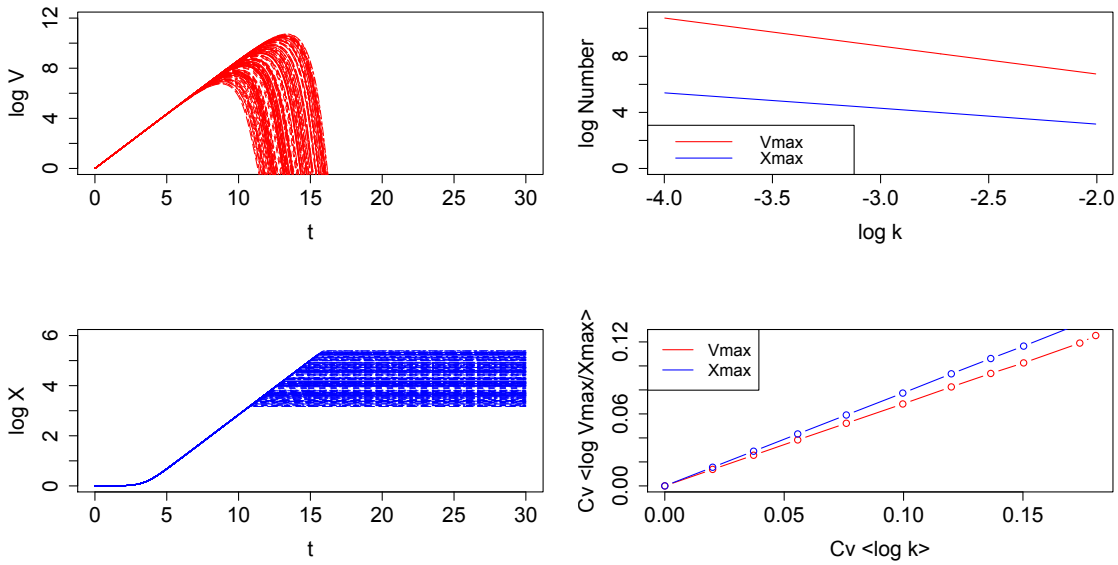

Figure 8: Variation in the mass action killing rate of adaptive immunity and heterogeneity in the dynamics of infection within a host. Increasing $\log_{10} k$ decreases $\log_{10} V_{max}$ and $\log_{10} X_{max}$, and increasing $C_v < \log_{10} k >$ yields the approximately similar amounts of variation in both $\log_{10} V_{max}$ and $\log_{10} X_{max}$. This can be seen in how changing $C_v < \log_{10} k >$ increases both $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ similarly.

As $\log_{10} k$ is increased both $\log_{10} V_{max}$ and $\log_{10} X_{max}$ decrease. The magnitude of decrease for $\log_{10} V_{max}$ is approximately 10^4 fold and the magnitude of decrease in $\log_{10} X_{max}$ is approximately 10^2 fold. It can be seen that $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase as $C_v < \log_{10} k >$ increases. $C_{V_{max}}$ increases to 0.12 and $C_{X_{max}}$ increases to 0.15.

4.5 Variation in the Initial Viral Load: $V(0)$

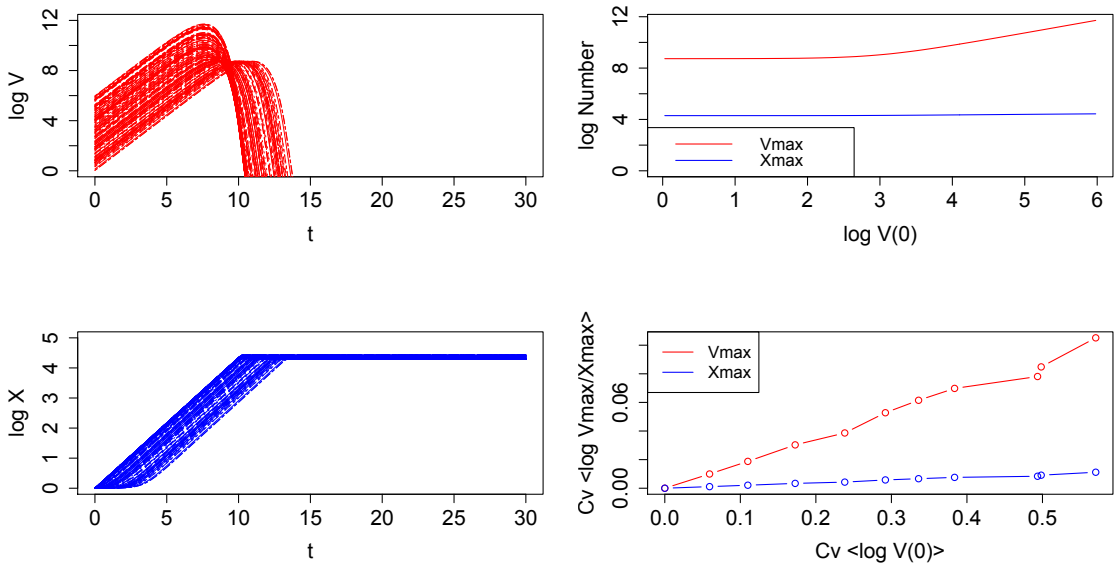

Figure 9: Heterogeneity in the initial infectious dose causes changes in the dynamics of infection within the host. As $\log_{10} V(0)$ increases $\log_{10} V_{max}$ increases. However $\log_{10} X_{max}$ does not increase with increasing $\log_{10} V(0)$. Furthermore as $C_v < \log_{10} V(0) >$ increases, $C_v < \log_{10} V_{max} >$ increases much more than $C_v < \log_{10} X_{max} >$.

We see that as $\log_{10} V(0)$ increases there is no change in $\log_{10} V_{max}$ until $\log_{10} V(0) > 3$. Once $\log_{10} V(0) > 3$, $\log_{10} V_{max}$ increases as $\log_{10} V(0)$ increases. As $V(0)$ increases there is no change in $\log_{10} X_{max}$. As $C_v < \log_{10} V(0) >$ increases both $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase as well. $C_v < \log_{10} V_{max} >$ increases to 0.10, and $C_v < \log_{10} X_{max} >$ increases to 0.01.

4.6 Variation in Initial Immunity: X(0)

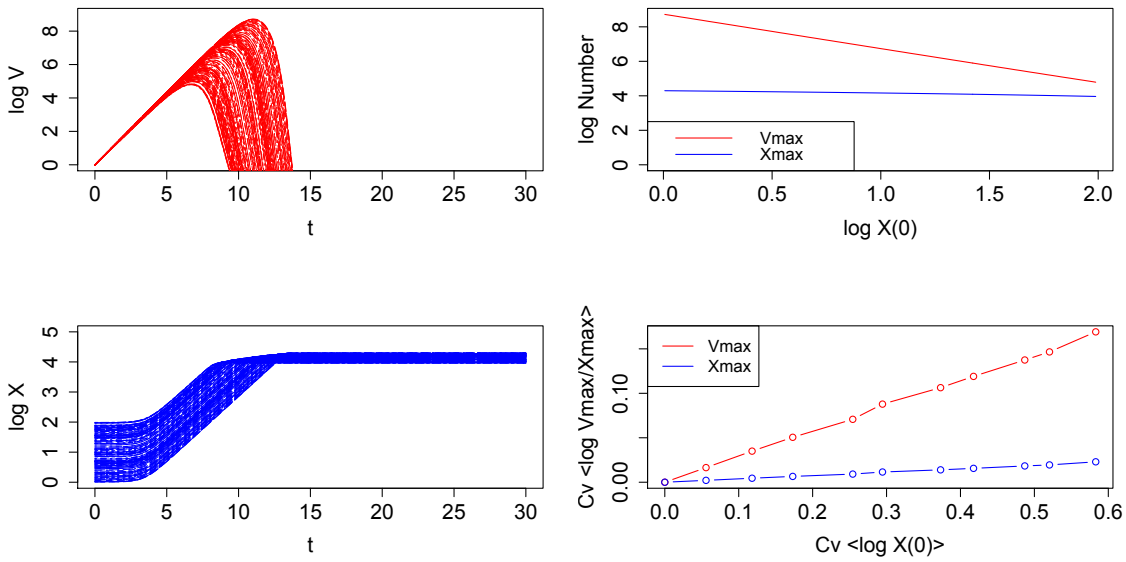

Figure 10: Differences in the initial immune cell density specific to the pathogen elicits variation in the intrahost dynamics of infection. Heterogeneity in $\log_{10} X(0)$ causes more variation in $\log_{10} V_{max}$ but less variation in $\log_{10} X_{max}$. As $\log_{10} X(0)$ increases there is a reduction in $\log_{10} V_{max}$ but no change in $\log_{10} X_{max}$. It can also be seen that as $C_v < \log_{10} X(0) >$ increases $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ both increase as well.

We see that by increasing $\log_{10} X(0)$, $\log_{10} V_{max}$ decreases and $\log_{10} X_{max}$ has no change. The magnitude of change in $\log_{10} V_{max}$ is about 10^4. As $C_v < X(0) >$ increases both $C_v < \log_{10} V_{max} >$ and $C_v < \log_{10} X_{max} >$ increase. The magnitude of change in $C_v < \log_{10} V_{max} >$ is 0.15 and the magnitude of change in $C_v < \log_{10} X_{max} >$ is 0.020.

4.7 Summarization of the Results

Parameter (units)	$C_v < \log V_{max} >$	$C_v < \log X_{max} >$	$\frac{C_v<\log V_{max}>}{C_v<\log X_{max}>}$
r, $days^{-1}$	0.20	0.030	6.67
s, $days^{-1}$	0.20	1.5e-6	1.33e5
ϕ, V	0.06	1.2e-5	5e3
k, $(X * days)^{-1}$	0.12	0.15	0.8
$V(0)$, V	0.10	0.01	10
$X(0)$, X	0.15	0.025	6

Table 2: A summarization of the results: Parameters and their coefficients of variation for V_{max} and X_{max}. The range of the coefficients of variation for both $\log V_{max}$ and $\log X_{max}$ were measured for each parameter, and they are shown in Table 2 with the corresponding parameter. These results are also displayed in Figure 11 which best encapsulates the relative amount of variation present in peak viral load and peak immune cell density caused by heterogeneity in each parameter. It can also be seen that none of the ratios of the coefficient of variation of V_{max} to the coefficient of variation of X_{max} are equal to 100 for each parameter. This also shows that our model output does not recapitulate the data

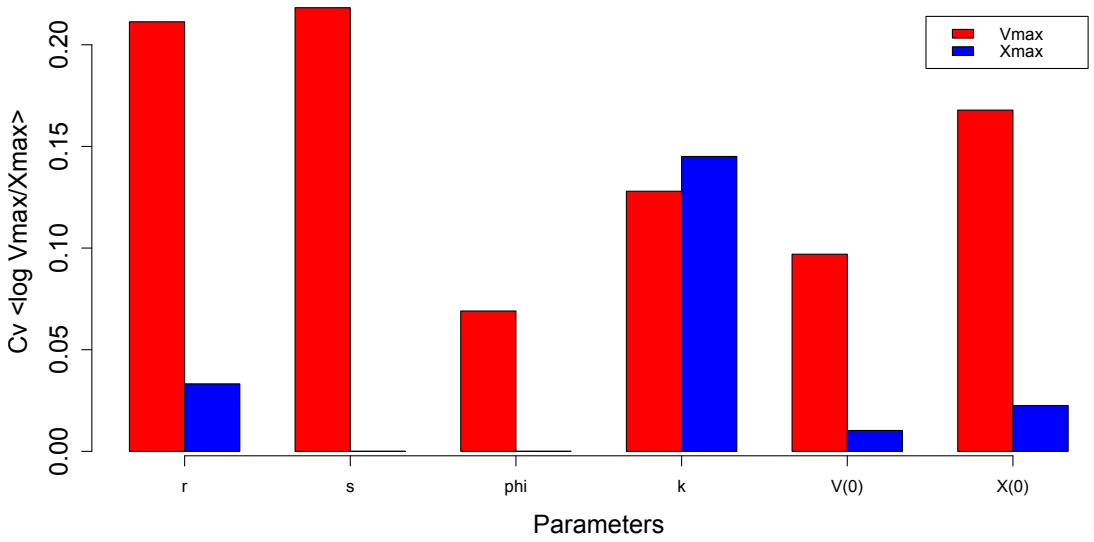

Figure 11: Variation in all parameters generate more heterogeneity in the peak viral load than the peak immune cell density. The coefficients of variation for both $\log V_{max}$ and $\log X_{max}$ for each parameter are shown above in red and blue bars respectively. It can be seen that the amount of variation in $\log V_{max}$ is greater than the amount of variation present in $\log X_{max}$ for all parameters, and that only k causes similar changes in both $\log V_{max}$ and $\log X_{max}$.

