The Last Plague

Investigating the Lethality of Pandemics Using Deterministic and Stochastic Epidemic Models

Co-Authors: Li Qing Wang & Uma Wu
Faculty Advisor: James D. Berger
SCI001 T2 Project
March 9, 2016

Abstract

This project investigates the effects of seven parameters on the spread of viral epidemics: infectivity, infectious period, mobility, incubation period, mortality, immunity and recovery period. Two algorithms were created to simulate the progression – one stochastic, and one deterministic. While the deterministic model simulates the spread through regulating the flux between compartments, the stochastic model simulates the spread by applying the parameters as a set of probabilities. The models were calibrated with current data on SARS and Ebola from WHO. Using optimization methods, we found the characteristics of viral diseases that would lead to the most lethal disease, marked by death rates of 92.26% of the world population in the deterministic model and 98.76% in the stochastic model. By investigating the effect of each parameter on the disease, we found that the key factors in regulating the spread of an epidemic are percentage immunity, mobility and recovery rate.

Introduction

Scientific question: what parameter values define the most deadly disease possible, and how do they each contribute to the epidemic?

An epidemic is defined as a sudden spread of a disease that affects an unexpectedly high number of people in a certain population [1]. The determinant factors for the lethality examined in this study include infectivity, infectious period, mobility, incubation period, mortality, immunity and recovery period (refer to Appendix Pages for definitions).

Although we are working with simulated diseases, the combination of parameters that define a "super viral disease" is important because any of these combinations may be present in a future infection. Therefore, by knowing the most dangerous aspects of an infectious disease, we can better focus our resources to counteract it.

This project has four phases: 0, I, II and III. The model is developed in Phase 0, calibrated in Phase I and used to answer our scientific question in Phases II and III. The models use the susceptible-infected-removed model with a few more compartments to provide insight into the effect of each on the spread of diseases.
The deterministic model is a closed model. The flux in and out of the compartments is represented as a series of differential equations. In order to optimize this program for certain desired outputs, the L-BFGS-B method was implemented [14]. The stochastic model is also closed, but it relies on a matrix system instead. To optimize this program for desired outputs, the Bayesian Optimization package Spearmint was implemented.

Different sets of parameters can contribute to the same death rate; therefore, further analysis was done to find the trend in those combinations.

Procedure

Phase 0

Principle Objective: To create the deterministic and stochastic models

Deterministic

Assumption 1: The parameters remain constant throughout the simulation.

Justification: This means the effects of borders, quarantines, cures, and other currently implementable disease-control tactics are unavailable. The implementation of these features would impede the observations on the spread of the disease amidst all of the other elements affecting it.

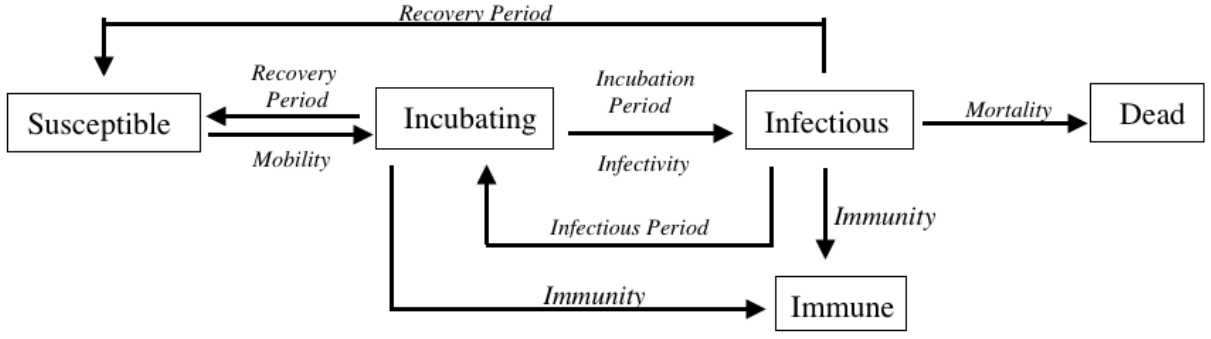


FIGURE 1: Flowchart of Deterministic Model. [For the complete code, refer to Appendix Pages 10-16]

Stochastic

-See also Assumptions 1 for the Deterministic Model-

Assumption 1: The disease can only spread between adjacent individuals.

Justification: While it is possible to simulate the disease spreading as a function of the total number of infected/incubating individuals regardless of their position in the simulated matrix, it defeats the point of having a stochastic process.

Assumption 2: The parameter values produced by optimizing the program are scalable and will produce around the same proportions between individuals of different states regardless of population size.

Justification: The spread of the disease overall have the same progression as long as the population is >9, since the disease needs to be spreading between unique individuals. This has been verified.

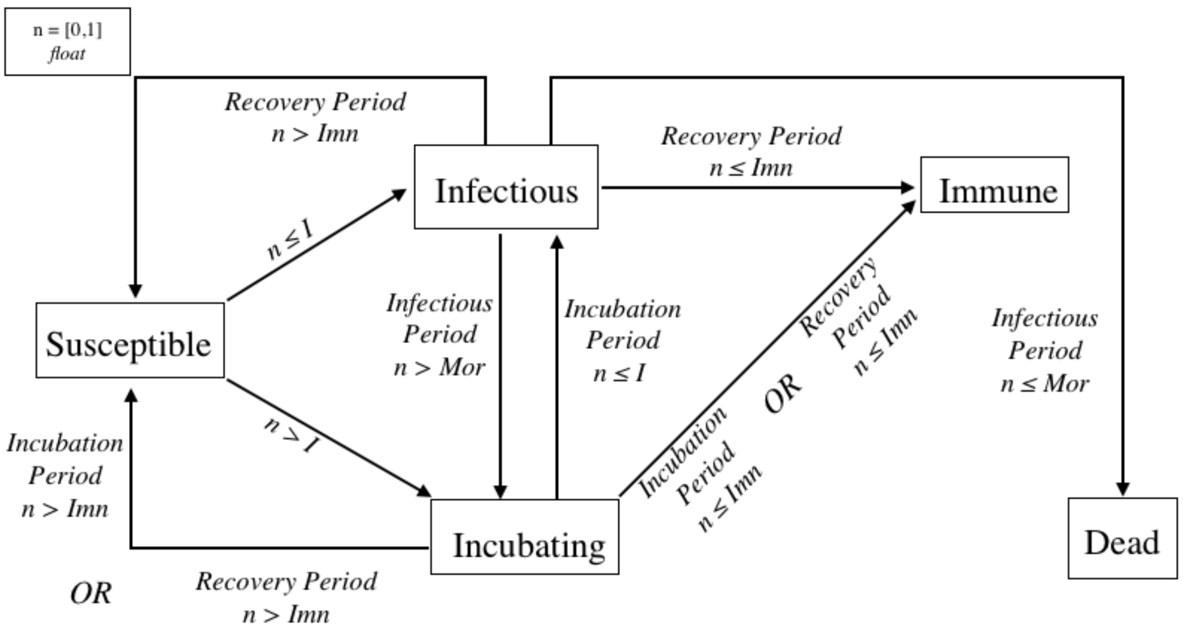


FIGURE 2: Flowchart of one individual's possible status within Stochastic Model
[For the complete code, refer to Appendix Pages 21-32]

Phase I

Principle Objective: Fitting and calibrating the program to real disease data

Cumulative data on the number of cases, recoveries and deaths for SARS and cumulative data on the number cases and deaths for Ebola were collected from the World Health Organization (WHO) website [7, 11].

Target points were chosen for the simulations to fit. The best trend line for each category of data was found by minimizing the coefficient of determination (R^2) to a value below a biological alpha of 5% with respect to the population using minimize function L-BFGS-B. For each disease, the category of data with the highest number of data points on the trend line was selected, and target points were chosen from these points. The death data and infected data for SARS and Ebola were used, respectively. The selected data points were representative of the trend of the disease, and minimized the noise in the raw data. [For the raw data used to fit the data points, refer to Appendix Pages 1-7.]

Phase II

Principal Objective: Finding the Parameters that result in the most deadly pandemics

In both models, the functions representing the simulation were reconfigured to return the negative maximum percentage of the population that is dead over the time frame of a year.

Deterministic - L-BFGS-B

The percentage mortality is maximized using the L-BFGS-B gradient evaluation method. The L-BFGS-B evaluates the derivative of the gradient from a given starting point and follows the slope to find a local minimum. To counteract the possibility of multiple minima within our function, multiple parameter sets were generated (~10000) to ideally produce as many minima as possible. Among these, the ones with the highest death rates were selected for analysis.

Stochastic - Spearmint
The Spearmint package uses Bayesian Optimization in a global black-box operation in order to find the global minimum of a function. This experiment required no more than using Spearmint to find the global minimum of the Stochastic outputs while minimizing the noise that comes from the random factors that govern its processes.
Assumption: everyone was susceptible.
Justification: if a significant fraction of a population is pre-immune, the disease may not become an epidemic at all.

Phase III
Principal Objective: investigating the effect of each parameter on percentage death independently.
While keeping other parameters at the values that lead to the most deadly disease, each parameter in the deterministic program was varied independently from 0.1~1.0, in increments of 10%. The output percentage death, defined by the percentage of the world population dying from the disease, was recorded.

Results

Each of the following sets of parameters fit the respective disease data collected from WHO with the sum of residuals squared indicated.

PHASE I RESULTS
*Deterministic Fitting Results (PHASE I) (*FIGURE 3)

	EBOLA	SARS	SARS
Infectivity	62.9133%	12.2454%	63.4744%
Infection Rate	30.6023%	98.3275%	37.0586%
Mobility	99%	98.211%	76.4858%
Incubation Period	1.395	101.01	150.443
Incubation Rate	71.68%	99%	0.6647%
Mortality	46.2063%	42.6484%	29.8424%
Percent Immunity	0.1%	0.1%	0.9516%
Recovery Rate	10.6243%	43.7013%	32.6739%
Population	6990000000	1795595082	1407579615
Residual Sum	2.546% of Pop	0.0083558% of Pop.	0.0081459% of Pop

Stochastic Fitting Results (PHASE I) (FIGURE 4)

	EBOLA	SARS
Infectivity	99%	43.14%
Infectious Period	25	11
Mobility	27.6382%	40.22%
Incubation Period	23	12

Mortality	36.72%	13.79%
Percentage Immunity	50.266%	85.85%
Recovery Period	28	11
Population	Any	Any
Standard Deviation	0.022747	0.022434

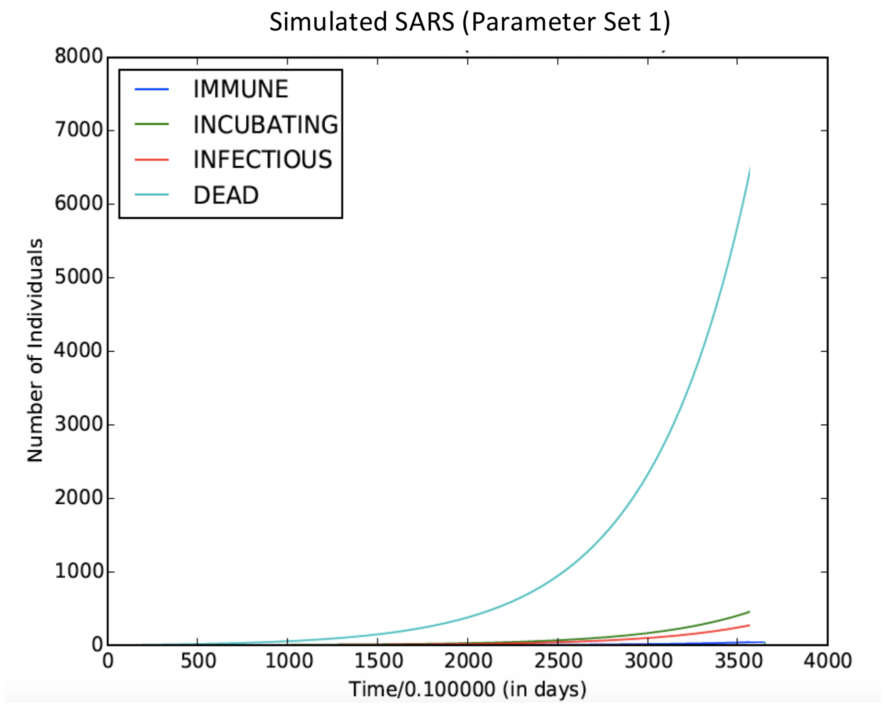


FIGURE 5.1: Plots of epidemic progression as generated by Deterministic Model & SARS Data

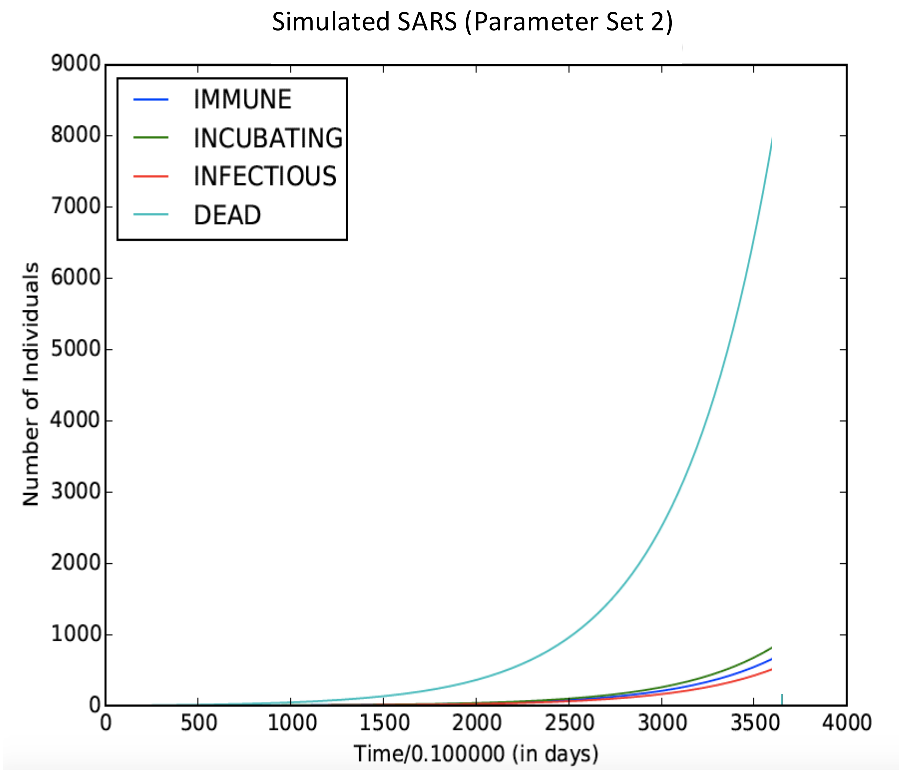


FIGURE 5.2: Plots of epidemic progression as generated by Deterministic

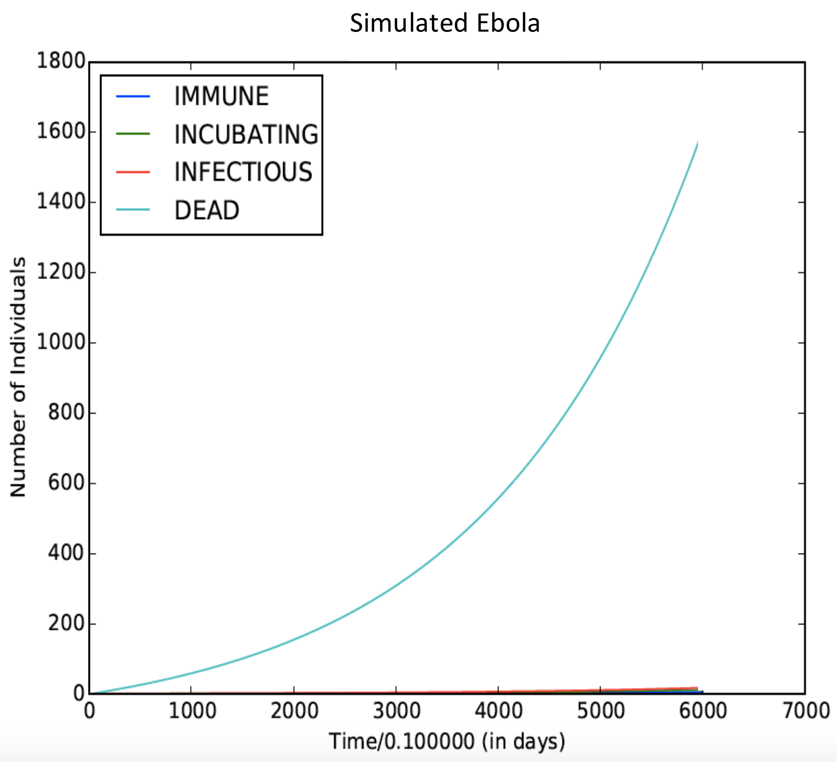


FIGURE 6: Plot of epidemic progression as generated by Deterministic Model & Ebola Data

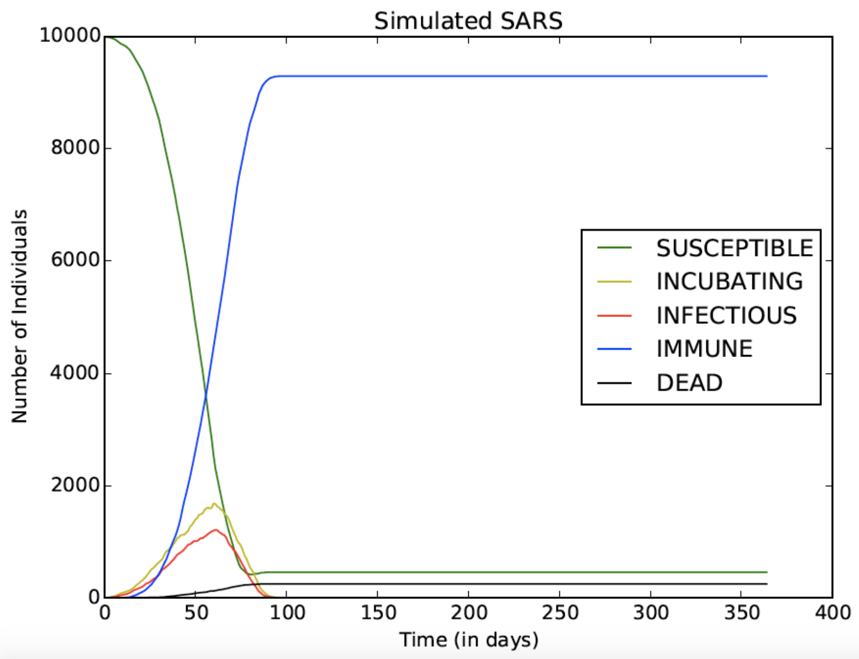


FIGURE 7: Plot of epidemic progression as generated by Stochastic Model & SARS Data

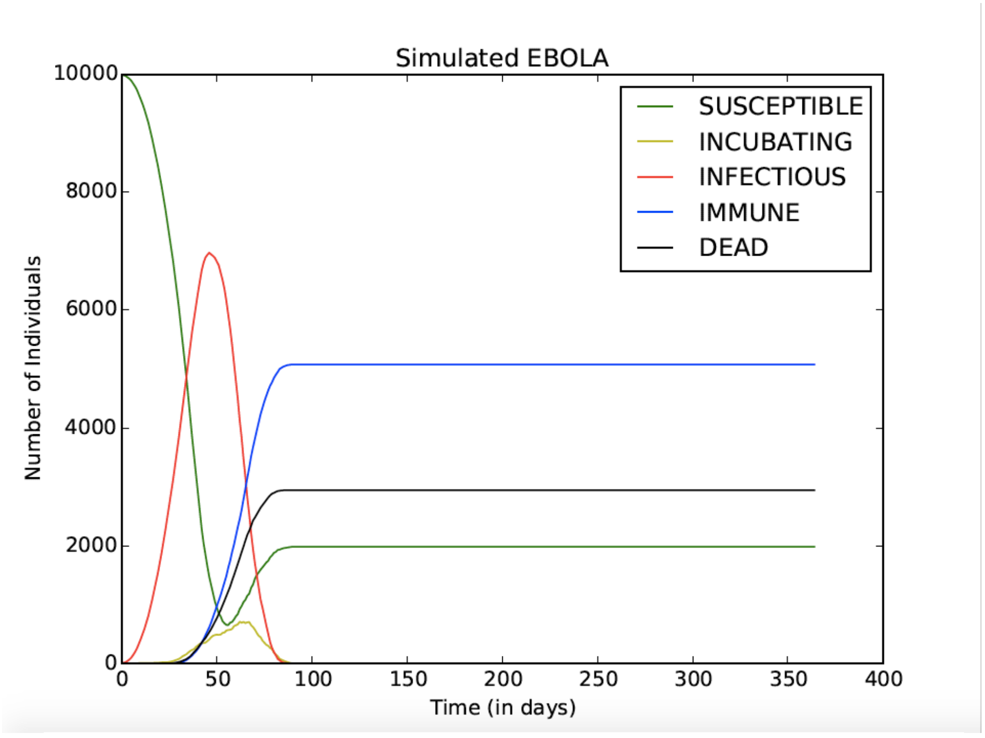


FIGURE 8: Plot of epidemic progression as generated by Stochastic Model &

Model	Disease	Estimated Basic Reproductive Number	R_0 from Literature [3, 15]
Deterministic	SARS	2.25	0.24-2.47, 2.87 2.4-3.6 (R_e)
Deterministic	SARS	1.13	
Deterministic	Ebola	2.88	1.50-2.67
Stochastic	Ebola	1.12	
Stochastic	SARS	1.0	0.24-2.47, 3.87 2.4-3.6 (R_e)

FIGURE 9: Estimation of Basic Reproduction Number

PHASE II RESULTS

Deterministic		Stochastic	
Infectivity	19.82%	Infectivity	99%
Infectious Period	1.124	Infectious Period	14
Infection Rate	90.53%	Infection Rate	7.14%
Mobility	99.00%	Mobility	65.61%
Incubation Period	7.731	Incubation Period	4
Incubation Rate	18.65%	Incubation Rate	25%
Mortality	99.00%	Mortality	90.45%
Percentage Immunity	1.0%	Percentage Immunity	0.00%
Recovery Rate	1.0%	Recovery Rate	4%
Recovery Period	100	Recovery Period	25
Percentage Dead	92.26%	Percentage Dead	98.76%

FIGURE 10: Parameters that constitute the most lethal disease in each model

	Average	Standard Deviation	Maximum	Minimum
Infectivity	19.82362286%	9.03404125%	37.5849%	1.5306%
Infectious Period	1.123803493	0.1633870674	1.6784099416	1.010101
Infectious Rate	90.53117143%	11.04190317%	99%	59.5802%
Mobility	99%	0%	99%	0.99
Incubation Period	7.7306150843	7.0662311461	38.6085479325	2.6623430216
Incubation Rate	18.64900571%	8.47061018%	37.5609%	2.5901%
Mortality	99%	0%	99%	99%
Percentage Immunity	1%	0%	1%	1%
Recovery Rate	1%	0%	1%	1%
Percentage Dead	92.25818%	8.622e-4%	92.2588%	92.2555%

FIGURE 11: PHASE II Deterministic Parameter Features

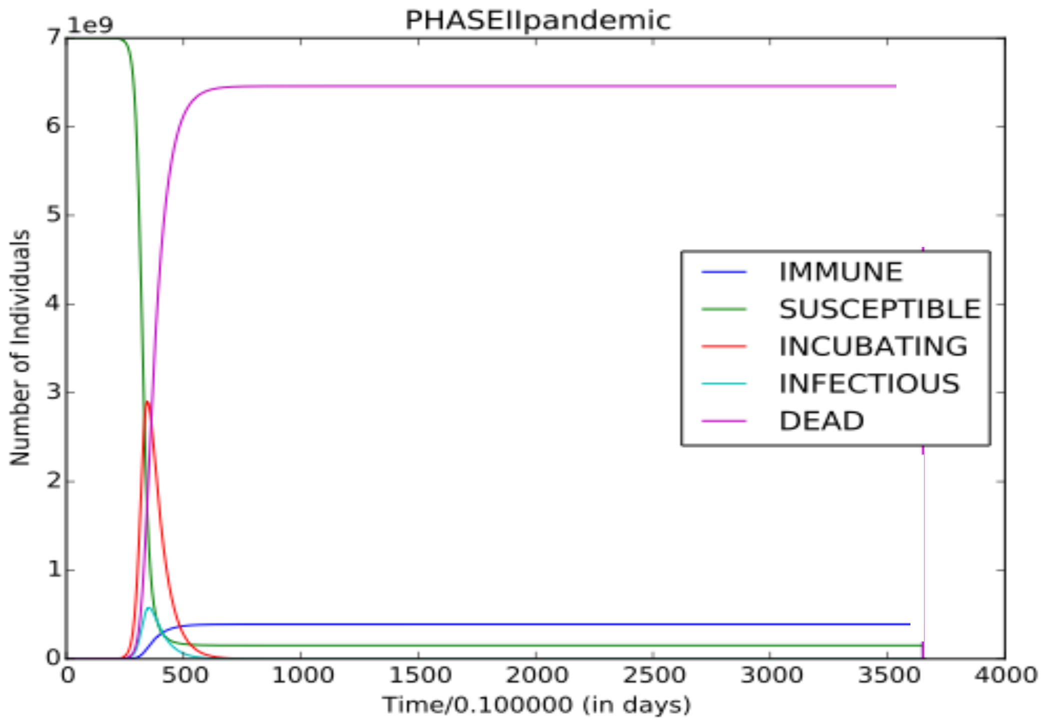


FIGURE 12: Trend of optimized pandemic (Deterministic)

Deterministic Model

The data shows parameter values that contribute to the most deadly disease predicted by the deterministic model. (Figure 10)

The disease is infectious slightly over 1 day. A diseased individual may infect 99% of the healthy people he meets. Only 19.82% of infecteds will display symptoms after infection. Other individuals do not show any symptom until 7.7 days after infection. 99% of infected individuals will die from the disease, and only 1% will gain immunity. Infected individuals, if capable of recovering from the disease, take 100 days to do so.