5 Discussion

We will first discuss the consequences of variation in each of the model parameters on the variation in the peak viral load and the maximum immune cell density, and the possible mechanisms that this variation could be caused. Then we will discuss the limitations of our model and future directions.

5.1 Viral Growth Rate

We have found that varying the viral growth rate causes changes in the peak viral load and the peak immune cell density. By changing the growth rate, a pathogen is able to grow either more or less efficiently in its host. Increases

in r increase V_{max} because the pathogen can grow more quickly before it can be controlled and cleared by the immune system. On the other hand a higher density of immune cells is needed to clear a faster growing pathogen which corresponds to the increase in X_{max} after r has been increased.

We can approximate how varying r affects X_{max} looking at the steady state equation. The steady state equation is calculated by setting $\dot{V} = 0$. Solving for X yields the equation $X = \frac{r}{k}$. It can be seen that the density of immune cells needed to clear the infection, X_{max}, is proportional to the growth rate divided by the mass action killing rate of the immune system, k. This equation shows why increases in r increase X_{max}.

It has been shown that single nucleotide polymorphisms (SNPs) in the cytokine-inducible SRC homology 2 domain (CISH) are associated with susceptibility to tuberculosis (TB) in the Chinese Han population and in the global population [25]. It could be that variation in this domain causes cytokine production to be different between individuals, which could be reflected in the model as a higher pathogen growth rate. Deficiencies in type 2 helper T cell cytokine secretions cause orthopoxvirus infection to be worse for the host [26]. Furthermore studies have shown that racial and gender differences in natural killer p46, a natural cytotoxicity receptor, were associated with differences in immunity towards hepatitis C. It was found that NKp46 expression was correlated with more antiviral activity [27]. Since our model does not consider the effects of the innate system this could also be reflected as differences in the pathogen growth rate.

5.2 Immune Growth Rate

Changes in s reflect changes in the maximum growth rate of immunity. This means that the immune system either grows more or less rapidly after stimulation by a given pathogen. Increasing s decreases V_{max} because a faster growing immune system reaches the density required to control and clear the pathogen more quickly. It does not affect X_{max} because the number of immune cells needed to control the pathogen has not changed. As indicated by the steady state equation above, the clearing density is dependent on r and k.

Recent studies have shown the fruits of black pepper plants contain a compound known as piperine, and piperine is an inhibitor of interleukin 2 driven T cell proliferation [28]. It could be possible that differences in host diet could affect adaptive immune cell dynamics. The effects of piperine

on the immune system could be represented in the model as a decrease in the immune cell growth rate. It has also been shown that variation in the HLA-DRB1, a class II HLA gene, is associated with resistance to typhoid fever [29]. This could be represented in our model as a faster immune cell growth rate. More diverse antigen presentation could allow for a mounting of different immune responses, and allows for faster clearance of the pathogen because the max immune cell density is reached more quickly.

5.3 Pathogen Density at which Immunity Responds

By increasing ϕ, the density at which the immune system responds increases. Increases in ϕ mean that the immune system will respond at higher densities, which in this case means a later time. As a result the pathogen has more time to grow and increase V_{max} before it is brought under control. X_{max} does not change because the clearing density of the immune system is mostly dependent on the killing rate which has not changed.

Increases in ϕ might reflect lower affinity of the adaptive immune cell receptor for the antigen. If the affinity for the antigen is lower, more antigen may need to be present before an adaptive response is formed.

Changes in ϕ might also be caused by the ability of the pathogen to avoid the adaptive immune system. This could be done by somehow inhibiting the action of cytokines that stimulate the adaptive immune system. Some pathogens are known to secrete soluble receptors for cytokines to prevent the inhibitory effects of these cytokines on replication.

It been shown that influenza A viruses have a protein called PA-X that allows for inhibition of the host immune response [30]. A given pathogen may also have different strategies or mechanisms to avoid or suppress different aspects of the immune system, and this could be reflected as a higher density needed for the maximum rate of immune cell proliferation. By suppressing the immune system, the pathogen has more time to grow and therefore can reach a higher density before the immune system responds. HIV has also been shown to block the induction of interferon genes when the virus infects macrophage and dendritic cells by inhibiting the kinase activity of TBK1 [31].

5.4 Killing Rate by Adaptive Immunity

An individual whose immune system has a higher k has an immune system that kills at a faster rate compared to a lower k. An immune system that kills more efficiently decreases V_{max} because the immune system is able to kill more of the pathogen in a given amount of time. If the immune system is better at killing the pathogen then X_{max} is decreased because the amount of immune cells needed to control the infection is lower as well.

We assumed that the density of immune cells needed to clear the pathogen was very high, and so k was assumed to be at a very low value. It has been shown that the tyrosine kinase Lyn is a regulator of survival for plasma cells and that it limits plasma cell accumulation [32]. Plasma cells are B cells that have been activated, moved to the bone marrow, and become antibody secreting cells. Host differences in genes that affect the survivability of plasma cells and effector cells could change the killing rate.

It has also been shown that different HLA alleles can greatly affect the amount of liver damage from a chronic hepatitis C infection [33]. In this study, the researchers found one allele that had a protective effect and another that predisposed the host for severe liver damage. Differences in the host HLA alleles could also be reflected in the killing rate where protective alleles have a higher killing rate and vice versa.

Recent studies also showed that homozygosity in the HLA class I genes resulted in faster disease progression in HIV infected humans, and the results suggest that diversitiy and sequence specificity can affect the dynamics of disease progression [34]. In our model this could be represented as a decrease in k which would allow the pathogen to grow more quickly despite the presence of immunity.

5.5 Initial Viral Inoculum

Increasing the initial infectious dose increases V_{max} because the starting amount of pathogen in the host is higher. As a result the peak viral load will be higher compared to a host infected with a smaller infectious dose. However the maximum amount of immunity will not change because the killing rate of the pathogen by the adaptive immune system has not changed, and as a result the same density of immune cells is needed to clear the pathogen from the host.

A decrease in the proviral load of human T cell lymphotropic virus was

shown to be associated with heterozygosity in the HLA class I alleles. The researchers found that individuals who were heterozygous at all three HLA class I loci had a lower proviral load, and they were thought to have a reduced risk of disease [35]. This could be represented in our model as changes in the initial infectious dose. Studies have also shown that MHC diversity also results in resistance against coinfection and multiple-strain infections [36, 37].

5.6 Initial Immunity

Increasing the initial immunity present decreases V_{max} because more immune cells that are specific to the pathogen are already present. When the immune cells are in the presence of the pathogen, they clonally expand but if more immune cells are present then the density at which the immune cells control and clear the infection is reached earlier. On the other hand the density at which the immune system controls and clears the infection is not changed because the immune cells are still killing the pathogen at a rate of k.

The amount of initial immune cells specific for a given pathogen could be increased due to previous antigenic history or vaccination. Both of these processes form memory cells specific for the given pathogen. However heterozygosity in the HLA genes could also increase the diversity of antigens presented and allow for better clinical outcomes in hepatitis B infections [38]. This diversity could allow for adaptive immune cells specific for different epitopes on a given pathogen to respond, which could be reflected in our model as a change in the initial density of immune cells specific to the pathogen.

5.7 Summary: The Questions Revisited

Earlier we mentioned the questions we wanted to ask using our simple models. We wanted to ask what are the key parameters that influence the dynamics of infection and immunity? Which model parameters do we expect to vary? Why is there more variation in peak viremia compared to peak immunity? What is the effect of variation in each parameter? Does our model recapitulate the yellow fever virus vaccination data?

5.7.1 What are the key parameters?

The key parameters are $X(0)$, $V(0)$, and k. When these parameters are varied over the ranges in Table 1 and all others are held constant, there is more variation present in V_{max} compared to X_{max}. Heterogenety in the parameter k, the mass action killing rate, generates more variation in X_{max} compared to V_{max} which is the opposite trend of the data in Figure 2. Lastly varying the parameters s and ϕ do not generate any heterogeneity in X_{max} and only generate variation in V_{max}. This indicates that these two parameters are not key in generating the heterogeneity observed in the yellow fever imunization data.

5.7.2 Which parameters do we expect to vary?

We expect the parameters r, s, ϕ, and k to vary. We do not expect $V(0)$ and $X(0)$ to vary. The initial viral inoculum, $V(0)$, should not vary because viral load is controlled. All individuals received the same viral dose from the vaccine, and so we do not expect $V(0)$ to vary in the context of our data. However in a more general setting, we would expect $V(0)$ to vary. We also do not expect $X(0)$ to vary because these vaccinations were given to naive individuals. These were not booster shots that were given to the individuals, and so none of the individuals in the studies had any previous antigenic history with yellow fever virus. Therefore we would not expect $X(0)$ to vary between individuals in the yellow fever vaccination data. However, more generally, we could expect $X(0)$ to vary between individuals in the natural human population because of differing immunization and infections history. This is because some individuals may have been previously infected or had immunization to a given pathogen, and if different individuals get infected/vaccinated at different times then their population of cells specific for that given pathogen will be different upon subsequent infection as well.

5.7.3 Why is there more variation in peak viral load compared to peak immune cell density?

There is more variation present in the peak viral load compared to the peak immune cell density in our model because of the number of parameters that affect V_{max} and X_{max}. The steady state equation for virus, $\dot{V} = rV - kVX$, once solved yields the equation $X = \frac{r}{k}$. The density at which the immune system is able to stop the virus from growing and clear it from the host, also

X_{max}, is only dependent on two parameters, r and k. On the other hand V_{max} is affected by all model parameters. It can be seen in our results that varying any one parameter while holding all others constant will cause V_{max} to be changed. Since X_{max} is dependent on two parameters while V_{max} is dependent on six parameters, there is more heterogeneity in V_{max} compared to X_{max}.

5.7.4 What is the effect of variation in each parameter?

There is a greater amount of variation, due to the heterogeneties considered, present in V_{max} compared to X_{max} as seen in Figure 11. We have found that heterogeneity in parameters r and k generate the greatest amount of variation in X_{max}, and that varying k has a significantly greater effect upon the variation in X_{max}. We also found that varying parameters r and s generates the most amount of variation in V_{max} of the variables considered. Furthermore, we can see in Figure 11 that heterogeneity in parameters s and ϕ does not result in any variation in X_{max}. Lastly, variation in $V(0)$ and $X(0)$ generate the same magnitude of heterogeneity in X_{max}, but variation in $X(0)$ generates more variation in V_{max} compared to variation in $V(0)$.

5.7.5 Does our model recapitulate the yellow fever immunization data?

Our model output does not recapitulate the yellow fever vaccination data. Using the ranges indicated in Table 1, we varied each model parameter while holding all others constant. When we use the coefficient of variation C_v, we are measuring the relative amount of variation. After we varied a parameter, we looked for a 100 fold difference in the peak viral load and peak immune cell density. However, none of our model outputs recapitulated the data, and we did not find a 100 fold difference in V_{max} and X_{max}.
We cannot tease out whether this is a model limitation or whether the ranges we have used for our simulations are wrong. When we vary r in Figure 5, it can be seen that between the values of 1 and 2 there is a 100 fold difference in V_{max} and X_{max}. Since all individuals in the yellow fever vaccination studies all received the same pathogen, the heterogeneity in r must be due to variation in host factors. However we do not know which host factors cause this variation, and we would not be able to measure how variation in host factors affects the pathogen growth rate in vitro. As a result we cannot determine whether our

model does not recapitulate the yellow fever vaccination data due to model limitations, incorrect values for parameter ranges, or both.

5.8 Limitations and Future Directions

1. Our model is the simplest possible case. In the future we will add further complexity and realism to the model by introducing the following factors, and analyzing how variation with the new parameters could affect the dynamics of infection.

    (a) Resource limitation could be modeled in our equation by introducing a carrying capacity.

    (b) Host Lethal Density could be introduced into our models by ending simulations when a certain threshold for pathogen density has been crossed.

    (c) Innate Immunity could be introduced with a new set of equations that model macrophage response.

    (d) Immune Cell Exhaustion would be introduced in the form of a new parameter that represents the rate at which adaptive immune cells inactivate or die.