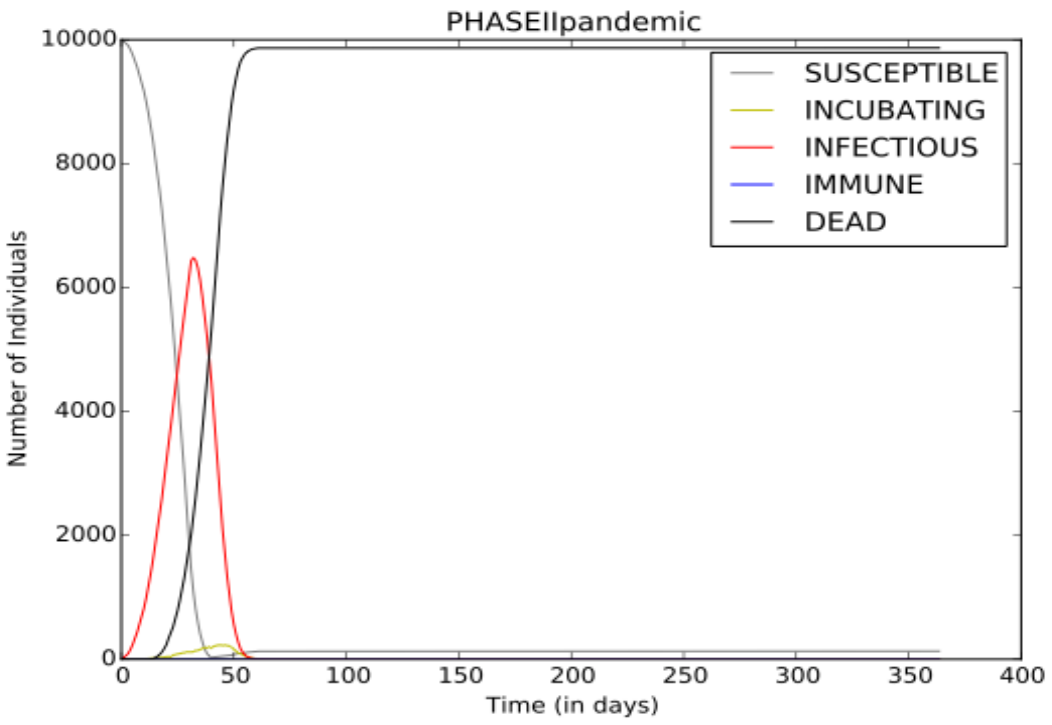


FIGURE 13: Trend of optimized pandemic (Stochastic)

Stochastic Model

The data shows parameter values that contribute to the most deadly disease predicted by the stochastic model. (Figure 10)

The disease incubates for 4 days. There is a 99% chance that an infected population will display symptoms of the disease, and the disease is infectious over a span of two weeks. A healthy individual has a 65.61% chance of contracting the disease from an infectious individual in close proximity. Infected individuals have a 90.45% chance of dying from the disease. No one can develop immunity against this disease, but individuals can recover from the disease over a period of 25 days.

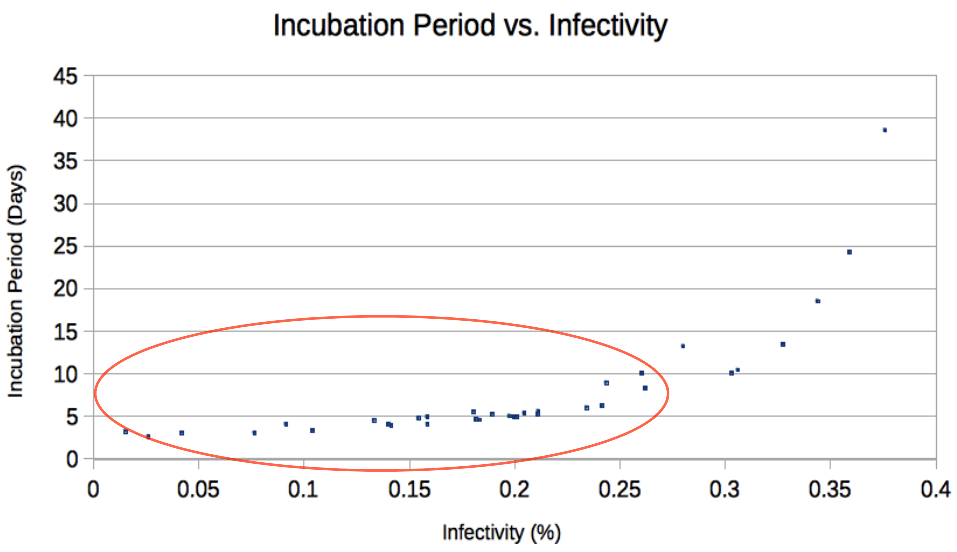


FIGURE 14.1: Incubation Period compared to Infectivity

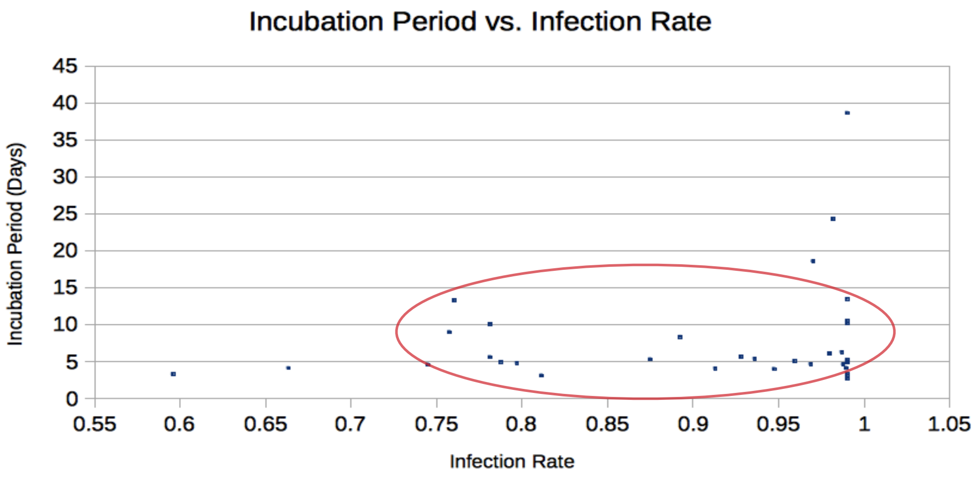


FIGURE 14.2: Percentage Death compared to Incubation Period

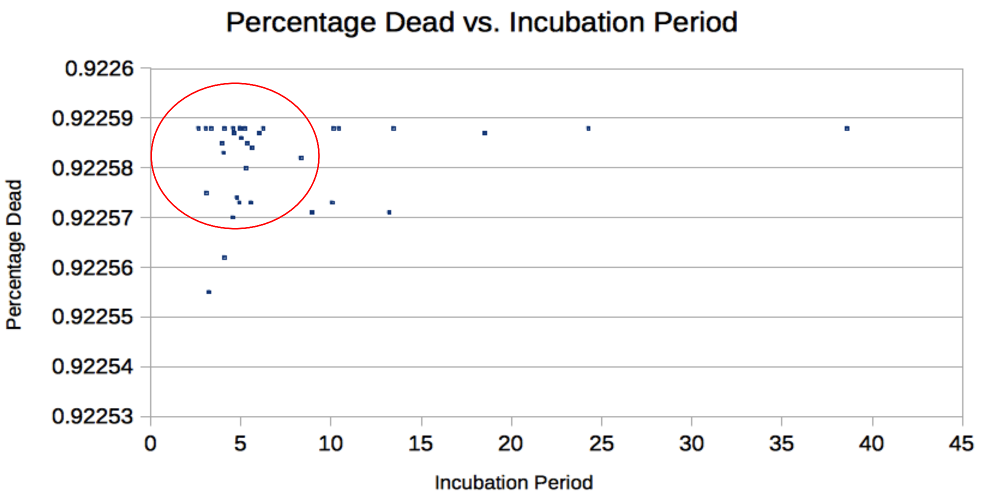


FIGURE 14.3: Incubation Period compared to Infection Rate

PHASE III RESULTS

Infectivity	Infectious Period	Mobility	Incubation Period	Mortality	Percentage Immunity	Recovery Rate
≤30%	1-2 days	≥20%	5-10 days	any	≤70%	≤70%

FIGURE 15: Table of criteria for a significant drop in population as a result of an epidemic

Analysis

Analysis of PHASE I Data

Estimation of Basic Reproduction Number

The basic reproduction number characterizes the ability of a disease to spread, and is calculated by dividing the infection rate by recovery rate. On the other hand, the effect reproduction number (R_e) represents the observed number of secondary infections from a primary infection. Since the two are sometimes used interchangeably in literature, both values were obtained from literature as references.

Two sets of parameters were selected for SARS because both provided R_0 that were within the literature range. No statistical comparison was feasible between our results and the literature values due to the limitations of the model. However, it is clear that the R_o estimated by both models (Figure 9) generally adhered to what was reported from the literature. The deterministic model generally predicts a higher R_0 than the stochastic model.

Incubation Periods

The stochastic model predicted a more reasonable incubation period for each disease compared to the deterministic one. The actual incubation periods for SARS and Ebola are up to two weeks [16] and up to 21 days [17], respectively. Both of these adhere significantly more to the stochastic results.

Deterministic vs. Stochastic

Comparing the shapes of the graphs of simulated SARS and Ebola produced by both models to the graphs of raw disease data, the stochastic model is clearly a better model than the deterministic one. The stochastic model is a better fit for the randomness in the nature of disease spread, unlike the deterministic.

Combining this with analyses of estimated R_0 and incubation period, the stochastic model seemed to have produced a lower residual for each disease (See Appendix Graphs).

Analysis of PHASE II Data

A Holistic Analysis of PHASE II Deterministic Data
This is done to deterministic only due to the various sets of optimized parameters it produced, all of which produced the desired maximal death rate. Of the 10000 parameter sets, 34 representative ones were selected for analysis.

The standard deviation for each parameter was computed using 34 sets of parameters that yield a percentage death of 92.26%. Since the standard deviation of mobility, mortality, percentage immunity and recovery are zero, these parameters must be at a specific value to result in the most lethal epidemic. Therefore, any changes made to those parameters alone will result in a death rate deviating from 92.26%.

The disease has high mobility (99%), high mortality (99%), low immunity (1%) and low recovery rate (1%). Infectivity can range from 1.53% to 37.58%, infection rate can range from 59.58% to 99% and incubation period can range from 2.66 ~ 38.61 days. A clear correlation between incubation period and infectivity can be observed below (Figure 14.1). When infectivity is below 20%, incubation period is less than a week. As infectivity goes beyond 20%, incubation period increases drastically and can be as high as 38 days.

Therefore, as shown in Figures 14.1-3, a combination of the following qualities is most likely to happen and will lead to a 92.26% death rate: 1.53% ~26.21% infectivity, a short infectious period of 1~1.33 days, 99% mobility, within two weeks of incubation, 99% mortality, 1% immunity and1% recovery.

Discussion

Accuracy & Precision

The accuracy of either models can be defined by how replicable the results are. The deterministic model is completely accurate since the same parameters will always yield the same output. Therefore, the accuracy of deterministic model is not representative of its uncertainty, since no model can have 0 uncertainty. However, for the stochastic model, standard deviations of outputs were determined. The standard deviation of the phase II result, (0.005512%), defines the uncertainty of the model since it is purely based on the outputs of the program and does not depend on any data. This small standard deviation indicates an accurate model.

The precision of either models can be defined as how well it fits to the real data in Phase I. This was obtained by fitting the simulation to the real data by minimizing the residuals. The stochastic model's precision cannot be determined through its precision due to the restraint in runtime as a function of population (it can only fit the trend), and is therefore not representative of its uncertainty. The deterministic model does yield a wide array of residuals, however. In our fitting of the data, the lowest residual was 2.546% of Population with Ebola, while the two SARS fittings yielded 0.0083558% of population and 0.0081459% of population respectively. Since the percentages are less than 5%, the model is likely precise.

Limitations

Limitations of Collected Data

Inconsistency in WHO data

The cumulative number of SARS cases fluctuated for the last few days for which data were collected. SARS is a diagnosis of exclusion; as previous cases were further investigated, some patients were re-examined and re-diagnosed [6]. Since the fluctuation only involved a few individuals, we disregarded the occasional decrease, and recorded further data by adding the number of newly dead people to the cumulative number.

Limitation of Obtainable Categories of Data

While there are 7 parameters, only the aforementioned 3 had available data.

Limitations of Model

1. Quarantine & Borders

In April 2003, the Chinese government started implementing quarantine and restricting social interactions by methods such as closing theatres and implementing border control [4]. Quarantine would limit the mobility of the disease; however, this is not accounted for in the model. (See Assumptions)

2. Parameters change with time

The parameters in both models are constant over time, while in reality they do change with time. For example, the mortality of both disease increased with time.

Run Time Constraints

Deterministic

The deterministic model does not take long to complete one function evaluation (~0.1s for Population = 7000000000) due to its nature as a linear-time computation, but its optimization took much longer due to the methodology of the L-BFGS-B method of evaluating the function multiple times until it reached a local minimum. Therefore, a global optimization would have been better for this instance.

Stochastic

The stochastic model takes a lot longer than the deterministic (~1s for Population = 400) to complete one function evaluation. With the Spearmint Optimization package, we were able to bypass that issue. Not only does Spearmint find the global minimum, it only takes around 5 hours to yield applicable and accurate results. However, it is still subject to the limitations of populations, since the run time increases linearly with the addition of an individual.

Inferences

Deterministic Phase II

Summary: The most deadly disease according to the deterministic model is a disease that doesn't show symptoms immediately (19.82% infectivity), transmits quickly from person to person (99% mobility), almost certainly and immediately kills its host upon infection (99% mortality), and is almost impossible to develop immunity from (only 1% of the population can become immune). . This disease can kill 92.26 % of the world population in one year.

This set of parameters produces a greater incubating population than infectious population. There are many currently known viruses that can remain dormant in individuals (e.g Rabies, STDs) for a relatively long time before manifesting symptoms. The disease may cover its lethality by displaying nonlethal typical flu symptoms as fever, cough and diarrhea when individuals are first infected, thus creating a high incubating population. This feature of the simulated disease greatly contributes to the risk of unknowingly getting infected since the number of susceptible people becoming incubating is dependent on both the number of carriers (infecteds) and susceptibles.