    (e) Stochasticity would be introduced to the model by using stochastic tau-leaping algorithm to model the course of infection [39].

2. Our analysis was limited to varying a single parameter while keeping all others constant. Future directions include:

    (a) Multiparameter Variation - More than one parameter could be varied while all others are held constant.

    (b) Introducing probability density functions - Parameters could be assumed to have a prior probability density function with its own variance and mean, and how the variance of this distribution affects the dynamics of infection could be explored.

    (c) Parameter Correlation - Certain parameters could have biological correlation, and the correlation between these parameters and how it affects the dynamics of infection could be analyzed in the future as well

6 References

[1] Reprinted from *Immunity*, 8, Murali-Krishna K, et al., Counting Antigen-Specific CD8 T Cells: A Reevaluation of Bystander Activation during Viral Infection. 177-187, Copyright(1998), with permission from Elsevier.
[2] Monath TP. (2005) Yellow fever vaccine. *Expert Rev. Vaccines* 4(4):553-574.
[3] Akondy RS, et al. (2015) Initial viral load determines the magnitude of the human CD8 T cell response to yellow fever vaccination. *PNAS*.
[4] Lee KJ, et al. (2000) NP and L Proteins of Lymphocytic Choriomeningitis Virus (LCMV) Are Sufficient for Efficient Transcription and Replication of LCMV Genomic RNA Analogs. In*J Virol* 74(8):3470-3477.
[5] Apanius V, et al. (1997) The Nature of Selection On The Major Histocompatibility Complex. *Crit Rev Immunol* 17(2):179-224.
[6] Collison M, et al. (2015) Homozygosity for HLA group 2 alleles predicts treatment failure with interferon-I's and ribavirin in chronic hepatitis C virus genotype 1 infection. *J Interferon Cytokine Res* 35(2):126-133.
[7] Blackwell JM. (1998) Genetics of host resistance and susceptibility to intra-macrophage pathogens: a study of multicase families of tuberculosis,leprosy, and leishmaniasis in north-eastern Brazil. *Int J Parasitol* 28(1):21-28.
[8] O'Connor SL, et al. (2010) MHC heterozygote advantage in simian immunodeficiency virus-infected Mauritian cynomolgus macaques. *Sci Transl Med* 2(22):22ra18.
[9] Hill AV. (1998) The immunogenetics of human infectious diseases. *Annu Rev Immunol* 16:593-617.
[10] Phillips AC, et al. (2008) Preliminary evidence that morning vaccination is associated with an enhanced antibody response in men. *Psychophysiology* 45:633-666.
[11] Edwards KM, et al. (2006) Acute Stress exposure prior to influenza vaccination enhances antibody response in women". *Brain, Behavior, and Immunity* 20:159-168.
[12] McClelland EE, Smith JM. (2011) Gender Specific Differences in the Immune Response to Infection. *Arch. Immunol. Ther. Exp.* 59:203-213.
[13] Gasper DJ, Tejera MM, Suresh M. (2014) CD4 T-cell Memory Generationand Maintenance. *Crit Rev Immunol* 34(2):121-146.
[14] Cizauskas CA, et al. (2014) Gastrointestinal helminths may affect host susceptibility to anthrax through seasonal immune trade-offs. *BMC Ecol*

14(27).
[15] Fisman D. (2012) Seasonality of viral infections: mechanisms and unknowns.*Clin Microbiol Infect* 18(10):946-954.
[16] Hanson DF. (1997) Fever, temperature, and the immune response. *Ann N Y Acad Sci* 453464.
[17] Shan J, et al. (2014) Temperature dependent bacteriophages of a tropical bacterial pathogen. *Front Microbiol* 14(5):599.
[18] Kapasi ZF, Mcrae ML, Ahmed R. (2005) Suppression of viral specific primary T-cell response following intense physical exercise in young but not old mice. *J Appl Physiol* 98:663-671.
[19] Iyer SS, et al. (2012) Protein Energy Malnutrition Impairs Homeostatic Proliferation of Memory CD8 T Cells. *J. Immunol* 188(1):77-84.
[20] Moore SE, et al. (1999) Prenatal or early postnatal events predict infectious deaths in young adulthood in rural Africa. *Int J Epidemiol* 28(6):1088-1095.
[21] Toth Z, et al. (2010) Epigenetic analysis of KSHV latent and lytic genomes.*PLoS Pathog* 6(7).
[22] Chen Y, et al. (2005) Population Fitness and the Regulation of Escherichiacoli Genes by Bacterial Viruses. *PLoS Biol* 3(7):e229.
[23] Kapasi ZF, Murali-Krishna K, McRae ML, Ahmed R. (2002) Defective generation but normal maintenance of memory T cells in old mice. *Eur. J. Immunol* 32:1567-1573.
[24] Antia R, Levin BR, May RM. (1994) Within-Host Population Dynamics and the Evolution And Maintenance of Microparasite Virulence. *American Naturalist* 144(3):457-472.
[25] Ji LD, et al. (2014) Polymorphisms in the CISH gene are associated with susceptibility to tuberculosis in the Chinese Han population." *Infect Genet Evol* 28:240-244.
[26] Sakala IG, et al. (2015) Deficiency in th2 cytokine responses exacerbate orthopoxvirus infection. *PloS One* 10(3).
[27] Golden-Mason L, Stone AE, Bambha KM, Cheng L, Rosen HR. (2012) Race-and gender-related variation in natural killer p46 expression associated with differential anti-hepatitis C virus immunity. *Hepatology* 56(4):1214-1222.
[28] Doucette CD, Greenshields AL, Liwski RS, Hoskin DW. (2015) Piperine blocks interleukin-2-driven cell cycle progression in CTLL-2 T lymphocytes by inhibiting multiple signal transduction pahtways. *ToxicolLett* 234(1):1-12.
[29] Dunstan SJ, et al. (2014) Variation at HLA-DRB1 is associated with

resistance to enteric fever. *Nat Genet.* 46(12):1333-1336.
[30] Hayashi T, MacDonald LA, Takimoto T. (2015) Influenza A virus protein PA-X contributes to viral growth and suppression of the host antiviral and immune responses." *J Virol.* [31] Harman AN, et al. (2015) HIV Blocks Interferon Induction in Human Dendritic cells and Macrophages by Dysregulation of TBK1. *J Virol.*
[32] Infantino S, et al. (2014) The tyrosine kinase Lyn limits the cytokine responsiveness of plasma cells to restrict their accumulation in mice. *SciSignal* 7(338).
[33] Marangon AV, et al. (2012) Protective effect of HLA-DRB1 11 and predisposition of HLA-C 04 in the development of severe liver damage in Brazilian patients with chronic hepatitis C virus infection. *Scand J Immunol* 76(4):440-447.
[34] Tang J, et al. (1999) HLA class I homozygosity accelerates disease progression in human immunodeficiency virus type 1 infection. *AIDS Res Hum Retroviruses* 15(4): 317-324.
[35] Jeffery KJ, et al. (2000) The influence of HLA class I alleles and heterozygosity on the outcome of human T cell lymphotropic virus type I infection. *J Immunol* 165(12): 7278-7284.
[36] Penn DJ, Damjanovich K, Potts WK. (2002) MHC heterozygosity confers a selective advantage against multiple-strain infections. *Proc Natl Acad Sci USA* 99(17):11260-11264.
[37] McClelland EE, Penn DJ, Potts WK. (2003) Major histocompatibility complex heterozygote superiority during coinfection.*Infect Immun* 71(4):2079-2086.
[38] Thursz MR, Thomas HC, Greenwood BM, Hill AV. (1997) Heterozygote advantage for HLA class-II type in hepatitis B virus infection. *Nat Genet.* 17(1):11-12.
[39] Cao Y, Li H, Petzold L. (2004) Efficient formulation of the stochastic algorithm for chemically reacting systems". *J Chem Phys* 121: 4059-4067.

7 Appendix

7.1 Code

7.1.1 Variation in r

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/Antia-Jonathan/
>       #WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> set.seed()
> ODE_Ebola <- function(t, y, parms)
+   {
+   with(as.list(c(y,parms)),
+       # allows the parameter file parms
+                 #to be as a list
+ {     # and the state variables to be the
+     #same as chosen in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))                #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
```

```
+   phi = 1e3,  # parasite density at
+               # which immunity responds
+   k = 1e-3,   # killing of virus by immunity
+   p = 1e3,      # cytokine production rate
+   phi_c = 1e3,  # cytokine production begins
+              # when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0, maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola, y=initial.conditions,
+                          times=time.points,
+                           parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]+soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red", type="l",
+      ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time, col = 'blue', type = 'l')
> lines(soln$C ~ soln$time, col = 'green', type = 'l')
> plot(log10(soln$V)  ~ soln$time,col="red", type="l",
+      ylim=c(1,log10(ymax)),
+      ylab="log number",xlab="time (days)")
> lines(log10(soln$X) ~ soln$time, col = 'blue', type = 'l')
> lines(log10(soln$C) ~ soln$time, col = 'green', type = 'l')
> legend("topright", c("Virus", "Adaptive", "Cytokines"),
+        lty = 1, lwd = 2, col=c("red","blue","green"),
+        cex = 2)
> ######################################
> ##create these empty vectors so
```

```
> #that the results can be stored later
> ####################################
> ##vector for duration of infection
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ################################################
> #integrating function
> ################################################
> Tfunction <- function(x, y)
+ ################################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the
+   ##difference between the times
+   #rollmean finds the average
+   ##between the two numbers in y
+   #this method leaves out one number
+   ##(the last number in the list)
+   #so it underestimates transmission
+   sum(diff(x)*rollmean(y,2))
+ }
> ################################################
> #simulation function
```

```
> ################################################
> Simulation <- function(initialconditions,params)
+ {
+   solution.of.ode = lsodar(ODE_Ebola,
+                            y=initialconditions,
+                            times=time.points,
+                            parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V)) - params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
+   return(y)
+ }
> ################################################
> #calculation function
> ################################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density as a m
+   ##easure of pathology
+   #we are measuring total pathology
+   ##and max parasitemia
+   ##find the maximum of the total
+   #pathogen and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
```

```
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density for duration of infection
+   #maximizing immunity is desired in antibiotic treatment
+   #might increase likelihood of generating memory
+   #we calculate the max and total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of cytokines
+   ##and total cytokine production
+   Cmax = max(x$C)
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(durationofinfection = durationofinfection,
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
```

```
+            V_sim = V_sim, X_sim = X_sim,
+            t_sim = t_sim)
+ }
> ############################################################
> a = Simulation(initial.conditions,parameters)
> b = Calculation(a)
> #averaging function
> ############################################################
> Averaging <- function(initialconditions,params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,params)
+     b = Calculation(a)
+     durationofinfectionvec[i] = b$durationofinfection
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] = b$haltingTransition
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection = mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
```

```
+   y = list(durationofinfection = durationofinfection,
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
+            numdeath = numdeath,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
> results = Averaging(initial.conditions,parameters,
+                     numsim)
> ######################################################
> ##defining functions
> #hold all other parameters constant, vary one
> ######################################################
> # averaging functions
> ######################################################
> # r averaging function
> ######################################################
> AveragingRange_r <- function(inputvec,
+                              initialconditions,
+                              params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     params["r"] = inputvec[i]
```

```
+     c = Averaging(initialconditions,params,1)
+     durationofinfectionvec[i] = c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(
+     durationofinfectionvec = durationofinfectionvec,
+            Vmaxvec = Vmaxvec,
+            total_Vvec = total_Vvec,
+            total_lnVvec = total_lnVvec,
+            Xmaxvec = Xmaxvec,
+            total_Xvec = total_Xvec,
+            Cmaxvec = Cmaxvec,
+            total_Cvec = total_Cvec,
+            numdeathvec = numdeathvec,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> ######################################################
> r_dist = runif(150,1,3)
> #rvec = sample(r_dist, 150)
>
> quartz()
> par(mfcol=c(2,2))
> #hist(r_dist,prob=TRUE, main = paste(""),
```