Stochastic Phase II

Summary: The most deadly disease according to the stochastic model is one that shows symptoms almost immediately (99% infectivity), transmits at a moderate rate (65.61%) from person to person, is infectious over two weeks, and is develop immunity. This disease can kill 98.76% of the world population in one year.

Instead of a high mobility, the disease becomes highly infective with a small incubation period, which means the patients would immediately show symptoms upon infection. A zero percent immunity means that even if a patient did manage to survive, they would become susceptible again with another chance of being infected and eventually succumbing to the disease.

This disease resembles past epidemics such as AIDS, since it mutates very quickly and suppresses the immune system. The disease would exhibit symptoms that lead to more infection and eventually death, such as skin lesions (e.g smallpox), discharging fluids (e.g diarrhea), and even irregular behavior (e.g rabies patients may bite other individuals).

Deterministic Phase III results and inferences

Since the purpose of Phase III is purely making inferences, its results will be shown here along with the inferences.

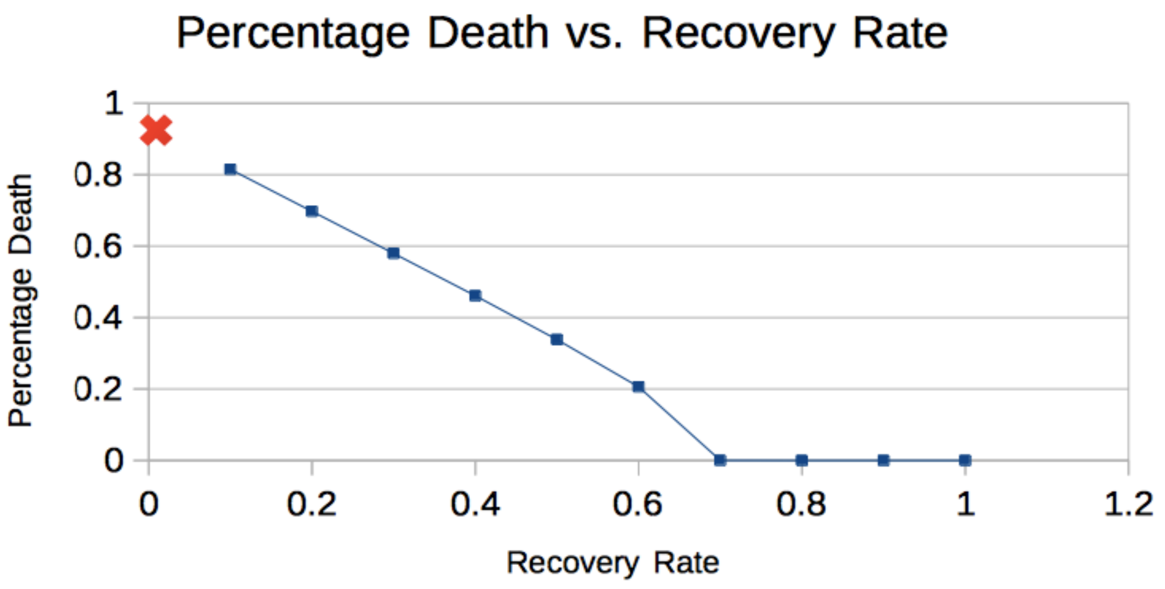


Figure 16.1 The percentage dead seems to decrease somewhat linearly with increased recovery rate when the recovery rate is lower than 70%. A recovery rate of 70% seems to be a threshold beyond which the disease is no longer an epidemic.

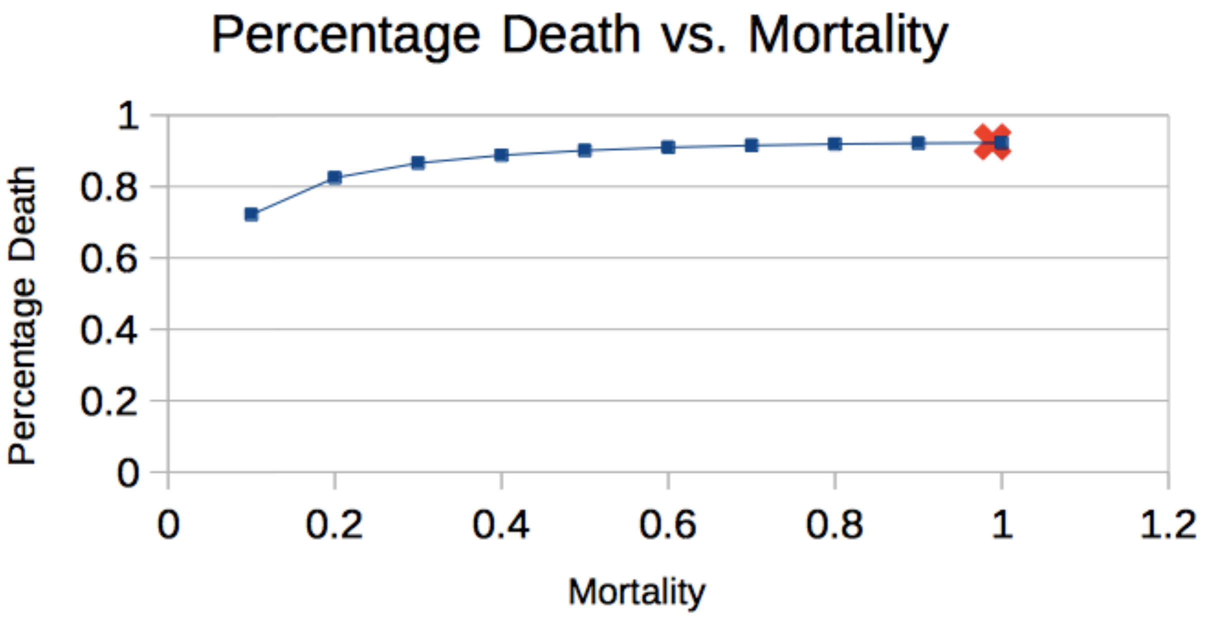


Figure 16.2 A disease with low mortality (10%) can have a high percentage death (~70%). The percentage death increase by ~10% in response to a 10 fold increase in mortality.

This may indicate that the mortality of a disease is not the major determinant of its percentage death.

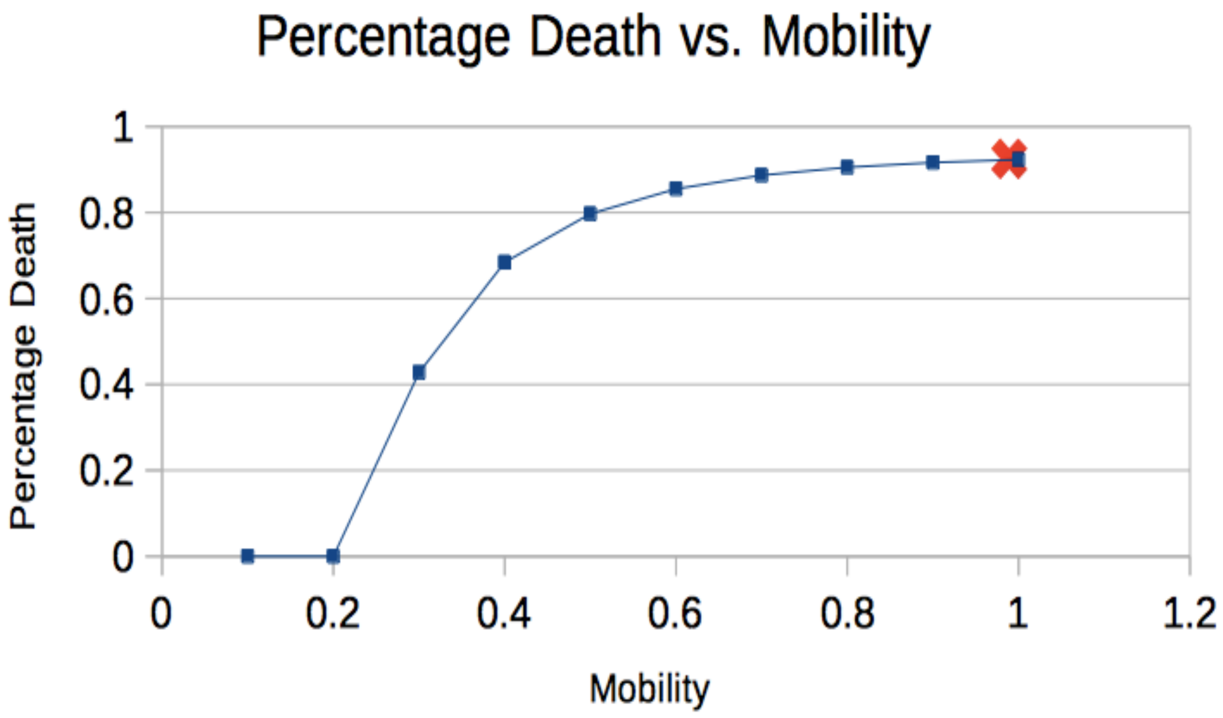


Figure 16.3 A disease for which an infected can only spread the infection to less than 20% of the people he contacts will not cause an epidemic. The percentage death increases drastically with the increase in mobility. This suggests that mobility is a major determinant in the spread of a disease.

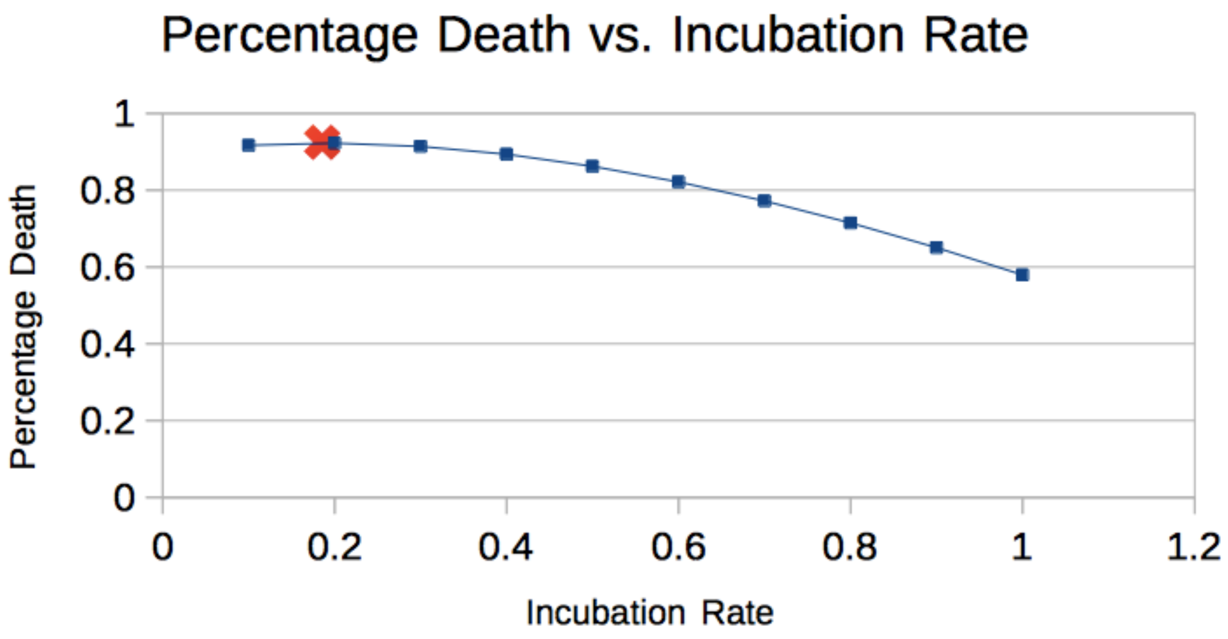


Figure 16.4 As incubation rate increases, or as the length of incubation period decreases, the percentage death decreases.

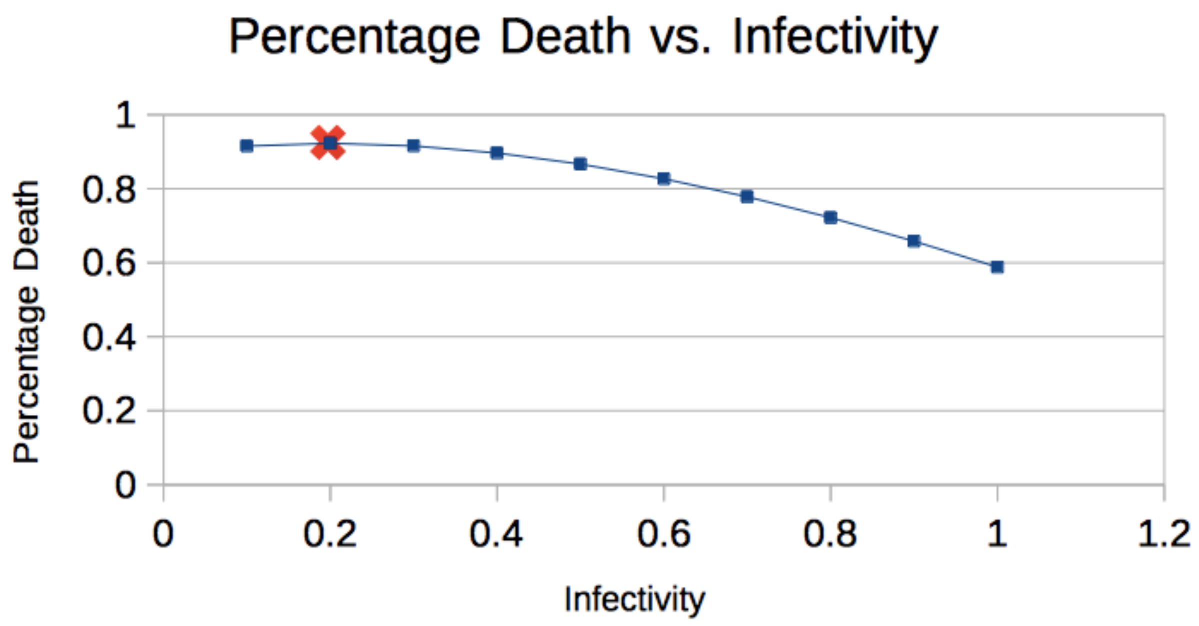


Figure 16.5 The larger the fraction of the infected population that shows symptoms, the less deadly the disease becomes. This may suggest that a disease must not let the patients display symptoms immediately in order to let it spread unknowingly, and eventually kill a large percentage of the world population.

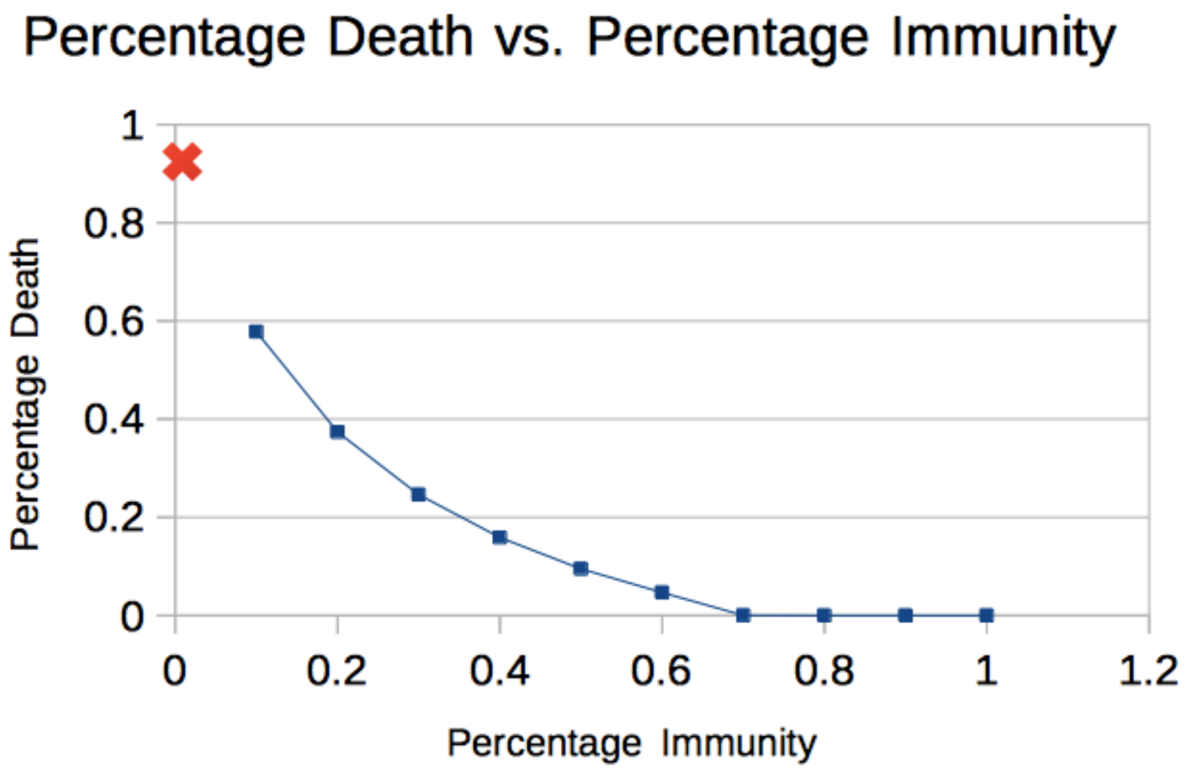


Figure 16.6 The disease is the most deadly when zero percent of the population is immune to it. The death rate decreases with increased immunity. There seems to be a threshold at 70% immunity, beyond which the disease is essentially no longer an epidemic.

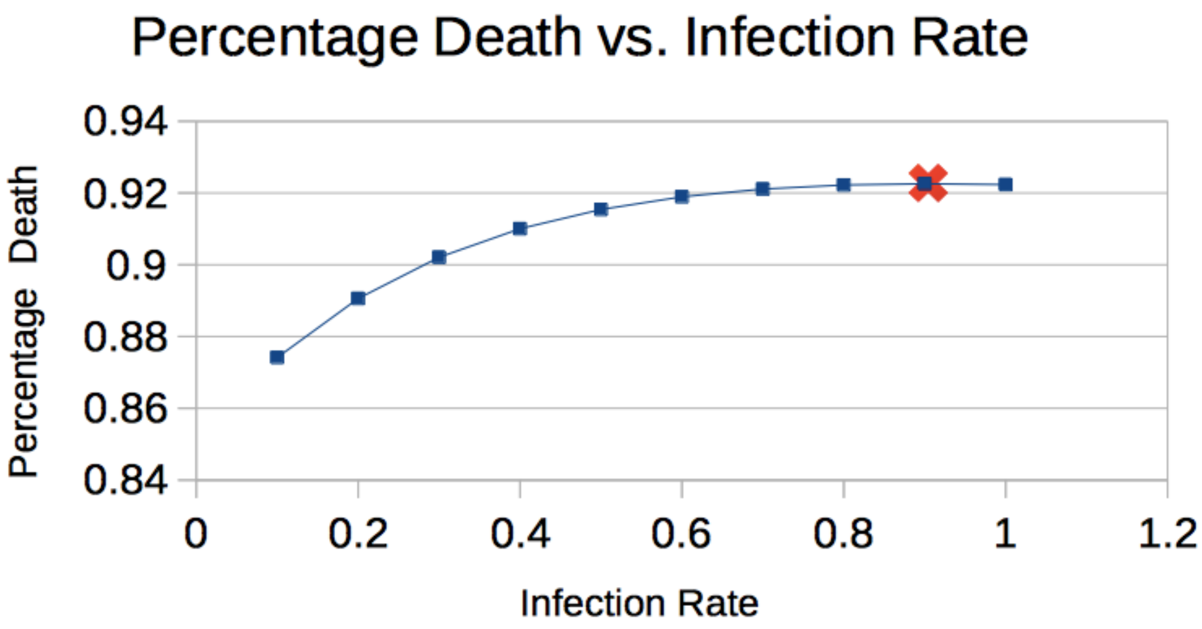


Figure 16.7 A low infection rate such as 10% can lead to a death percentage of ~87%. The increase in infection rate from 10% to 70% causes the most increase in percentage death. Beyond that, the increase in infection rate has minute effect on percentage death.

Key Inferences and Observations

In agreement with what was observed in phase II, any deviation from the set of the most deadly parameters led to a decrease in percentage death. However, this phase allowed a closer investigation of the magnitude of such effect. For example, the mortality barely alters the percentage death while independent changes in mobility, recovery rate, and percentage immunity led to much greater changes. The effect of infection rate, incubation rate, and infectivity is somewhere in between. Some threshold values were observed in percentage immunity (≤70%), mobility (≥20%) and recovery rate (≤70%). These three parameters are the major contributors to the percentage death of an epidemic; however, they must reach a certain threshold level for a disease to become an epidemic. These three parameters need to be regulated in order to prevent or control the spread of epidemic most efficiently.

As a summary of phase III, a chart of criteria for the most lethal epidemic predicted by the deterministic model was produced (figure 15). If an outbreak is suspected and several of its parameters fall into these values, it is likely to develop into a fairly lethal epidemic. Such disease can then be most effectively regulated by controlling percentage immunity, mobility and recovery rate.

Conclusion

Both the deterministic and stochastic models developed are able to fit realistic disease data and predict combinations of parameters that would lead to the most deadly epidemic. Both models also predict a high death rate when the chances of developing immunity are low, indicating the threat of an immune-suppressive disease is especially high.

Ranges of values for each parameter that constitute the most lethal epidemic outlined can be used to assess the lethality of a real disease. With further development, future potentials of this pair of models include investigation of other viral diseases, predicting the outcome of a disease in real time, and modelling the effect of prevention and control methods on the epidemic.

References

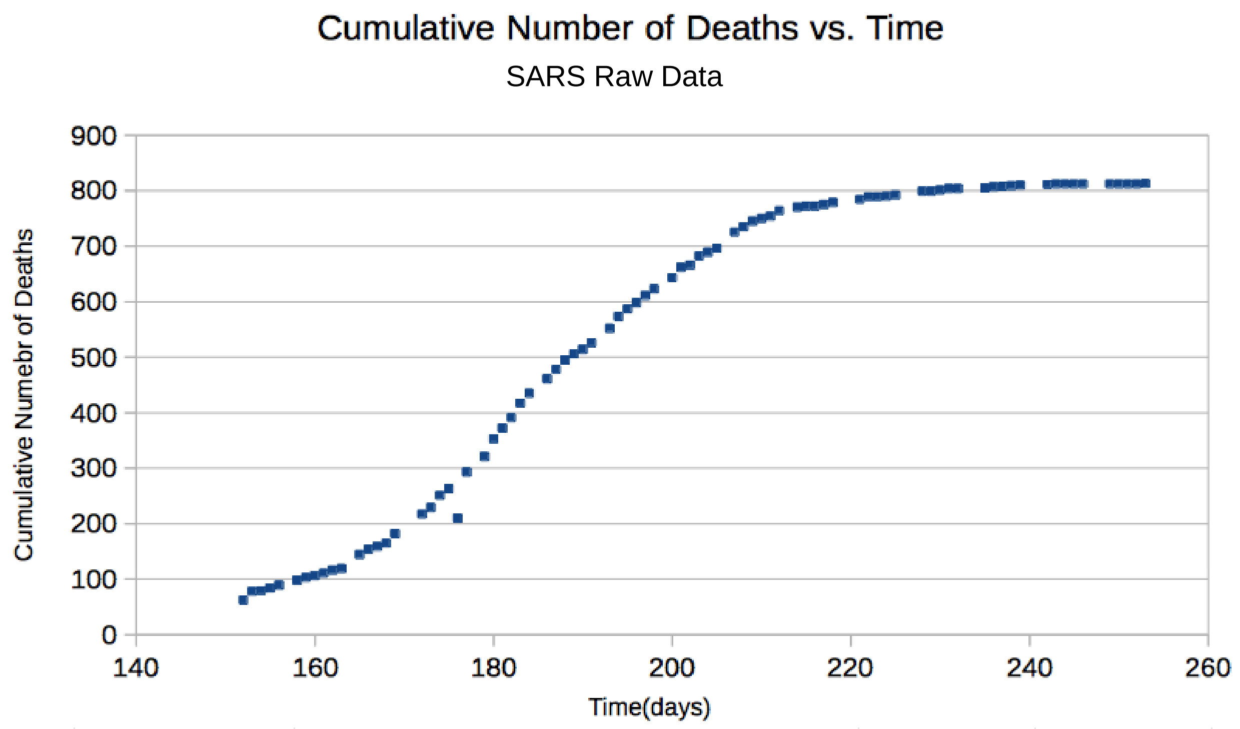
[1]"Lesson 1: Introduction to Epidemiology." *Centers for Disease Control and Prevention*. Centers for Disease Control and Prevention, 2012. Web. 11 Mar. 2016.
[2]"Learning Basic Epidemic Models with Python." *Learning Basic Epidemic Models with Python*. Web. 11 Mar. 2016.
[3]"Estimating the Reproduction Number of Ebola Virus (EBOV) During the 2014 Outbreak in West Africa – PLOS Currents Outbreaks." *PLOS Currents Outbreaks*. Web. 11 Mar. 2016.
[4]"SARS Reference | SARS Timeline." *SARS Reference | SARS Timeline*. Web. 11 Mar. 2016.
[5]"The SIR Model for Spread of Disease - The Differential Equation Model." *The SIR Model for Spread of Disease*. Web. 11 Mar. 2016.
[6] "Cumulative Number of Reported Probable Cases of SARS." *WHO*. Web. 13 Mar. 2016. <http://www.who.int/csr/sars/country/2003_07_09/en/>.
[7]"Cumulative Number of Reported Probable Cases of Severe Acute Respiratory Syndrome (SARS)." *WHO*. Web. 11 Mar. 2016.
[8]"WHO IRIS: Consensus Document on the Epidemiology of Severe Acute Respiratory Syndrome (SARS)." *WHO IRIS: Consensus Document on the Epidemiology of Severe Acute Respiratory Syndrome (SARS)*. Web. 11 Mar. 2016.
[9]Wallinga, J. "Different Epidemic Curves for Severe Acute Respiratory Syndrome Reveal Similar Impacts of Control Measures." *American Journal of Epidemiology* 160.6 (2004): 509-16. Web.
[10]"Epidemic." *Wikipedia*. Wikimedia Foundation. Web. 11 Mar. 2016.
[11]"Ebola Virus Disease." *World Health Organization*. Web. 11 Mar. 2016.
[12]Zhang, Zhibin. "The Outbreak Pattern of SARS Cases in China as Revealed by a Mathematical Model." *Ecological Modelling* 204.3-4 (2007): 420-26. Web.
[13]Zhou, Yicang, Zhien Ma, and F. Brauer. "A Discrete Epidemic Model for SARS Transmission and Control in China." *Mathematical and Computer Modelling* 40.13 (2004): 1491-506. Web.
[14]"Scipy.optimize.minimize¶." *Scipy.optimize.minimize — SciPy V0.17.0 Reference Guide*. Web. 11 Mar. 2016.
[15] Althaus, Christian L. "Estimating the Reproduction Number of Ebola Virus (EBOV)

During the 2014 Outbreak in West Africa." *PLoS Curr PLoS Currents* (2014). Web. 13 Mar. 2016.
[16] "Frequently Asked Questions About SARS." *Centers for Disease Control and Prevention*. Centers for Disease Control and Prevention, 2012. Web. 21 Mar. 2016. <http://www.cdc.gov/sars/about/faq.html>.
[17] "Ebola Virus Disease." *World Health Organization*. Web. 21 Mar. 2016. <http://www.who.int/mediacentre/factsheets/fs103/en/>.