```
> #xlab = "r", cex = 2)
> #hist(rvec,prob=TRUE)
>
> #define the matrices before you begin
> ##the multiple simulation as the size
> #is completely dependent on the time points
> #and number of values in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(r_dist))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(r_dist))
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(r_dist))
> #leave it at 1 simulation for now
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations
> #in finding the averaging simulation values
> numsim = 1
> #solve our function for the sample data we have used
> results_r = AveragingRange_r(r_dist,
+                              initial.conditions,
+                              parameters,numsim)
> #plot multisimulation of V
>   for(i in 1:ncol(results_r$V_simmat))
+   {
+     if(i == 1)
+     {
+       plot(log10(results_r$V_simmat[,i]) ~
+              results_r$time_simmat[,i],
+            type = "l", col = "red", lty = i, lwd = 0.1,
+            ylab = "log V", xlab = "t", ylim = c(0,12),
+            cex.axis = 1.5, cex.lab = 1.5)
+     }
+     else
+     {
+       lines(log10(results_r$V_simmat[,i]) ~
+               results_r$time_simmat[,i],
```

```
+             col = "red", lty = i, lwd = 0.1,
+             ylab = "Log V", xlab = "t",
+             cex.axis = 1.5, cex.lab = 1.5)
+     }
+   }
>   #plot multsimulation of X
>   for(i in 1:ncol(results_r$X_simmat))
+   {
+     if(i == 1)
+     {
+       plot(log10(results_r$X_simmat[,i]) ~
+              results_r$time_simmat[,i],
+            type = "l", col = "blue", lty = i,
+            lwd = 0.1, xlab = "t",
+            ylab = "log X", ylim = c(0,5),
+            cex.axis = 1.5, cex.lab = 1.5)
+     }
+     else
+     {
+       lines(log10(results_r$X_simmat[,i]) ~
+               results_r$time_simmat[,i],
+             type = "l", col = "blue",
+             lty = i, lwd = 0.1,
+             xlab = "t", ylab = "log X",
+             cex.axis = 1.5, cex.lab = 1.5)
+     }
+   }
>   #plot how our values of interest
> #change with changing r
>   #sort command orders the values in ascending order
> #so that they can be plotted as lines
>   plot(log10(sort(results_r$Vmax)) ~ sort(r_dist),
+        type = "l",col = "red", lty = 1,
+        xlab = "r", ylab = "log Number",
+        ylim = c(0, max(log10(results_r$Vmax))),
+        cex.axis = 1.5, cex.lab = 1.5)
>   lines(log10(sort(results_r$Xmax)) ~ sort(r_dist),
+         type = "l", col = "blue", lty = 1, xlab = "r",
```

```
+         cex.axis = 1.5, cex.lab = 1.5)
>   ##add a legend
>   legend("topleft", c("Vmax", "Xmax"), lty = 1,
+          lwd = 1, col=c("red","blue"), cex = 1.2)
> #create a vector for one of the shape parameters
> #as a increases, the variance
> #of the distribution decreases
> #r_unif = runif(1e3, min = 2, max = 2)
> #for a uniform distribution
> #the mean is the median (max - min)/2
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
> r_varvec = NULL
> r_meanvec = NULL
> r_Vmax_varvec = NULL
> r_Vmax_meanvec = NULL
> r_totalV_varvec = NULL
> r_Xmax_varvec = NULL
> r_Xmax_meanvec = NULL
> rangevec = seq(0,1,0.1)
> for(i in 1:length(rangevec))
+ {
+ rvec = runif(100,
+              min = (2-rangevec[i]),
+              max = (2+rangevec[i]))
+
+ #define the matrices before you
+ #begin the multiple simulation as the size
+ #is completely dependent on the time points
+ #and number of values in the input vector
+ V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(rvec))
+ X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(rvec))
+ time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(rvec))
+
+ r_varvec[i] = var(rvec)
```

```
+ r_meanvec[i] = mean(rvec)
+
+ results2_r = AveragingRange_r(rvec,
+                                initial.conditions,
+                                parameters,
+                                numsim)
+
+ r_Vmax_varvec[i] = var(log10(results2_r$Vmax))
+ r_Vmax_meanvec[i] = mean(log10(results2_r$Vmax))
+ r_totalV_varvec[i] = var(results2_r$total_V)
+ r_Xmax_varvec[i] = var(log10(results2_r$Xmax))
+ r_Xmax_meanvec[i] = mean(log10(results2_r$Xmax))
+ }
> r_cvvec = sqrt(r_varvec)/r_meanvec
> r_Vmax_cvvec = sqrt(r_Vmax_varvec)/r_Vmax_meanvec
> r_Xmax_cvvec = sqrt(r_Xmax_varvec)/r_Xmax_meanvec
> plot(r_Vmax_cvvec ~ r_cvvec,
+      type = "b", col = "red",
+      lty = 1, xlab = "Cv <r>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim = c(0,max(r_Vmax_cvvec)),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(r_Xmax_cvvec ~ r_cvvec,
+       type = "b", col = "blue",
+       lty = 1, xlab = "Cv <r>",
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max(r_Xmax_cvvec)),
+       cex.axis = 1.5, cex.lab = 1.5)
> legend("topleft", c("Vmax", "Xmax"), lty = 1,
+        lwd = 1, col=c("red","blue"), cex = 1.2)
```

7.1.2 Variation in s

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/
```

```
> #Antia-Jonathan/WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> set.seed()
> ODE_Ebola <- function(t, y, parms)
+ {
+   with(as.list(c(y,parms)),
+        # allows the parameter
+        #file parms to be as a list
+ {      # and the state variables to be the
+       #same as chosen in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))                #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
+   phi = 1e3,  # parasite density at
+               #which immunity responds
+   k = 1e-3,   # killing of virus by immunity
+   p = 1e3,      # cytokine production rate
+   phi_c = 1e3,  # cytokine production begins
+                 #when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0,
```

```
+                   maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola,
+                          y=initial.conditions,
+                          times=time.points,
+                          parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]
+                       +soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red",
+      type="l",ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time,
+       col = 'blue', type = 'l')
> lines(soln$C ~ soln$time,
+       col = 'green', type = 'l')
> plot(log10(soln$V)  ~ soln$time,col="red",
+      type="l",ylim=c(1,log10(ymax)),
+      ylab="log number",xlab="time (days)")
> lines(log10(soln$X) ~ soln$time,
+       col = 'blue', type = 'l')
> lines(log10(soln$C) ~ soln$time,
+       col = 'green', type = 'l')
> legend("topright", c("Virus", "Adaptive",
+                      "Cytokines"),
+        lty = 1, lwd = 2,
+        col=c("red","blue","green"), cex = 2)
> ############################################################
> ##create these empty vectors so
> #that the results can be stored later
> ############################################################
> ##vector for duration of infection
```

```
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ##############################################
> #integrating function
> ##############################################
> Tfunction <- function(x, y)
+   ##########################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the
+   ##difference between the times
+   #rollmean finds the average
+   ##between the two numbers in y
+   #this method leaves out one number
+   ##(the last number in the list)
+   #so it underestimates transmission
+   sum(diff(x)*rollmean(y,2))
+ }
> ##############################################
> #simulation function
> ##############################################
> Simulation <- function(initialconditions,
+                        params)
```

```
+ {
+ solution.of.ode = lsodar(ODE_Ebola,
+                          y=initialconditions,
+                          times=time.points,
+                           parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V))-params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
+   return(y)
+ }
> ##############################################
> #calculation function
> ##############################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density as a
+   #measure of pathology
+   ##we are measuring total
+   #pathology and max parasitemia
+   ##find the maximum of the
+   #total pathogen and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
```

```
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density
+   ##for duration of infection
+   #maximizing immunity is
+   ##desired in antibiotic treatment
+   #might increase likelihood
+   ##of generating memory
+   #we calculate the max
+   #and total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of cytokines
+   #and total cytokine production
+   Cmax = max(x$C)
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(
+     durationofinfection = durationofinfection,
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
```

```
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
+            V_sim = V_sim, X_sim = X_sim,
+            t_sim = t_sim)
+ }
> ################################################
> a = Simulation(initial.conditions,
+                parameters)
> b = Calculation(a)
> #averaging function
> ################################################
> Averaging <- function(initialconditions,
+                       params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,params)
+     b = Calculation(a)
+ durationofinfectionvec[i] = b$durationofinfection
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] = b$haltingTransition
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection = mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
```

```
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
+   y = list(
+     durationofinfection = durationofinfection,
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
+            numdeath = numdeath,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
> results = Averaging(initial.conditions,
+                     parameters,
+                     numsim)
> ################################################
> ##defining functions
> #hold all other parameters constant, vary one
> ################################################
> # averaging functions
> ################################################
> # r averaging function
> ################################################
```

```
> AveragingRange_s <- function(inputvec,
+                              initialconditions,
+                              params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     params["s"] = inputvec[i]
+     c = Averaging(initialconditions,params,1)
+     durationofinfectionvec[i] = c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(
+     durationofinfectionvec = durationofinfectionvec,
+            Vmaxvec = Vmaxvec,
+            total_Vvec = total_Vvec,
+            total_lnVvec = total_lnVvec,
+            Xmaxvec = Xmaxvec,
+            total_Xvec = total_Xvec,
+            Cmaxvec = Cmaxvec,
+            total_Cvec = total_Cvec,
+            numdeathvec = numdeathvec,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
```

```
> ######################################################
> smean = 1
> s_dist = runif(150,0.9,1.1)
> quartz()
> par(mfcol=c(2,2))
> #hist(s_dist,prob=TRUE,
> #main = paste(""), xlab = "s", cex = 2)
> #hist(svec,prob=TRUE)
>
> #define the matrices before you
> #begin the multiple simulation as the size
> #is completely dependent on the time points
> #and number of values in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(s_dist))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(s_dist))
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(s_dist))
> #leave it at 1 simulation for now
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations in
> #finding the averaging simulation values
> numsim = 1
> #solve our function for the sample data we have used
> results_s = AveragingRange_s(s_dist,
+                              initial.conditions,
+                              parameters,numsim)
> #plot multisimulation of V
> for(i in 1:ncol(results_s$V_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_s$V_simmat[,i])
+          ~ results_s$time_simmat[,i],
+          type = "l", col = "red", lty = i,
+          lwd = 0.2, ylab = "log V", xlab = "t",
```

```
+          ylim = c(0,10), cex.axis = 1.5,
+          cex.lab = 1.5)
+   }
+   else
+   {
+     lines(log10(results_s$V_simmat[,i])
+           ~ results_s$time_simmat[,i],
+           col = "red", lty = i, lwd = 0.2,
+           ylab = "Log V", xlab = "t",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot multsimulation of X
> for(i in 1:ncol(results_s$X_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_s$X_simmat[,i])
+          ~ results_s$time_simmat[,i],
+          type = "l", col = "blue",
+          lty = i, lwd = 0.2,
+          xlab = "t", ylab = "log X",
+          ylim = c(0,5),
+          cex.axis = 1.5, cex.lab = 1.5)
+   }
+   else
+   {
+     lines(log10(results_s$X_simmat[,i])
+           ~ results_s$time_simmat[,i],
+           type = "l", col = "blue",
+           lty = i, lwd = 0.2,
+           xlab = "t", ylab = "log X",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot how our values of interest
> ##change with changing r
> #sort command orders the
```

```
> ##values in ascending order
> #so that they can be plotted as lines
> plot(log10(sort(results_s$Vmax,
+                 decreasing = TRUE))
+                 ~ sort(s_dist),
+      type = "l",col = "red", lty = 1,
+      xlab = "s", ylab = "log Number",
+      ylim = c(0,max(log10(results_s$Vmax))),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(log10(sort(results_s$Xmax)) ~ sort(s_dist),
+       type = "l", col = "blue", lty = 1,
+       cex.axis = 1.5, cex.lab = 1.5)
> #add a legend
> legend("bottomleft", c("Vmax", "Xmax"),
+        lty = 1,  lwd = 1,
+        col=c("red","blue"), cex = 1.2)
> #mean = a/(a+b) for beta distribution
> #variance = a*b/((a+b)^2(a+b+1))
>
> #as a increases, the variance of the
> #distribution decreases
> s_unif = runif(1e3, min = smean, max = smean)
> #for a uniform distribution
> #the mean is the median (max - min)/2
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
> s_varvec = NULL
> s_meanvec = NULL
> s_Vmax_varvec = NULL
> s_Vmax_meanvec = NULL
> s_totalV_varvec = NULL
> s_Xmax_varvec = NULL
> s_Xmax_meanvec = NULL
> rangevec = seq(0,0.5,0.1)
> for(i in 1:length(rangevec))
+ {
+   svec = runif(150, min = (smean-rangevec[i]),
+                max = (smean+rangevec[i]))
```

```
+
+   #define the matrices before you
+   #begin the multiple simulation as the size
+   #is completely dependent on the time points
+   #and number of values in the input vector
+   V_simmat = matrix(nrow = length(time.points),
+                     ncol = length(svec))
+   X_simmat = matrix(nrow = length(time.points),
+                     ncol = length(svec))
+   time_simmat = matrix(nrow = length(time.points),
+                        ncol = length(svec))
+
+   s_varvec[i] = var(svec)
+   s_meanvec[i] = mean(svec)
+
+   results2_s = AveragingRange_s(svec,
+                                 initial.conditions,
+                                 parameters,
+                                 numsim)
+
+   s_Vmax_varvec[i] = var(log10(results2_s$Vmax))
+   s_Vmax_meanvec[i] = mean(log10(results2_s$Vmax))
+   s_totalV_varvec[i] = var(results2_s$total_V)
+   s_Xmax_varvec[i] = var(log10(results2_s$Xmax))
+   s_Xmax_meanvec[i] = mean(log10(results2_s$Xmax))
+ }
> s_cvvec = sqrt(s_varvec)/s_meanvec
> s_Vmax_cvvec = sqrt(s_Vmax_varvec)/s_Vmax_meanvec
> s_Xmax_cvvec = sqrt(s_Xmax_varvec)/s_Xmax_meanvec
> plot(s_Vmax_cvvec ~ s_cvvec,
+      type = "b", col = "red",
+      lty = 1, xlab = "Cv <s>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim = c(0,max(s_Vmax_cvvec)),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(s_Xmax_cvvec ~ s_cvvec,
+       type = "b", col = "blue",
+       lty = 1, xlab = "Cv <s>",
```

```
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max(s_Xmax_cvvec)),
+       cex.axis = 1.5, cex.lab = 1.5)
> legend("topleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
```