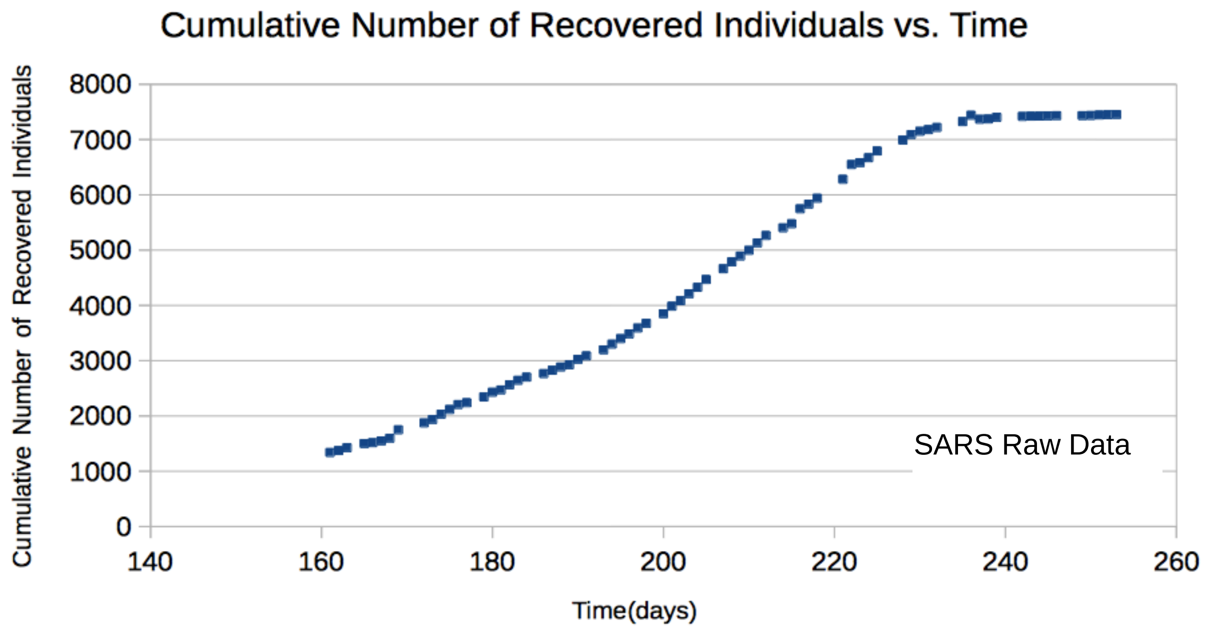
Special Thanks
James D. Berger for mentoring this project
Michael Gelbart for implementing Spearmint, suggesting fitting methods, and debugging
Pam Kalas for reviewing and advising this project
Omer Angel for providing insight on the data analysis
Costanza Piccolo for refining our parameters
Eric Cytrynbaum for refining the mechanics of the model

Appendix

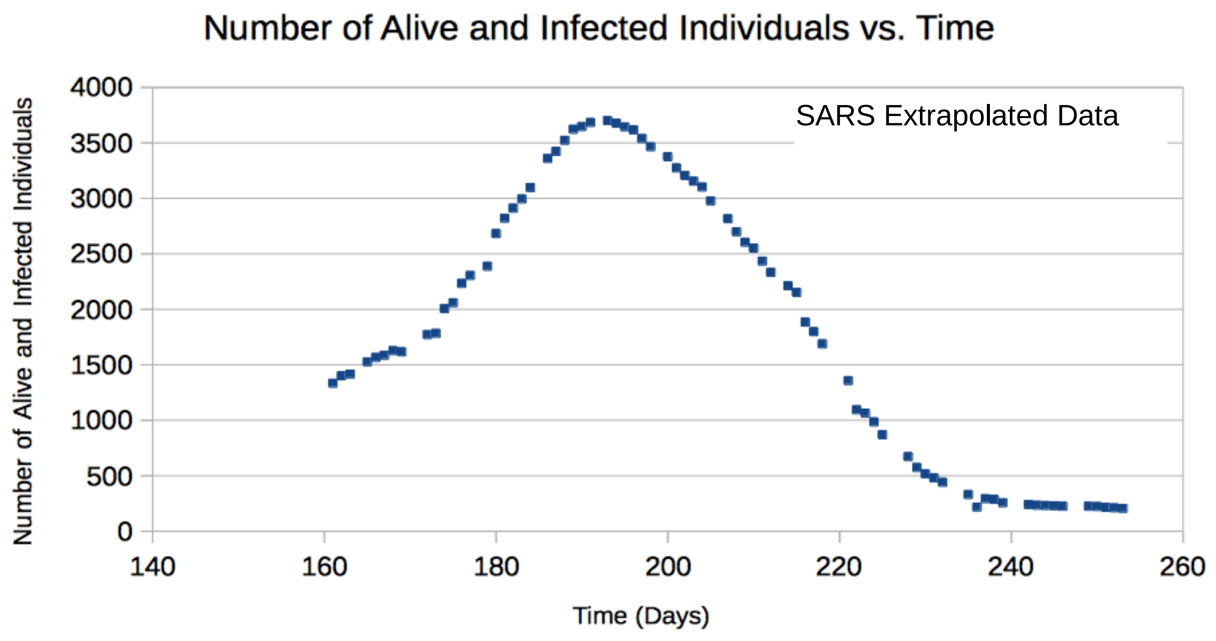
Table of Contents



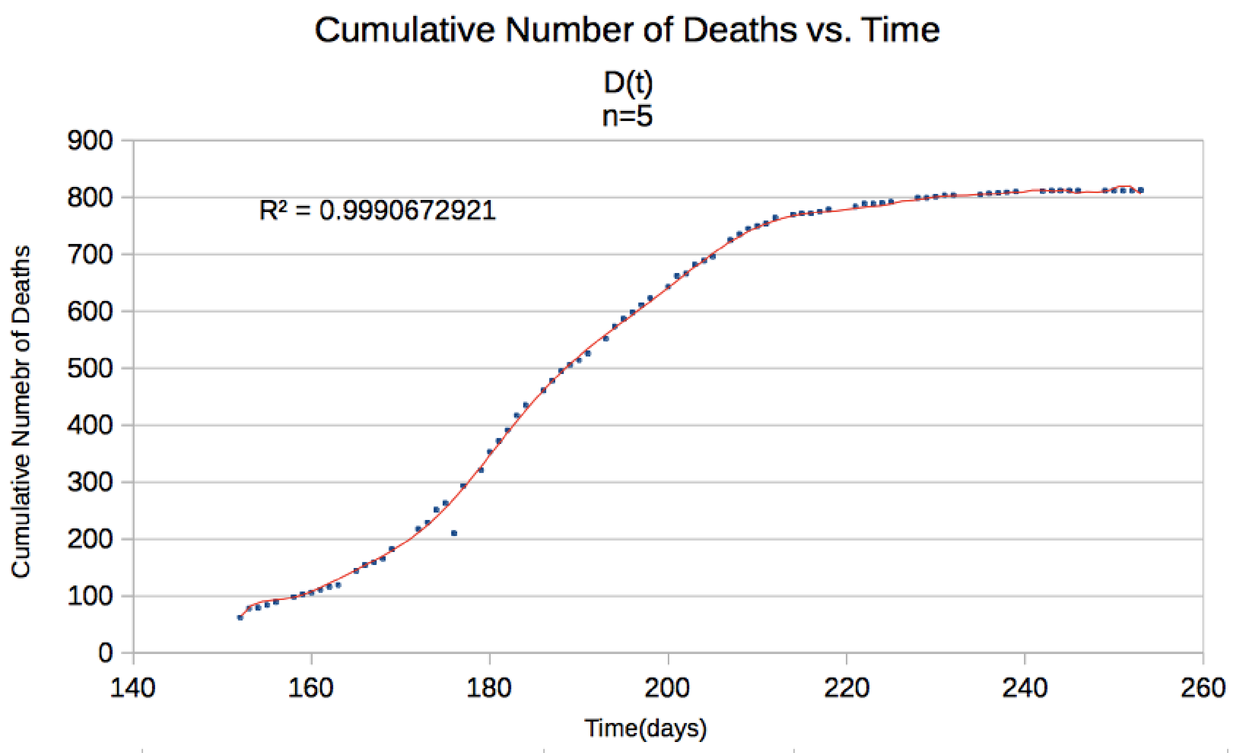
Appendix Fig1



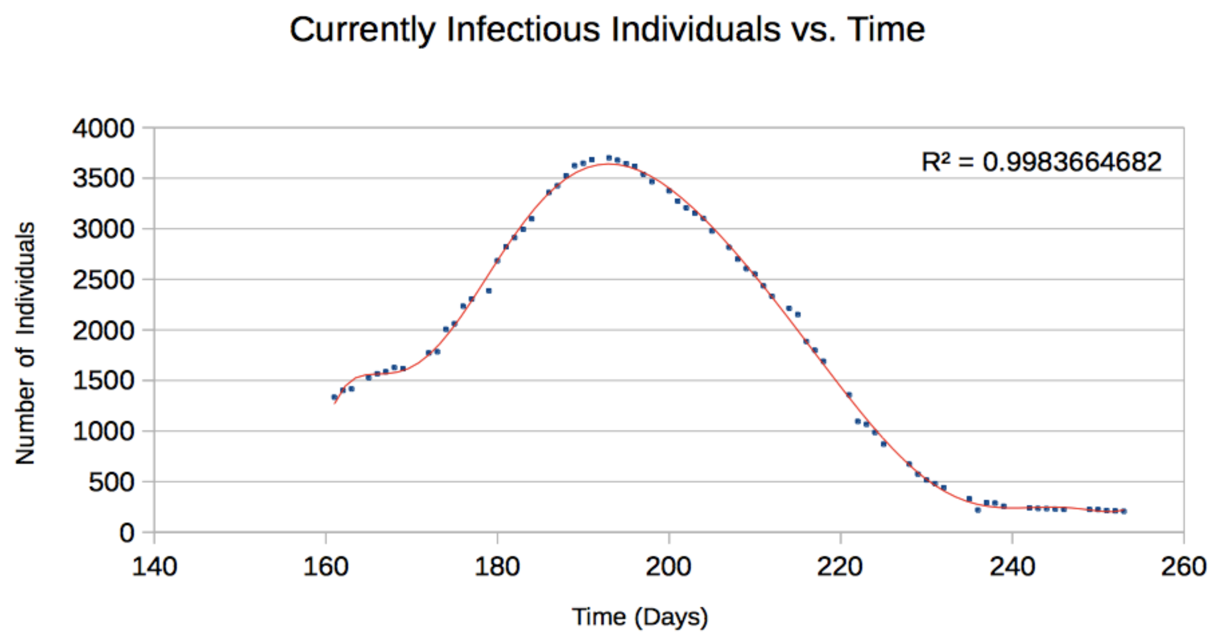
Appendix Fig2



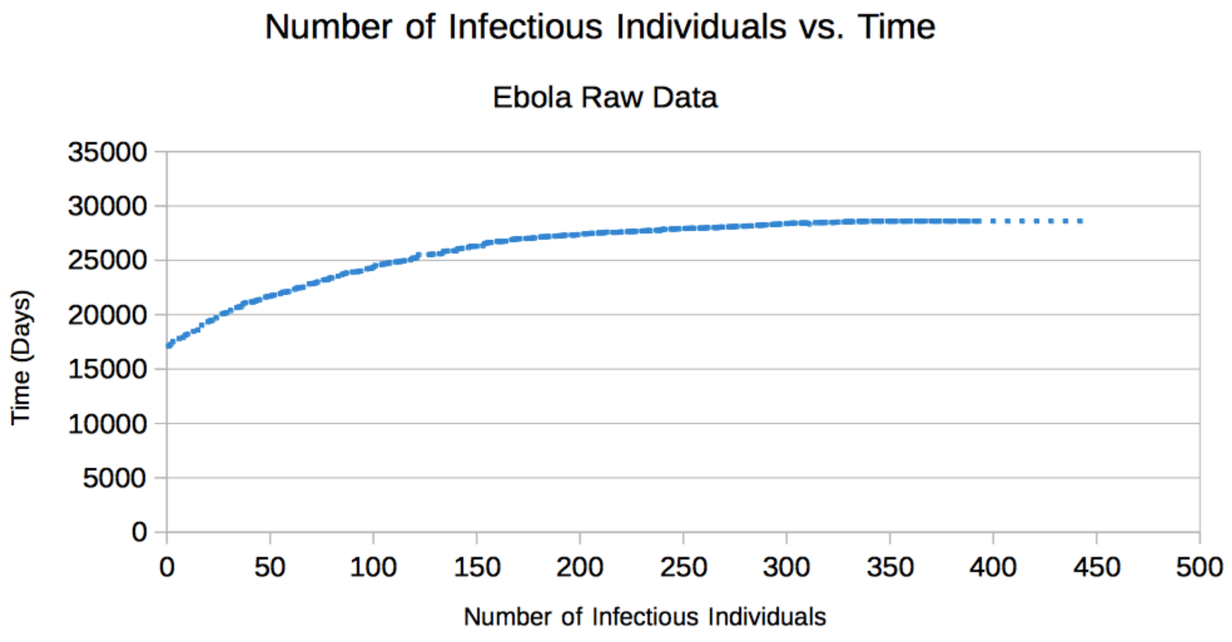
Appendix Fig3



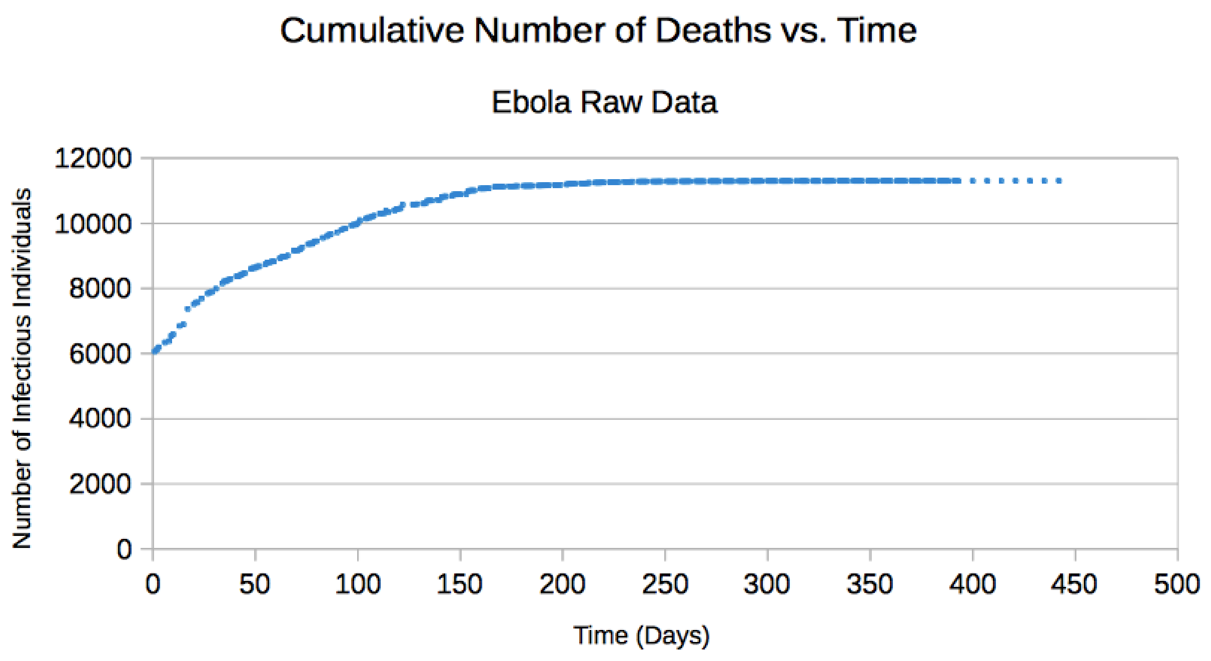
Appendix Fig4



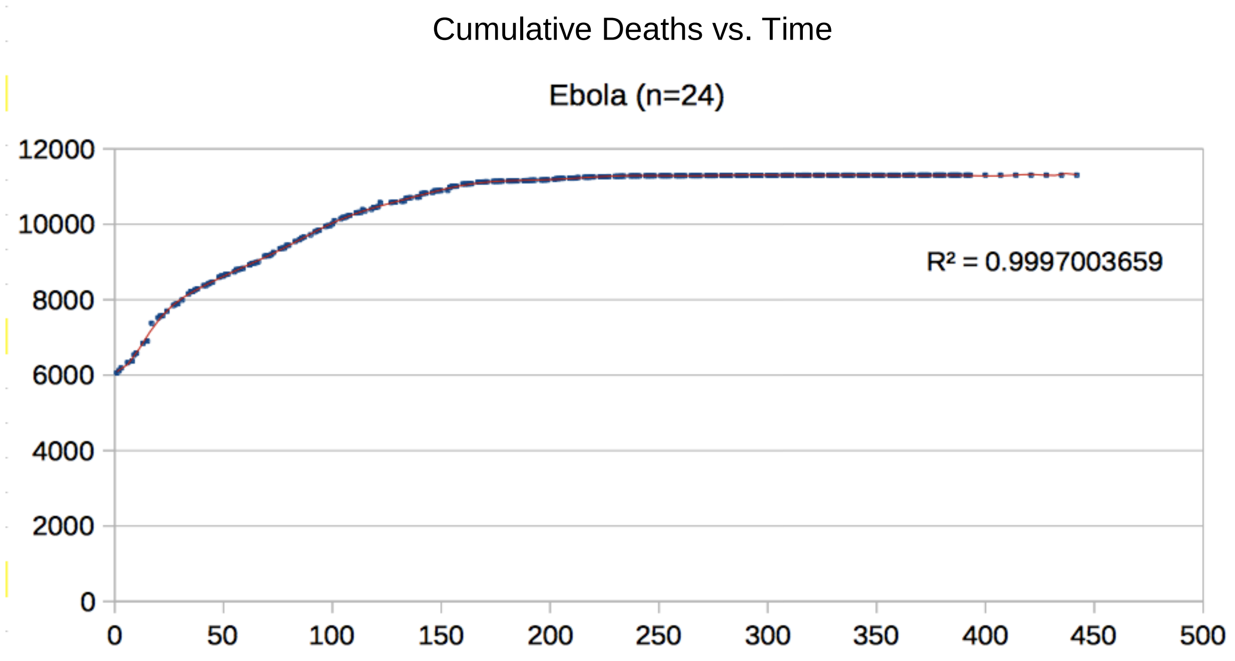
Appendix Fig5



Appendix Fig6

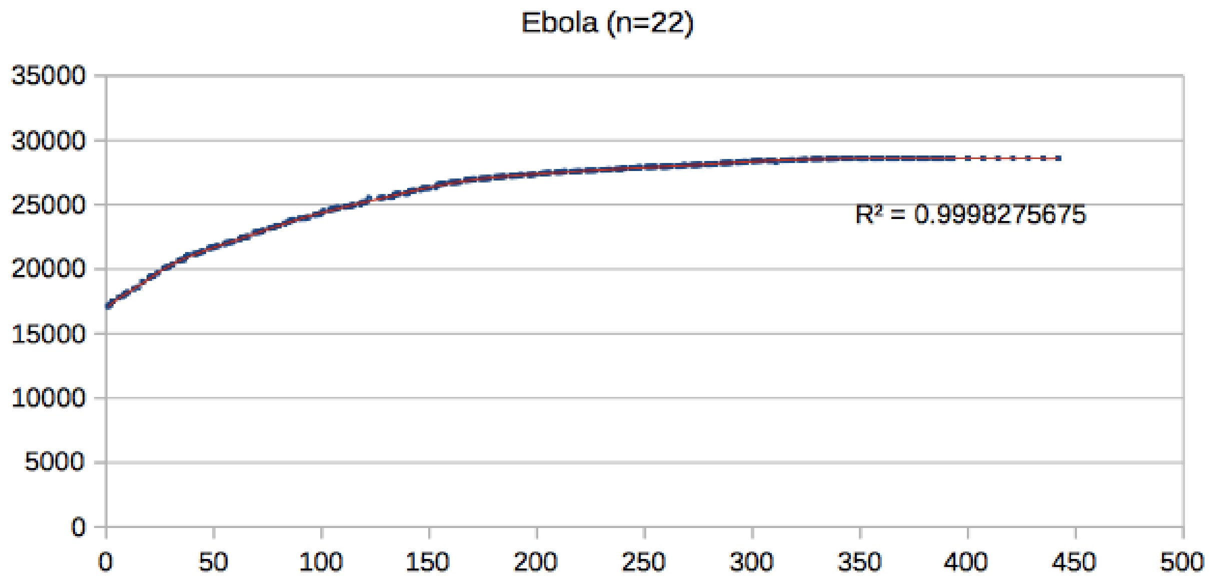


Appendix Fig7



Appendix Fig7

Cumulative Infectious vs. Time



Appendix Fig8

Ebola Parameter Sets									
	Infectivity	Infection Rate	Mobility	Incubation Rate	Mortality	Percentage Immunity	Recovery Rate	Population	ResidualSum
Average	0.99	0.001	0.9887100417	0.989113625	0.8238342917	0.0010002083	0.0010002083	3470330353.29167	0.000287125
Standard Deviation	0	0	0.0044474239	0.0030006973	0.0039407441	1.02062072615966E-06	1.02062072615966E-06	2126288880.69625	5.50345741132434E-06
Maximum Population								6866802093	
Minimum Population								729407782	
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	2218138142	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	2584033055	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	1901057942	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	3827085168	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	5857301234	0.000285
	0.99	0.001	0.99	0.982891	0.82749	0.001	0.001	6114060146	0.00029
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	930036007	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	729407782	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	801051285	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	6193655290	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	2750261610	0.000285
	0.99	0.001	0.977391	0.99	0.81404	0.001	0.001	3354761252	0.000299
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	6205577078	0.000285
CHOSEN	0.99	0.001	0.971673	0.989046	0.809716	0.001005	0.001005	1040879429	0.000307
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	1737874180	0.000285
	0.99	0.001	0.99	0.97679	0.830056	0.001	0.001	6085799484	0.000295
	0.99	0.001	0.989994	0.99	0.824532	0.001	0.001	6866802093	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	4768289329	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	2566033336	0.000285

	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	3319300965	0.000285
	0.99	0.001	0.989983	0.99	0.824523	0.001	0.001	1374802524	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	1356416950	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	6781393146	0.000285
	0.99	0.001	0.99	0.99	0.824537	0.001	0.001	3923911052	0.000285

SARS Parameter Sets												
	Infectivity	Infection Rate	Mobility	Incubation Period	Incubation Rate	Mortality	Percentage Immunity	Recovery Period	Recovery Rate	Population	Residual Sum	Estimated Basic Reproduction Number
Standard Deviation	0.3224714094	0.2345012131	0.097017819		0.2934826039	0.0495100597	0.0636622241		0.0496357966	2350778466.15791		1.7285763331
	0.486726	0.253035	0.866356	1.1178792489	0.894551	0.215118	0.029584	16.0815657012	0.062183	4930935930	0.456655	4.0691989772
Chosen Parameter set#1	0.851473	0.187879	0.868682	10.0161259628	0.099839	0.233785	0.024091	5.8622832421	0.170582	1959432004	0.492016	1.1013999132
	0.644418	0.26588	0.944756	1.489913287	0.67118	0.245621	0.02129	7.9605158414	0.12562	114133266	0.370468	2.1165419519
	0.200633	0.124275	0.796056	1.4747587663	0.678077	0.249025	0.07947	21.1671570391	0.047243	781370871	0.276411	2.630548441
	0.942965	0.970427	0.973074	1.0972913363	0.911335	0.265489	0.118805	6.096631611	0.164025	4384338355	0.980407	5.9163359244
	0.08657	0.255206	0.706624	1.0120136136	0.988129	0.283516	0.021446	20.461195343	0.048873	5630901875	0.317624	5.2218198187
	0.033523	0.142002	0.853813	1.7794385871	0.561975	0.322865	0.211555	7.2050781391	0.138791	1453683352	0.739969	1.0231355059
	0.245728	0.430487	0.826955	1.3681443994	0.730917	0.325093	0.117911	7.381544662	0.135473	3293913925	0.625295	3.1776590169
	0.533429	0.364678	0.873111	1.2384805825	0.807441	0.338197	0.001	8.5397825771	0.117099	1607023420	0.233318	3.1142708307
	0.533066	0.387292	0.907111	1.589327349	0.629197	0.339495	0.148519	15.5265037419	0.064406	5784967215	0.348509	6.0132906872
Chosen Parameter set#2	0.840515	0.539667	0.801637	13.8900463928	0.071994	0.347849	0.098526	5.1692943913	0.19345	6631318516	0.819369	2.7896975963
	0.095789	0.538563	0.62483	1.2243318821	0.816772	0.349846	0.055587	9.1958250954	0.108745	73865743	0.724964	4.9525311509

PHASE II PARAMETER SETS										
	Infectivity	Infectious Period	Infection Rate	Mobility	Incubation Period	Incubation Rate	Mortality	Percentage Immunity	Recovery Rate	Percentage Dead
Max	0.375849	1.6784099416	0.99	0.99	38.6085479325	0.375609	0.99	0.01	0.01	0.922588
Min	0.015306	1.0101010101	0.595802	0.99	2.6623430216	0.025901	0.99	0.01	0.01	0.922555
Average	0.1982362286	1.123803493	0.9053117143	0.99	7.7306120843	0.1864900571	0.99	0.01	0.01	0.9225818
Standard Deviation	0.0903404125	0.163870674	0.1104190317	0	7.0662311461	0.0847061018	0	0	0	8.62213294205885E-06
	0.260417	1.2797854056	0.781381	0.99	10.059147789	0.099412	0.99	0.01	0.01	0.922573
	0.104144	1.0101010101	0.99	0.99	3.3601360183	0.297607	0.99	0.01	0.01	0.922588
	0.189311	1.1429681741	0.874915	0.99	5.2823132306	0.189311	0.99	0.01	0.01	0.92258
	0.262111	1.1204130291	0.892528	0.99	8.3298625573	0.12005	0.99	0.01	0.01	0.922582
	0.141358	1.0555983659	0.94733	0.99	3.9715162852	0.251793	0.99	0.01	0.01	0.922585
	0.243455	1.3201633834	0.757482	0.99	8.9630632165	0.111569	0.99	0.01	0.01	0.922571
	0.076546	1.0101010101	0.99	0.99	3.0750023831	0.325203	0.99	0.01	0.01	0.922588
	0.28003	1.3154761826	0.760181	0.99	13.23872061	0.075536	0.99	0.01	0.01	0.922571
	0.158479	1.2698122456	0.787518	0.99	4.9362970861	0.202581	0.99	0.01	0.01	0.922573
	0.139883	1.0952806547	0.913008	0.99	4.0586392194	0.246388	0.99	0.01	0.01	0.922583
	0.303098	1.0101010101	0.99	0.99	10.1366419333	0.098652	0.99	0.01	0.01	0.922588
	0.210875	1.0101010101	0.99	0.99	5.2390307793	0.190875	0.99	0.01	0.01	0.922588
	0.358884	1.0185975541	0.981742	0.99	24.2824534991	0.041182	0.99	0.01	0.01	0.922588
	0.199946	1.0101010101	0.99	0.99	4.9553031654	0.201804	0.99	0.01	0.01	0.922588