7.1.3 Variation in ϕ

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/
> #Antia-Jonathan/WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> #set.seed()
>
> ODE_Ebola <- function(t, y, parms)
+ {
+   with(as.list(c(y,parms)),
+        # allows the parameter file
+       #parms to be as a list
+ {     # and the state variables to be
+       #the same as chosen
+       #in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))                #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
```

```
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
+   phi = 1e3,  # parasite density at
+   #which immunity responds
+   k = 1e-3,   # killing of virus by immunity
+   p = 1e3,      # cytokine production rate
+   phi_c = 1e3,  # cytokine production
+   #begins when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0, maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola,
+                          y=initial.conditions,
+                          times=time.points,
+                          parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]+soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red",
+      type="l",ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time,
+       col = 'blue', type = 'l')
> lines(soln$C ~ soln$time,
+       col = 'green', type = 'l')
> plot(log10(soln$V)  ~ soln$time,
+      col="red", type="l",
```

```
+      ylim=c(1,log10(ymax)),
+      ylab="log number",
+      xlab="time (days)")
> lines(log10(soln$X) ~ soln$time,
+       col = 'blue', type = 'l')
> lines(log10(soln$C) ~ soln$time,
+       col = 'green', type = 'l')
> legend("topright", c("Virus", "Adaptive",
+                      "Cytokines"),
+        lty = 1, lwd = 2,
+        col=c("red","blue","green"),
+        cex = 2)
> ########################################
> ##create these empty vectors
> #so that the results can be stored later
> ########################################
> ##vector for duration of infection
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ########################################
> #integrating function
> ########################################
> Tfunction <- function(x, y)
```

```
+   ########################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the
+   ##difference between the times
+   #rollmean finds the average
+   ##between the two numbers in y
+   #this method leaves out one number
+   ##(the last number in the list)
+   #so it underestimates transmission
+   sum(diff(x)*rollmean(y,2))
+ }
> ########################################
> #simulation function
> ########################################
> Simulation <- function(initialconditions,
+                        params)
+ {
+   solution.of.ode = lsodar(ODE_Ebola,
+                            y=initialconditions,
+                            times=time.points,
+                            parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V))-params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
+   return(y)
+ }
> ################################################
> #calculation function
> ################################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
```

```
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density as a
+   ##measure of pathology
+   ##we are measuring total pathology
+   ##and max parasitemia
+   ##find the maximum of the
+   ##total pathogen and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density
+   ##for duration of infection
+   #maximizing immunity is
+   ##desired in antibiotic treatment
+   #might increase likelihood
+   ##of generating memory
+   #we calculate the max
```

```
+   ##and total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of
+   ##cytokines and total cytokine production
+   Cmax = max(x$C)
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(
+     durationofinfection = durationofinfection,
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
+            V_sim = V_sim, X_sim = X_sim,
+            t_sim = t_sim)
+ }
> ################################################
> a = Simulation(initial.conditions,parameters)
> b = Calculation(a)
> #averaging function
> ################################################
> Averaging <- function(initialconditions,
+                       params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,params)
+     b = Calculation(a)
+     durationofinfectionvec[i] =
+       b$durationofinfection
```

```
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] =
+       b$haltingTransition
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection =
+     mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
+   y = list(
+   durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             numdeath = numdeath,
+             V_simmat = V_simmat,
+             X_simmat = X_simmat,
+             time_simmat = time_simmat)
+   return(y)
```

```
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
> results = Averaging(initial.conditions,
+                     parameters,numsim)
> ################################################
> ##defining functions
> #hold all other parameters constant, vary one
> ################################################
> # averaging functions
> ################################################
> # r averaging function
> ################################################
> AveragingRange_phi <- function(inputvec,
+                                initialconditions,
+                                params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     params["phi"] = inputvec[i]
+     c = Averaging(initialconditions,params,1)
+     durationofinfectionvec[i] =
+       c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
```

```
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(
+ durationofinfectionvec = durationofinfectionvec,
+             Vmaxvec = Vmaxvec,
+             total_Vvec = total_Vvec,
+             total_lnVvec = total_lnVvec,
+             Xmaxvec = Xmaxvec,
+             total_Xvec = total_Xvec,
+             Cmaxvec = Cmaxvec,
+             total_Cvec = total_Cvec,
+             numdeathvec = numdeathvec,
+             V_simmat = V_simmat,
+             X_simmat = X_simmat,
+             time_simmat = time_simmat)
+   return(y)
+ }
> ################################################
> phimean = 3
> phi_dist = runif(1e3,2,4)
> phivec = sample(phi_dist,150)
> quartz()
> par(mfcol=c(2,2))
> #hist(phi_dist,prob=TRUE,
> #main = paste(""), xlab = "phi", cex = 2)
> #hist(phivec,prob=TRUE)
>
> #define the matrices before you begin
> #the multiple simulation as the size
> #is completely dependent on the time
> #points and number of values
> #in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(phivec))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(phivec))
```

```
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(phivec))
> #leave it at 1 simulation for snow
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations
> #in finding the averaging simulation values
> numsim = 1
> #solve our function for the
> #sample data we have used
> results_phi = AveragingRange_phi(10^(phivec)
+                                  ,initial.conditions
+                                  ,parameters
+                                  ,numsim)
> #plot multisimulation of V
> for(i in 1:ncol(results_phi$V_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_phi$V_simmat[,i])
+          ~ results_phi$time_simmat[,i],
+          type = "l", col = "red",
+          lty = i, lwd = 0.1,
+          ylab = "log V", xlab = "t",
+          ylim = c(0,10),
+          cex.axis = 1.5, cex.lab = 1.5)
+   }
+   else
+   {
+     lines(log10(results_phi$V_simmat[,i])
+           ~ results_phi$time_simmat[,i],
+           col = "red", lty = i, lwd = 0.1,
+           ylab = "Log V", xlab = "t",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot multsimulation of X
> for(i in 1:ncol(results_phi$X_simmat))
```

```
+ {
+   if(i == 1)
+   {
+     plot(log10(results_phi$X_simmat[,i])
+          ~ results_phi$time_simmat[,i],
+          type = "l", col = "blue",
+          lty = i, lwd = 0.1, xlab = "t",
+          ylab = "log X", ylim = c(0,5),
+          cex.axis = 1.5, cex.lab = 1.5)
+   }
+   else
+   {
+     lines(log10(results_phi$X_simmat[,i])
+           ~ results_phi$time_simmat[,i],
+           type = "l", col = "blue",
+           lty = i, lwd = 0.1, xlab = "t",
+           ylab = "log X",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot how our values of interest
> #change with changing r
> #sort command orders
> #the values in ascending order
> #so that they can be plotted as lines
> plot(log10(sort(results_phi$Vmax))
+      ~ sort(phivec),
+      type = "l",col = "red",
+      lty = 1, xlab = "log phi",
+      ylab = "log Number",
+      ylim = c(0,
+           max(log10(results_phi$Vmax))),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(log10(sort(results_phi$Xmax,
+           decreasing = TRUE))
+       ~ sort(phivec),
+        type = "l", col = "blue",
+       lty = 1,
```

```
+        cex.axis = 1.5, cex.lab = 1.5)
> #add a legend
> legend("bottomleft", c("Vmax", "Xmax"),
+        lty = 1,
+        lwd = 1,
+        col=c("red","blue"), cex = 1.2)
> #mean = a/(a+b) for beta distribution
> #variance = a*b/((a+b)^2(a+b+1))
>
> #as a increases, the variance of
> #the distribution decreases
> phi_unif = runif(1e3, min = phimean,
+                  max = phimean)
> #for a uniform distribution
> #the mean is the median (max - min)/2
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
> phi_varvec = NULL
> phi_meanvec = NULL
> phi_Vmax_varvec = NULL
> phi_Vmax_meanvec = NULL
> phi_totalV_varvec = NULL
> phi_Xmax_varvec = NULL
> phi_Xmax_meanvec = NULL
> rangevec = seq(0,1,0.1)
> for(i in 1:length(rangevec))
+ {
+   phivec = runif(150, min = (phimean-rangevec[i]),
+                  max = (phimean+rangevec[i]))
+
+   #define the matrices before you begin
+   #the multiple simulation as the size
+   #is completely dependent on the time points
+   #and number of values in the input vector
+   V_simmat = matrix(nrow = length(time.points),
+                     ncol = length(phivec))
+   X_simmat = matrix(nrow = length(time.points),
+                     ncol = length(phivec))
```

```
+   time_simmat = matrix(nrow = length(time.points),
+                        ncol = length(phivec))
+
+   phi_varvec[i] = var(phivec)
+   phi_meanvec[i] = mean(phivec)
+
+   results2_phi = AveragingRange_phi(10^(phivec)
+                                     , initial.conditions
+                                     , parameters
+                                     , numsim)
+
+   phi_Vmax_varvec[i] = var(log10(results2_phi$Vmax))
+   phi_Vmax_meanvec[i] = mean(log10(results2_phi$Vmax))
+   phi_totalV_varvec[i] = var(results2_phi$total_V)
+   phi_Xmax_varvec[i] = var(log10(results2_phi$Xmax))
+   phi_Xmax_meanvec[i] = mean(log10(results2_phi$Xmax))
+ }
> phi_Vmax_cvvec = sqrt(phi_Vmax_varvec)/phi_Vmax_meanvec
> phi_cvvec = sqrt(phi_varvec)/phi_meanvec
> phi_Xmax_cvvec = sqrt(phi_Xmax_varvec)/phi_Xmax_meanvec
> plot(phi_Vmax_cvvec ~ phi_cvvec, type = "b",
+      col = "red", lty = 1, xlab = "Cv <log phi>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim= c(0,max((phi_Vmax_cvvec))),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(phi_Xmax_cvvec ~ phi_cvvec,
+       type = "b", col = "blue",
+       lty = 1, xlab = "Cv <log phi>",
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max((phi_Xmax_cvvec))),
+       cex.axis = 1.5, cex.lab = 1.5)
> legend("topleft", c("Vmax", "Xmax"),
+        lty = 1,lwd = 1,
+        col=c("red","blue"), cex = 1.2)
```

7.1.4 Variation in k

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/
>       #Antia-Jonathan/WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> #set.seed()
>
> ODE_Ebola <- function(t, y, parms)
+ {
+   with(as.list(c(y,parms)),
+        # allows the parameter
+        #file parms to be as a list
+ {      # and the state variables to be the
+       #same as chosen in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))           #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
+   phi = 1e3,  # parasite density at
+   #which immunity responds
+   k = 1e-3,   # killing of virus by immunity
```

```
+   p = 1e3,      # cytokine production rate
+   phi_c = 1e3,  # cytokine production
+   #begins when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0, maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola,
+                          y=initial.conditions,
+                          times=time.points,
+                          parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]+soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red",
+      type="l",ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time,
+       col = 'blue', type = 'l')
> #lines(soln$C ~ soln$time,
> #col = 'green', type = 'l')
>
> plot(log10(soln$V)  ~ soln$time,col="red",
+      type="l",ylim=c(1,log10(ymax)),
+      ylab="log number",xlab="time (days)")
> lines(log10(soln$X) ~ soln$time,
+       col = 'blue', type = 'l')
> #lines(log10(soln$C) ~ soln$time,
> #col = 'green', type = 'l')
>
> legend("topright", c("Virus", "Adaptive"),
```

```
+        lty = 1, lwd = 2,
+        col=c("red","blue"), cex = 2)
> ##############################################
> ##create these empty vectors
> #so that the results can be stored later
> ##############################################
> ##vector for duration of infection
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ##############################################
> #integrating function
> ##############################################
> Tfunction <- function(x, y)
+   ##########################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the difference
+   #between the times
+   #rollmean finds the average
+   #between the two numbers in y
+   #this method leaves out one number
+   #(the last number in the list)
+   #so it underestimates transmission
```

```
+   sum(diff(x)*rollmean(y,2))
+ }
> ##########################################
> #simulation function
> ##########################################
> Simulation <- function(initialconditions
+                        ,params)
+ {
+   solution.of.ode = lsodar(ODE_Ebola,
+                            y=initialconditions,
+                            times=time.points,
+                            parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V)) - params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
+   return(y)
+ }
> ##################################################
> #calculation function
> ##################################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density
+   #as a measure of pathology
+   ##we are measuring total pathology
+   #and max parasitemia
+   ##find the maximum of the total pathogen
```

```
+   #and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density
+   #for duration of infection
+   #maximizing immunity is desired
+   #in antibiotic treatment
+   #might increase likelihood of
+   #generating memory
+   #we calculate the max and
+   #total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve
+   #and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of cytokines
+   #and total cytokine production
+   Cmax = max(x$C)
```

```
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(
+ durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             V_sim = V_sim, X_sim = X_sim,
+             t_sim = t_sim)
+ }
> ##################################################
> a = Simulation(initial.conditions,parameters)
> b = Calculation(a)
> #averaging function
> ##################################################
> Averaging <- function(initialconditions
+                       ,params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,params)
+     b = Calculation(a)
+     durationofinfectionvec[i] =
+       b$durationofinfection
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] =
+       b$haltingTransition
```

```
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection =
+     mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
+   y = list(
+ durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             numdeath = numdeath,
+             V_simmat = V_simmat,
+             X_simmat = X_simmat,
+             time_simmat = time_simmat)
+   return(y)
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
> results = Averaging(initial.conditions,parameters,
```

```
+                     numsim)
> ######################################################
> ##defining functions
> #hold all other parameters constant, vary one
> ######################################################
> # averaging functions
> ######################################################
> # r averaging function
> ######################################################
> AveragingRange_k <- function(inputvec,
+                              initialconditions,
+                              params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     params["k"] = inputvec[i]
+     c = Averaging(initialconditions,params,1)
+     durationofinfectionvec[i] = c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(
+ durationofinfectionvec = durationofinfectionvec,
+            Vmaxvec = Vmaxvec,
+            total_Vvec = total_Vvec,
+            total_lnVvec = total_lnVvec,
+            Xmaxvec = Xmaxvec,
```

```
+            total_Xvec = total_Xvec,
+            Cmaxvec = Cmaxvec,
+            total_Cvec = total_Cvec,
+            numdeathvec = numdeathvec,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> ###################################################
> kmean = -3
> k_dist = runif(1e3,-4,-2)
> kvec = sample(k_dist, 150)
> quartz()
> par(mfcol=c(2,2))
> #hist(k_dist,prob=TRUE, main = paste(""),
> #xlab = "k", cex = 2)
> #hist(kvec,prob=TRUE)
>
> #define the matrices before you begin
> #the multiple simulation as the size
> #is completely dependent on the time points
> #and number of values in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(kvec))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(kvec))
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(kvec))
> #leave it at 1 simulation for snow
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations
> #in finding the averaging simulation values
> numsim = 1
> #solve our function for the sample data we have used
> results_k = AveragingRange_k(10^(kvec)
+                              ,initial.conditions
```

```
+                              ,parameters
+                              ,numsim)
> #plot multisimulation of V
> for(i in 1:ncol(results_k$V_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_k$V_simmat[,i])
+          ~ results_k$time_simmat[,i],
+          type = "l", col = "red",
+          lty = i, lwd = 0.1,
+          ylab = "log V", xlab = "t",
+          ylim = c(0,12),
+          cex.axis = 1.5, cex.lab = 1.5)
+   }
+   else
+   {
+     lines(log10(results_k$V_simmat[,i])
+           ~ results_k$time_simmat[,i],
+           col = "red", lty = i, lwd = 0.1,
+           ylab = "Log V", xlab = "t",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot multsimulation of X
> for(i in 1:ncol(results_k$X_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_k$X_simmat[,i])
+          ~ results_k$time_simmat[,i],
+          type = "l", col = "blue",
+          lty = i, lwd = 0.1,
+          xlab = "t", ylab = "log X",
+          ylim = c(0,6), cex.axis = 1.5,
+          cex.lab = 1.5)
+   }
+   else
```

```
+   {
+     lines(log10(results_k$X_simmat[,i])
+           ~ results_k$time_simmat[,i],
+           type = "l", col = "blue",
+           lty = i, lwd = 0.1,
+           xlab = "t", ylab = "log X",
+           cex.axis = 1.5, cex.lab = 1.5)
+   }
+ }
> #plot how our values of interest change with changing r
> #sort command orders the values in ascending order
> #so that they can be plotted as lines
> plot(log10(sort(results_k$Vmax,
+                 decreasing = TRUE))
+      ~ sort(kvec),
+      type = "l",col = "red", lty = 1,
+      xlab = "log k", ylab = "log Number",
+      ylim = c(0,max(log10(results_k$Vmax))),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(log10(sort(results_k$Xmax,
+                  decreasing = TRUE))
+       ~ sort(kvec),
+       type = "l", col = "blue", lty = 1,
+       cex.axis = 1.5, cex.lab = 1.5)
> #add a legend
> legend("bottomleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
> #mean = a/(a+b) for beta distribution
> #variance = a*b/((a+b)^2(a+b+1))
>
> #as a increases, the variance of
> #the distribution decreases
> k_unif = runif(1e3, min = kmean, max = kmean)
> #for a uniform distribution
> #the mean is the median (max - min)/2
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
```

```
> k_varvec = NULL
> k_meanvec = NULL
> k_Vmax_varvec = NULL
> k_Vmax_meanvec = NULL
> k_totalV_varvec = NULL
> k_Xmax_varvec = NULL
> k_Xmax_meanvec = NULL
> rangevec = seq(0,1,0.1)
> for(i in 1:length(rangevec))
+ {
+   kvec = runif(150, min = (kmean-rangevec[i]),
+                max = (kmean+rangevec[i]))
+
+   #define the matrices before you begin
+   #the multiple simulation as the size
+   #is completely dependent on the time points
+   #and number of values in the input vector
+   V_simmat = matrix(nrow = length(time.points),
+                     ncol = length(kvec))
+   X_simmat = matrix(nrow = length(time.points),
+                     ncol = length(kvec))
+   time_simmat = matrix(nrow = length(time.points),
+                        ncol = length(kvec))
+
+   k_varvec[i] = var(kvec)
+   k_meanvec[i] = mean(kvec)
+
+   results2_k = AveragingRange_k(10^(kvec)
+                                 , initial.conditions
+                                 , parameters
+                                 , numsim)
+
+   k_Vmax_varvec[i] = var(log10(results2_k$Vmax))
+   k_Vmax_meanvec[i] = mean(log10(results2_k$Vmax))
+   k_totalV_varvec[i] = var(results2_k$total_V)
+   k_Xmax_varvec[i] = var(log10(results2_k$Xmax))
+   k_Xmax_meanvec[i] = mean(log10(results2_k$Xmax))
+ }
```

```
> k_Vmax_cvvec = sqrt(k_Vmax_varvec)/k_Vmax_meanvec
> k_Xmax_cvvec = sqrt(k_Xmax_varvec)/k_Xmax_meanvec
> ##might need to use the absolute value
> #command to ensure positive results
> k_cvvec = sqrt(k_varvec)/abs(k_meanvec)
> plot(k_Vmax_cvvec ~ k_cvvec, type = "b",
+      col = "red", lty = 1,
+      xlab = "Cv <log k>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim = c(0,max(k_Vmax_cvvec)),
+      cex.axis = 1.5, cex.lab = 1.5)
> lines(k_Xmax_cvvec ~ k_cvvec, type = "b",
+       col = "blue", lty = 1,
+       xlab = "Cv <log k>",
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max(k_Xmax_cvvec)),
+       cex.axis = 1.5, cex.lab = 1.5)
> legend("topleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
```

7.1.5 Variation in V(0)

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/
>       #Antia-Jonathan/WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> #set.seed()
>
> ODE_Ebola <- function(t, y, parms)
+ {
+   with(as.list(c(y,parms)),
+        # allows the parameter file parms
+        #  to be as a list
+ {      # and the state variables to be the same
```

```
+        #as chosen in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))           #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
+   phi = 1e3,  # parasite density at
+   #which immunity responds
+   k = 1e-3,   # killing of virus by immunity
+   p = 1e3,      # cytokine production rate
+   phi_c = 1e3,  # cytokine production begins
+   #when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0, maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola,
+                          y=initial.conditions,
+                          times=time.points,
+                          parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]
```

```
+                       +soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red",
+      type="l",ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time,
+       col = 'blue', type = 'l')
> lines(soln$C ~ soln$time,
+       col = 'green', type = 'l')
> plot(log10(soln$V)  ~ soln$time,col="red",
+      type="l",ylim=c(1,log10(ymax)),
+      ylab="log number",xlab="time (days)")
> lines(log10(soln$X) ~ soln$time,
+       col = 'blue', type = 'l')
> lines(log10(soln$C) ~ soln$time,
+       col = 'green', type = 'l')
> legend("topright", c("Virus", "Adaptive",
+                      "Cytokines"),
+        lty = 1, lwd = 2,
+        col=c("red","blue","green"), cex = 2)
> ##############################################
> ##create these empty vectors so
> ##that the results can be stored later
> ##############################################
> ##vector for duration of infection
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
```

```
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ################################################
> #integrating function
> ################################################
> Tfunction <- function(x, y)
+   ##############################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the
+   #difference between the times
+   #rollmean finds the average
+   #between the two numbers in y
+   #this method leaves out one number
+   #(the last number in the list)
+   #so it underestimates transmission
+   sum(diff(x)*rollmean(y,2))
+ }
> ################################################
> #simulation function
> ################################################
> Simulation <- function(initialconditions
+                        ,params)
+ {
+   solution.of.ode = lsodar(ODE_Ebola,
+                            y=initialconditions,
+                            times=time.points,
+                            parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V)) - params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
```

```
+   return(y)
+ }
> ################################################
> #calculation function
> ################################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density as
+   #a measure of pathology
+   ##we are measuring total pathology
+   #and max parasitemia
+   ##find the maximum of the
+   #total pathogen and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
```

```
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density
+   #for duration of infection
+   #maximizing immunity is
+   #desired in antibiotic treatment
+   #might increase
+   #likelihood of generating memory
+   #we calculate the max
+   #and total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of cytokines
+   #and total cytokine production
+   Cmax = max(x$C)
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(
+ durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             V_sim = V_sim, X_sim = X_sim,
+             t_sim = t_sim)
+ }
> ################################################
> a = Simulation(initial.conditions,
+                 parameters)
> b = Calculation(a)
```

```
> #averaging function
> ##############################################
> Averaging <- function(initialconditions,
+                       params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,params)
+     b = Calculation(a)
+     durationofinfectionvec[i] =
+       b$durationofinfection
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] =
+       b$haltingTransition
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection =
+     mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
+   y = list(
+ durationofinfection = durationofinfection,
```

```
+            Vmax = Vmax,
+            total_V = total_V,
+            total_lnV = total_lnV,
+            Xmax = Xmax,
+            total_X = total_X,
+            Cmax = Cmax,
+            total_C = total_C,
+            numdeath = numdeath,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
> results = Averaging(initial.conditions,
+                     parameters,numsim)
> ####################################################
> ##defining functions
> #hold all other parameters constant, vary one
> ####################################################
> # averaging functions
> ####################################################
> # r averaging function
> ####################################################
> AveragingRange_V0 <- function(inputvec,
+                               initialconditions,
+                               params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     initialconditions["V"] = inputvec[i]
+     c = Averaging(initialconditions,params,1)
```

```
+     durationofinfectionvec[i] =
+       c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(
+ durationofinfectionvec = durationofinfectionvec,
+             Vmaxvec = Vmaxvec,
+             total_Vvec = total_Vvec,
+             total_lnVvec = total_lnVvec,
+             Xmaxvec = Xmaxvec,
+             total_Xvec = total_Xvec,
+             Cmaxvec = Cmaxvec,
+             total_Cvec = total_Cvec,
+             numdeathvec = numdeathvec,
+             V_simmat = V_simmat,
+             X_simmat = X_simmat,
+             time_simmat = time_simmat)
+   return(y)
+ }
> ################################################
> V0mean = 3
> V0_dist = runif(1e3,0,6)
> V0vec = sample(V0_dist, 150)
> quartz()
> par(mfcol=c(2,2))
> #hist(V0_dist,prob=TRUE, main = paste(""),
```

```
> #xlab = "V(0)", cex = 2)
> #hist(V0vec,prob=TRUE)
>
> #define the matrices before you begin
> #the multiple simulation as the size
> #is completely dependent on the time points
> #and number of values in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(V0vec))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(V0vec))
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(V0vec))
> #leave it at 1 simulation for snow
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations
> #in finding the averaging simulation values
> numsim = 1
> #solve our function for the sample data we have used
> results_V0 = AveragingRange_V0(10^(V0vec)
+                                ,initial.conditions
+                                ,parameters
+                                ,numsim)
> #plot multisimulation of V
> for(i in 1:ncol(results_V0$V_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_V0$V_simmat[,i])
+          ~ results_V0$time_simmat[,i],
+          type = "l", col = "red",
+          lty = i, lwd = 0.1,
+          ylab = "log V", xlab = "t",
+          ylim = c(0,12),
+          cex.lab = 1.5, cex.axis = 1.5)
+   }
+   else
```

```
+   {
+     lines(log10(results_V0$V_simmat[,i])
+           ~ results_V0$time_simmat[,i],
+           col = "red", lty = i, lwd = 0.1,
+           ylab = "Log V", xlab = "t",
+           cex.lab = 1.5, cex.axis = 1.5)
+   }
+ }
> #plot multsimulation of X
> for(i in 1:ncol(results_V0$X_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_V0$X_simmat[,i])
+          ~ results_V0$time_simmat[,i],
+          type = "l", col = "blue",
+          lty = i, lwd = 0.1,
+          xlab = "t", ylab = "log X",
+          ylim = c(0,5),
+          cex.lab = 1.5,
+          cex.axis = 1.5)
+   }
+   else
+   {
+     lines(log10(results_V0$X_simmat[,i])
+           ~ results_V0$time_simmat[,i],
+           type = "l", col = "blue",
+           lty = i, lwd = 0.1,
+           xlab = "t", ylab = "log X",
+           cex.lab = 1.5, cex.axis = 1.5)
+   }
+ }
> #plot how our values of interest change with changing r
> #sort command orders the values in ascending order
> #so that they can be plotted as lines
> plot(log10(sort(results_V0$Vmax))
+      ~ sort(V0vec),
+      type = "l",col = "red", lty = 1,
```

```
+      xlab = "log V(0)", ylab = "log Number",
+      ylim = c(0,max(log10(results_V0$Vmax))),
+      cex.lab = 1.5, cex.axis = 1.5)
> lines(log10(sort(results_V0$Xmax)) ~ sort(V0vec),
+       type = "l", col = "blue", lty = 1,
+       cex.lab = 1.5, cex.axis = 1.5)
> #add a legend
> legend("bottomleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
> #mean = a/(a+b) for beta distribution
> #variance = a*b/((a+b)^2(a+b+1))
>
> #as a increases, the variance of
> #the distribution decreases
> V0_unif = runif(1e3, min = V0mean, max = V0mean)
> #for a uniform distribution
> #the mean is the median (max - min)/2
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
> V0_varvec = NULL
> V0_meanvec = NULL
> V0_Vmax_varvec = NULL
> V0_Vmax_meanvec = NULL
> V0_totalV_varvec = NULL
> V0_Xmax_varvec = NULL
> V0_Xmax_meanvec = NULL
> rangevec = seq(0,3,0.3)
> for(i in 1:length(rangevec))
+ {
+   V0vec = runif(150, min = (V0mean-rangevec[i]),
+                 max = (V0mean+rangevec[i]))
+
+   #define the matrices before you begin
+   #the multiple simulation as the size
+   #is completely dependent on the time points
+   #and number of values in the input vector
+   V_simmat = matrix(nrow = length(time.points),
```

```
+                     ncol = length(V0vec))
+   X_simmat = matrix(nrow = length(time.points),
+                     ncol = length(V0vec))
+   time_simmat = matrix(nrow = length(time.points),
+                        ncol = length(V0vec))
+
+   V0_varvec[i] = var(V0vec)
+   V0_meanvec[i] = mean(V0vec)
+
+   results2_V0 = AveragingRange_V0(10^(V0vec),
+                                   initial.conditions,
+                                   parameters, numsim)
+
+   V0_Vmax_varvec[i] = var(log10(results2_V0$Vmax))
+   V0_Vmax_meanvec[i] = mean(log10(results2_V0$Vmax))
+   V0_totalV_varvec[i] = var(log10(results2_V0$total_V))
+   V0_Xmax_varvec[i] = var(log10(results2_V0$Xmax))
+   V0_Xmax_meanvec[i] = mean(log10(results2_V0$Xmax))
+ }
> V0_Vmax_cvvec = sqrt(V0_Vmax_varvec)/V0_Vmax_meanvec
> V0_Xmax_cvvec = sqrt(V0_Xmax_varvec)/V0_Xmax_meanvec
> V0_cvvec = sqrt(V0_varvec)/V0_meanvec
> plot(sort(V0_Vmax_cvvec) ~ sort(V0_cvvec),
+      type = "b", col = "red", lty = 1,
+      xlab = "Cv <log V(0)>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim = c(0,max(V0_Vmax_cvvec)),
+      cex.lab = 1.5, cex.axis = 1.5)
> lines(sort(V0_Xmax_cvvec) ~ sort(V0_cvvec),
+       type = "b", col = "blue", lty = 1,
+       xlab = "Cv <log V(0)>",
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max(V0_Xmax_cvvec)),
+       cex.lab = 1.5, cex.axis = 1.5)
> legend("topleft", c("Vmax", "Xmax"), lty = 1,
+        lwd = 1, col=c("red","blue"), cex = 1.2)
```

7.1.6 Variation in X(0)

```
> require(deSolve)
> require(rootSolve)
> require(zoo)
> require(cacheSweave)
> #setwd("/Volumes/HDD/Dropbox/
>   #Antia-Jonathan/WorkInProgress/current/Ebola")
> setCacheDir("Cache")
> set.seed()
> ODE_Ebola <- function(t, y, parms)
+ {
+   with(as.list(c(y,parms)),
+        # allows the parameter
+       #file parms to be as a list
+ {    # and the state variables to be the
+     # same as chosen in the init conditions
+
+   dV = r*V - k*V*X
+   dX = s*X*V/(phi+V)
+
+   dY=c(dV,dX)
+   return(list(dY))     #
+ })
+ }
> ## define initial conditions
> initial.conditions = c(
+   V = 1,  # Virus
+   X = 1   # Adaptive immunity
+ )
> ## define parameters
> parameters = c(
+   r = 2,     # viral growth rate
+   s = 1,        # growth of adaptive immunity
+   phi = 1e3,  # parasite density at
+               #which immunity responds
+   k = 1e-3,   # killing of virus by immunity
+   p = 1e3,      # cytokine production rate
```

```
+   phi_c = 1e3,  # cytokine production begins
+   #when virus reaches this value
+   d_c = 72      # cytokine decay
+ )
> maxtime=30
> ntimepoints=200
> time.points = seq(0, maxtime,
+                   maxtime/ntimepoints)
> rootfun <- function(t, y, parms) return(y[1]-1)
> solution.of.ode = lsodar(ODE_Ebola,
+                          y=initial.conditions,
+                          times=time.points,
+                          parms = parameters)
> soln = as.data.frame(solution.of.ode)
> C = (parameters["p"]*soln$X*soln$V)/
+   (parameters["d_c"]*(parameters["phi_c"]
+                       +soln$V))
> soln = cbind(soln,C)
> par(mfcol=c(1,2))
> ymax=max(soln)
> ymin=min(soln)
> plot(soln$V ~ soln$time,col="red",
+      type="l",ylim=c(0,ymax),
+      ylab="number",xlab="time (days)")
> lines(soln$X ~ soln$time,
+       col = 'blue', type = 'l')
> lines(soln$C ~ soln$time,
+       col = 'green', type = 'l')
> plot(log10(soln$V)  ~ soln$time,col="red",
+      type="l",ylim=c(1,log10(ymax)),
+      ylab="log number",xlab="time (days)")
> lines(log10(soln$X) ~ soln$time,
+       col = 'blue', type = 'l')
> lines(log10(soln$C) ~ soln$time,
+       col = 'green', type = 'l')
> legend("topright", c("Virus", "Adaptive",
+                      "Cytokines"),
+        lty = 1, lwd = 2,
```

```
+        col=c("red","blue","green"),
+        cex = 2)
> ########################################
> ##create these empty vectors so
> #that the results can be stored later
> ########################################
> ##vector for duration of infection
> durationofinfectionvec = NULL
> ##vector for measures of pathology
> Vmaxvec = NULL
> ##vector for total transmission
> total_Vvec = NULL
> total_lnVvec = NULL
> ##vector for total immunity
> Xmaxvec= NULL
> total_Xvec = NULL
> ##total cytokine amount generated
> Cmaxvec = NULL
> total_Cvec = NULL
> haltingTransitionvec = NULL
> numdeathvec = NULL
> V_simvec = NULL
> X_simvec = NULL
> t_simvec = NULL
> ######################################
> #integrating function
> ######################################
> Tfunction <- function(x, y)
+   ###################################
+ {
+   #use the sum function to add the numbers
+   #diff function finds the difference
+   #between the times
+   #rollmean finds the average between
+   #the two numbers in y
+   #this method leaves out one number
+   #(the last number in the list)
+   #so it underestimates transmission
```

```
+   sum(diff(x)*rollmean(y,2))
+ }
> ########################################
> #simulation function
> ########################################
> Simulation <- function(initialconditions
+                        ,params)
+ {
+   solution.of.ode = lsodar(ODE_Ebola,
+                            y=initialconditions,
+                            times=time.points,
+                            parms = params)
+   soln = as.data.frame(solution.of.ode)
+   C = (params["p"]*soln$X*soln$V)/
+     ((params["phi_c"]+soln$V)) - params["d_c"]
+   soln = cbind(soln,C)
+   y = soln
+   return(y)
+ }
> ################################################
> #calculation function
> ################################################
> Calculation <- function(x)
+ {
+   ###storing data#####
+   V_sim = x$V
+   X_sim = x$X
+   t_sim = x$time
+
+   #####Calculations######
+   durationofinfection = max(x$time)
+
+   #####Pathology######
+   #max parasite density as a
+   #measure of pathology
+   ##we are measuring
+   #total pathology and max parasitemia
+   ##find the maximum of the
```

```
+   #total pathogen and store in vector
+   Vmax = max(x$V)
+   ##Store the value in a vector
+   total_V = Tfunction(x$t,x$V)
+
+   temp = log(x$V)
+   temp[!is.finite(temp)] <- 0
+   #t_list = x[which(x$V!=0),"t"]
+   t_list = x$t
+
+   if(all.equal(length(temp),0)==TRUE)
+   {
+     total_lnV = 0
+   }
+   else
+   {
+     total_lnV = Tfunction(t_list,temp)
+
+   }
+
+   ##Total Immunity Generated
+   #integral of immune density
+   #for duration of infection
+   #maximizing immunity is
+   #desired in antibiotic treatment
+   #might increase likelihood
+   #of generating memory
+   #we calculate the max
+   #and total immunity generated
+   Xmax = max(x$X)
+   #integrate the curve
+   #and store the value
+   total_X = Tfunction(x$t,x$X)
+
+   ##total cytokine production
+   #also taking the max amount of cytokines
+   #and total cytokine production
+   Cmax = max(x$C)
```

```
+   total_C = Tfunction(x$t,x$C)
+
+   y = list(
+ durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             V_sim = V_sim, X_sim = X_sim,
+             t_sim = t_sim)
+ }
> ################################################
> a = Simulation(initial.conditions,parameters)
> b = Calculation(a)
> #averaging function
> ################################################
> Averaging <- function(initialconditions,
+                       params,N)
+ {
+   numdeath = 0
+   for(i in 1:N)
+   {
+     a = Simulation(initialconditions,
+                    params)
+     b = Calculation(a)
+     durationofinfectionvec[i] =
+       b$durationofinfection
+     Vmaxvec[i] = b$Vmax
+     total_Vvec[i] = b$total_V
+     total_lnVvec[i] = b$total_lnV
+     Xmaxvec[i] = b$Xmax
+     total_Xvec[i] = b$total_X
+     Cmaxvec[i] = b$Cmax
+     total_Cvec[i] = b$total_C
+     haltingTransitionvec[i] =
```

```
+       b$haltingTransition
+     V_simmat[,i] = b$V_sim
+     X_simmat[,i] = b$X_sim
+     time_simmat[,i] = b$t_sim
+   }
+   durationofinfection =
+     mean(durationofinfectionvec)
+   Vmax = mean(Vmaxvec)
+   total_V = mean(total_Vvec)
+   total_lnV = mean(total_lnVvec)
+   Xmax = mean(Xmaxvec)
+   total_X = mean(total_Xvec)
+   Cmax = mean(Cmaxvec)
+   total_C = mean(total_Cvec)
+   numdeath = sum(haltingTransitionvec)
+
+   y = list(
+ durationofinfection = durationofinfection,
+             Vmax = Vmax,
+             total_V = total_V,
+             total_lnV = total_lnV,
+             Xmax = Xmax,
+             total_X = total_X,
+             Cmax = Cmax,
+             total_C = total_C,
+             numdeath = numdeath,
+             V_simmat = V_simmat,
+             X_simmat = X_simmat,
+             time_simmat = time_simmat)
+   return(y)
+ }
> numsim = 1
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = numsim)
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = numsim)
```

```
> results = Averaging(initial.conditions,
+                     parameters,numsim)
> ################################################
> #hold all other parameters constant, vary one
> ################################################
> # averaging functions
> ################################################
> # r averaging function
> ################################################
> AveragingRange_X0 <- function(inputvec,
+                               initialconditions,
+                               params, N)
+ {
+   for(i in 1:length(inputvec))
+   {
+     initialconditions["X"] = inputvec[i]
+     c = Averaging(initialconditions,params,1)
+     durationofinfectionvec[i] =
+       c$durationofinfection
+     Vmaxvec[i] = c$Vmax
+     total_Vvec[i] = c$total_V
+     total_lnVvec[i] = c$total_lnV
+     Xmaxvec[i] = c$Xmax
+     total_Xvec[i] = c$total_X
+     Cmaxvec[i] = c$Cmax
+     total_Cvec[i] = c$total_C
+     numdeathvec[i] = c$numdeath
+     V_simmat[,i] = c$V_simmat[,1]
+     colnames(V_simmat) <- c(inputvec)
+     X_simmat[,i] = c$X_simmat[,1]
+     colnames(X_simmat) <- c(inputvec)
+     time_simmat[,i] = c$time_simmat[,1]
+   }
+   y = list(durationofinfectionvec =
+              durationofinfectionvec,
+            Vmaxvec = Vmaxvec,
+            total_Vvec = total_Vvec,
+            total_lnVvec = total_lnVvec,
```

```
+            Xmaxvec = Xmaxvec,
+            total_Xvec = total_Xvec,
+            Cmaxvec = Cmaxvec,
+            total_Cvec = total_Cvec,
+            numdeathvec = numdeathvec,
+            V_simmat = V_simmat,
+            X_simmat = X_simmat,
+            time_simmat = time_simmat)
+   return(y)
+ }
> ######################################
> X0mean = 1
> X0_dist = runif(1e3,0,2)
> X0vec = sample(X0_dist, 150)
> quartz()
> par(mfcol=c(2,2))
> #hist(X0_dist,prob=TRUE, main = paste(""),
> #xlab = "X(0)", cex = 2)
> #hist(X0vec,prob=TRUE)
>
> #define the matrices before you begin
> #the multiple simulation as the size
> #is completely dependent on the time points
> #and number of values in the input vector
> V_simmat = matrix(nrow = length(time.points),
+                   ncol = length(X0vec))
> X_simmat = matrix(nrow = length(time.points),
+                   ncol = length(X0vec))
> time_simmat = matrix(nrow = length(time.points),
+                      ncol = length(X0vec))
> #leave it at 1 simulation for snow
> #numsim actually tells you the number of times
> #to run the simulation at the same values
> #held more use for stochastic simulations
> #in finding the averaging simulation values
> numsim = 1
> #solve our function for the sample data we have used
> results_X0 = AveragingRange_X0(10^(X0vec),
```

```
+                                initial.conditions,
+                                parameters,numsim)
> #plot multisimulation of V
> for(i in 1:ncol(results_X0$V_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_X0$V_simmat[,i])
+          ~ results_X0$time_simmat[,i],
+          type = "l", col = "red",
+          lty = i, lwd = 0.1,
+          ylab = "log V", xlab = "t",
+          ylim = c(0,9),
+          cex.lab = 1.5, cex.axis = 1.5)
+   }
+   else
+   {
+     lines(log10(results_X0$V_simmat[,i])
+           ~ results_X0$time_simmat[,i],
+           col = "red", lty = i, lwd = 0.1,
+           ylab = "Log V", xlab = "t",
+           cex.lab = 1.5, cex.axis = 1.5)
+   }
+ }
> #plot multsimulation of X
> for(i in 1:ncol(results_X0$X_simmat))
+ {
+   if(i == 1)
+   {
+     plot(log10(results_X0$X_simmat[,i])
+          ~ results_X0$time_simmat[,i],
+          type = "l", col = "blue",
+          lty = i, lwd = 0.1,
+          xlab = "t", ylab = "log X",
+          ylim = c(0,5), cex.lab = 1.5,
+          cex.axis = 1.5)
+   }
+   else
```

```
+   {
+     lines(log10(results_X0$X_simmat[,i])
+           ~ results_X0$time_simmat[,i],
+           type = "l", col = "blue",
+           lty = i, lwd = 0.1,
+           xlab = "t", ylab = "log X",
+           cex.lab = 1.5, cex.axis = 1.5)
+   }
+ }
> #plot how our values of interest
> #change with changing r
> #sort command orders the values in ascending order
> #so that they can be plotted as lines
> plot(log10(sort(results_X0$Vmax,
+                 decreasing = TRUE))
+      ~ sort(X0vec),
+      type = "l",col = "red", lty = 1,
+      xlab = "log X(0)", ylab = "log Number",
+      ylim = c(0,max(log10(results_X0$Vmax))),
+      cex.lab = 1.5, cex.axis = 1.5)
> lines(log10(sort(results_X0$Xmax,
+                  decreasing = TRUE))
+       ~ sort(X0vec),
+       type = "l", col = "blue",
+       lty = 1,cex.lab = 1.5,
+       cex.axis = 1.5)
> #add a legend
> legend("bottomleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
> #mean = a/(a+b) for beta distribution
> #variance = a*b/((a+b)^2(a+b+1))
>
> #as a increases, the variance of
> #the distribution decreases
> X0_unif = runif(1e3, min = X0mean, max = X0mean)
> #for a uniform distribution
> #the mean is the median (max - min)/2
```

```
> #for a uniform distribution
> #the variance is 1/12(xmax -xmin) or 1/12(range)
> X0_varvec = NULL
> X0_meanvec = NULL
> X0_Vmax_varvec = NULL
> X0_Vmax_meanvec = NULL
> X0_Vmax_varmeanvec = NULL
> X0_totalV_varvec = NULL
> X0_Xmax_varvec = NULL
> X0_Xmax_meanvec = NULL
> X0_Xmax_varmeanvec = NULL
> rangevec = seq(0,1,0.1)
> for(i in 1:length(rangevec))
+ {
+   X0vec = runif(150, min = (X0mean-rangevec[i]),
+                 max = (X0mean+rangevec[i]))
+
+   #define the matrices before you begin
+   #the multiple simulation as the size
+   #is completely dependent on the time points
+   #and number of values in the input vector
+   V_simmat = matrix(nrow = length(time.points),
+                     ncol = length(X0vec))
+   X_simmat = matrix(nrow = length(time.points),
+                     ncol = length(X0vec))
+   time_simmat = matrix(nrow = length(time.points),
+                        ncol = length(X0vec))
+
+   X0_varvec[i] = var(X0vec)
+   X0_meanvec[i] = mean(X0vec)
+
+   results2_X0 = AveragingRange_X0(10^(X0vec),
+                                   initial.conditions,
+                                   parameters, numsim)
+
+   X0_Vmax_varvec[i] = var(log10(results2_X0$Vmax))
+   X0_Vmax_meanvec[i] = mean(log10(results2_X0$Vmax))
+   X0_totalV_varvec[i] = var(results2_X0$total_V)
```

```
+   X0_Xmax_varvec[i] = var(log10(results2_X0$Xmax))
+   X0_Xmax_meanvec[i] = mean(log10(results2_X0$Xmax))
+ }
> X0_cvvec = sqrt(X0_varvec)/X0_meanvec
> X0_Vmax_cvvec = sqrt(X0_Vmax_varvec)/X0_Vmax_meanvec
> X0_Xmax_cvvec = sqrt(X0_Xmax_varvec)/X0_Xmax_meanvec
> plot(sort(X0_Vmax_cvvec) ~ sort(X0_cvvec),
+      type = "b",
+      col = "red", lty = 1,
+      xlab = "Cv <log X(0)>",
+      ylab = "Cv <log Vmax/Xmax>",
+      ylim = c(0,max(X0_Vmax_cvvec)),
+      cex.lab = 1.5, cex.axis = 1.5)
> lines(sort(X0_Xmax_cvvec)
+       ~ sort(X0_cvvec),
+       type = "b", col = "blue",
+       lty = 1, xlab = "Cv <log X(0)>",
+       ylab = "Cv <log Vmax/Xmax>",
+       ylim = c(0,max(X0_Xmax_cvvec)),
+       ex.lab = 1.5, cex.axis = 1.5)
> legend("topleft", c("Vmax", "Xmax"),
+        lty = 1, lwd = 1,
+        col=c("red","blue"), cex = 1.2)
```

7.1.7 Creating the Barplot

```
> Cv_r_V = max(r_Vmax_cvvec)
> Cv_r_X = max(r_Xmax_cvvec)
> Cv_s_V = max(s_Vmax_cvvec)
> Cv_s_X = max(s_Xmax_cvvec)
> Cv_phi_V = max(phi_Vmax_cvvec)
> Cv_phi_X = max(phi_Xmax_cvvec)
> Cv_k_V = max(k_Vmax_cvvec)
> Cv_k_X = max(k_Xmax_cvvec)
> Cv_V0_V = max(V0_Vmax_cvvec)
> Cv_V0_X = max(V0_Xmax_cvvec)
> Cv_X0_V = max(X0_Vmax_cvvec)
> Cv_X0_X = max(X0_Xmax_cvvec)
```

```
> Cv_Vvec = c(Cv_r_V, Cv_s_V, Cv_phi_V,
+             Cv_k_V, Cv_V0_V, Cv_X0_V)
> Cv_Xvec = c(Cv_r_X, Cv_s_X, Cv_phi_X,
+             Cv_k_X, Cv_V0_X, Cv_X0_X)
> data <- rbind(Cv_Vvec, Cv_Xvec)
> data <- as.matrix(data)
> colnames(data) <- c("r", "s", "phi",
+                     "k", "V(0)", "X(0)")
> quartz()
> barplot(data,  xlab="Parameters",
+         ylab = "Cv <log Vmax/Xmax>",
+         col=c("red","blue"),
+         #legend = c("Vmax", "Xmax"),
+         beside=TRUE, axis.lty = 1,
+         cex.lab = 1.5, cex.axis = 1.5)
> legend("topright", c("Vmax", "Xmax"), lty = 1,
+        lwd = 1, col=c("red","blue"), cex = 1.5)
```