	0.015306	1.6784099416	0.595802	0.99	3.2547951269	0.307239	0.99	0.01	0.01	0.922555
	0.197484	1.0425093617	0.959224	0.99	5.0488730915	0.198064	0.99	0.01	0.01	0.922586
	0.200874	1.0101010101	0.99	0.99	4.978245069	0.200874	0.99	0.01	0.01	0.922588
	0.1805	1.2800475154	0.781221	0.99	5.5774625892	0.179293	0.99	0.01	0.01	0.922573
	0.24139	1.0132061287	0.986966	0.99	6.2602119708	0.159739	0.99	0.01	0.01	0.922588
	0.183169	1.0124039735	0.987748	0.99	4.5845692568	0.218123	0.99	0.01	0.01	0.922588
	0.343787	1.0307875629	0.970132	0.99	18.5325895587	0.053959	0.99	0.01	0.01	0.922587
	0.154238	1.2546358796	0.797044	0.99	4.7907175058	0.208737	0.99	0.01	0.01	0.922574
	0.091475	1.5073748314	0.663405	0.99	4.0880233181	0.244617	0.99	0.01	0.01	0.922562
	0.200875	1.0101010101	0.99	0.99	4.9782202862	0.200875	0.99	0.01	0.01	0.922588
	0.32745	1.0101010101	0.99	0.99	13.458769061	0.074301	0.99	0.01	0.01	0.922588
	0.375849	1.0101010101	0.99	0.99	38.6085479325	0.025901	0.99	0.01	0.01	0.922588
	0.204574	1.0683144349	0.936054	0.99	5.3677153393	0.186299	0.99	0.01	0.01	0.922585
	0.211214	1.0777011772	0.927901	0.99	5.6163052574	0.178053	0.99	0.01	0.01	0.922584
	0.133185	1.3427378694	0.744747	0.99	4.5602338488	0.219287	0.99	0.01	0.01	0.92257
	0.234127	1.0205143801	0.979898	0.99	6.0392308436	0.165584	0.99	0.01	0.01	0.922587
	0.306	1.0101010101	0.99	0.99	10.4435370171	0.095753	0.99	0.01	0.01	0.922588
	0.041942	1.2324423186	0.811397	0.99	3.0871724896	0.323921	0.99	0.01	0.01	0.922575
	0.026141	1.0101010101	0.99	0.99	2.6623430216	0.375609	0.99	0.01	0.01	0.922588
	0.181518	1.0322111822	0.968794	0.99	4.6306586649	0.215952	0.99	0.01	0.01	0.922587
	0.158633	1.0106195907	0.989492	0.99	4.1150739273	0.243009	0.99	0.01	0.01	0.922588

```
"""
DETERMINISTIC EPIDEMIC MODEL PHASE I
Uma Wu & LiQing Wang

~Ver.7 Updates~
-began implementation of fitting model
---------------------------------------------------------------------------

BLOCK DIAGRAM

########                                   #############
#IMMUNE#<---                               #SUSCEPTIBLE#
########    --|------RecP------>#############
          ^    | -Imn----------------        |Mob ^RecP
      Imn|     |                              |       v       |
############<------------I---############
#INFECTIOUS#<--------IncP----#INCUBATING#
############-----InfP------> ############
   |Mor
   v
######
#DEAD#
######

#####PARAMETERS#####
I = Infectivity (percentage of people that move from INCUBATING to INFECTIOUS without IncP
[OVERRIDE])
InfP = Length of Infectious Period (rate of individuals moving from INFECTIOUS to INCUBATING)
Mob = Mobility (rate of individuals moving from SUSCEPTIBLE to INCUBATING)
IncP = Length of Incubation Period (rate of individuals moving from INCUBATING to
INFECTIOUS)
Mor = Morbidity (percentage of infectious individuals that move from INFECTIOUS to DEAD)
Imn = Immunity (percentage of recovered individuals that move from INFECTIOUS to IMMUNE)
RecP = Length of Recovery Period (rate of individuals moving from INFECTIOUS and
INCUBATING to SUSCEPTIBLE)

#Note: Incubating refers to both the disease being dormant and the disease displaying
nonconsequential symptoms.
Pop = Population (Total number of people in each compartment. Remains constant throughout
simulation)

#####INITIAL VALUES#####
Pop = Population
T = Time of Simulation in Days
```

```
IMMUNE = Pop*PImmune
SUSCEPTIBLE = Pop - Pop*PImmune
INCUBATING = 1
INFECTIOUS = 0
DEAD = 0

#####DIFFERENTIAL EQUATIONS FROM BLOCK#####
IMMUNE = IMMUNE + INFECTIOUS*Imn + INCUBATING*Imn
SUSCEPTIBLE = SUSCEPTIBLE + INFECTIOUS*RecP + INCUBATING*RecP -
SUSCEPTIBLE*(INCUBATING/Population)Mob
INCUBATING = INCUBATING + SUSCEPTIBLE*Mob + INFECTIOUS*InfP - INCUBATING*Imn -
INCUBATING*I - INCUBATING*IncP -INCUBATING*RecP
INFECTIOUS = INFECTIOUS + INCUBATING*I + INCUBATING*IncP - INFECTIOUS*Imn -
INFECTIOUS*InfP - INFECTIOUS*Mor
DEAD = DEAD + INFECTIOUS*Mor

"""
import numpy as np
import random
from scipy.optimize import minimize
import matplotlib.pyplot as plt

################FITTING##################

SusFit = False
IncFit = False
InfFit = False
ImmFit = False
DeaFit = False

if SusFit:
	SUSCEPTIBLEquery = []
	with open("SUSCEPTIBLEfit.csv", "r") as SUSCEPTIBLEfit:
		for line in SUSCEPTIBLEfit:
			SUSCEPTIBLEquery.append(tuple([int(x) for x in line[:-1].split(",")]))
	preset = SUSCEPTIBLEquery
	status = "SusFit"

if IncFit:
	INCUBATINGquery = []
	with open("INCUBATINGfit.csv", "r") as INCUBATINGfit:
		for line in INCUBATINGfit:
			INCUBATINGquery.append(tuple([int(x) for x in line[:-1].split(",")]))
	preset = INCUBATINGquery
```

```
        status = "IncFit"

if InfFit:
        INFECTIOUSquery = []
        with open("INFECTIOUSfit.csv", "r") as INFECTIOUSfit:
                for line in INFECTIOUSfit:
                        INFECTIOUSquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = INFECTIOUSquery
        status = "InfFit"

if ImmFit:
        IMMUNEquery = []
        with open("IMMUNEfit.csv", "r") as IMMUNEfit:
                for line in IMMUNEfit:
                        IMMUNEquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = IMMUNEquery
        status = "ImmFit"

if DeaFit:
        DEADquery = []
        with open("DEADfit.csv", "r") as DEADfit:
                for line in DEADfit:
                        DEADquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = DEADquery
        status = "DeaFit"

"""
INPUTS:
        time: the current time value
        y: the current value of the function
        query: the list of points (t, value) to fit against
OUTPUTS:
        dy: the deviation from the query
"""
fit = False
if fit:
        def Fit(time, y, query = preset):
                for i in query:
                        if time == i[0]:
                                dy = y - i[1] #Calculates the residuals
                                return dy
                        else:
                                return "NONE"
else:
        status = "SusFit"

###############SIMULATION#################

def DeterministicEPIMOD1(Parameters, T = 365, Plot = True, Print = True, fit = fit, status = status,
dt = 0.1):
        IMMUNE = np.zeros(int((T+1)/dt))
        SUSCEPTIBLE = np.zeros(int((T+1)/dt))
        INCUBATING = np.zeros(int((T+1)/dt))
        INFECTIOUS = np.zeros(int((T+1)/dt))
        DEAD = np.zeros(int((T+1)/dt))

        #PARAMETER VALUES (INITIAL VALUES)
        I = Parameters[0]
        InfP = Parameters[1]
        Mob = Parameters[2]
        IncP = Parameters[3]
        Mor = Parameters[4]
        Imn = Parameters[5]
        RecP = Parameters[6]

        Pop = Parameters[7]

        #INITIAL VALUES
        IMMUNE[0] = 0 #1213 #Pop*PImmune
        INCUBATING[0] =    1 #1516
        INFECTIOUS[0] = 0 #286
        DEAD[0] = 0
        SUSCEPTIBLE[0] = Pop - IMMUNE[0] - INCUBATING[0] - INFECTIOUS[0] - DEAD[0]

        #FITTING
        residuals = []
        n = 0

        for t in range(1, int(float(T)/dt+1)):
                SUSCEPTIBLEin = INFECTIOUS[t-1]*RecP*dt + INCUBATING[t-1]*RecP*dt
                SUSCEPTIBLEout = SUSCEPTIBLE[t-1]*(INCUBATING[t-1]/float(Pop))*Mob*dt
                SUSCEPTIBLE[t] = SUSCEPTIBLE[t-1] + SUSCEPTIBLEin - SUSCEPTIBLEout

                INCUBATINGin = SUSCEPTIBLE[t-1]*(INCUBATING[t-1]/float(Pop))*Mob*dt +
INFECTIOUS[t-1]*InfP*dt
                INCUBATINGout = INCUBATING[t-1]*Imn*dt + INCUBATING[t-1]*I*dt +
INCUBATING[t-1]*IncP*dt + INCUBATING[t-1]*RecP*dt
                INCUBATING[t] = INCUBATING[t-1] + INCUBATINGin - INCUBATINGout

                INFECTIOUSin = INCUBATING[t-1]*I*dt + INCUBATING[t-1]*IncP*dt
                INFECTIOUSout = INFECTIOUS[t-1]*Imn*dt + INFECTIOUS[t-1]*InfP*dt + INFECTIOUS[t-1]*Mor*dt + INFECTIOUS[t-1]*RecP*dt
                INFECTIOUS[t] = INFECTIOUS[t-1] + INFECTIOUSin - INFECTIOUSout

                DEADin = INFECTIOUS[t-1]*Mor*dt
                DEADout = 0
                DEAD[t] = DEAD[t-1] + DEADin - DEADout

                IMMUNEin = INFECTIOUS[t-1]*Imn*dt + INCUBATING[t-1]*Imn*dt
                IMMUNEout = 0
                IMMUNE[t] = IMMUNE[t-1] + IMMUNEin - IMMUNEout

                if fit == True:
                    if status == "SusFit":
                        y = SUSCEPTIBLE[t]
                    elif status == "IncFit":
                        y = INCUBATING[t]
                    elif status == "InfFit":
                        y = sum(INFECTIOUS)
                    elif status == "ImmFit":
                        y = IMMUNE[t]
                    elif status == "DeaFit":
                        y = DEAD[t]
                    else:
                        print "ERROR: I DON'T KNOW WHAT IT IS BUT SOMETHING IS NOT RIGHT."

                    diffs = Fit(int(t*dt), y)

                    if diffs != "NONE":
                        residuals.append(diffs**2)
                        # plt.plot(t, preset[n][1], "*r")
                        # plt.plot(t, y, ".b")
                        # print preset[n][1]
                        # n += 1

        if Print == True:
                Survivors = IMMUNE[t] + SUSCEPTIBLE[t] + INCUBATING[t] + INFECTIOUS[t]
                print "TOTAL POPULATION = %i" %Pop
                print "REMAINING POPULATION = %f" %Survivors
                print "IMMUNE = %.0f" %round(IMMUNE[t])
                print "SUSCEPTIBLE = %.0f" %round(SUSCEPTIBLE[t])
                print "INCUBATING = %.0f" %round(INCUBATING[t])
```

```
			print "INFECTIOUS = %.0f" %round(INFECTIOUS[t])
			print "DEAD = %.0f" %round(DEAD[t])

		if Plot == True:
			plt.plot(IMMUNE, label="IMMUNE")
			plt.plot(SUSCEPTIBLE, label="SUSCEPTIBLE")
			plt.plot(INCUBATING, label="INCUBATING")
			plt.plot(INFECTIOUS, label="INFECTIOUS")
			plt.plot(DEAD, label="DEAD")

			plt.legend(loc="best")
			plt.title("PHASEIIpandemic")
			plt.ylabel("Number of Individuals")
			plt.xlabel("Time/%f (in days)" %dt)

			#plt.show()
			plt.savefig("PHASEIIpandemicDET.pdf")

		if fit == True:
			return abs(sum(residuals))
		else:
			return -np.log(DEAD[t])
		#return abs(7452-(IMMUNE[-1])) + abs(831 - DEAD[-1])

##############EXECUTING SIMULATION###################

#Parameters = [0.8515,0.1879,0.8687,0.09984,0.2338,0.02409,0.1706,1959432004]
Parameters = [0.1982, 0.9053, 0.99, 0.1865, 0.99, 0.01, 0.01, 7000000000]
print np.exp(-DeterministicEPIMOD1(Parameters))

# print minimize(DeterministicEPIMOD1, [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 5000], method = 'L-BFGS-B', bounds = ((0.01, 1),(0.01, 1),(0.01, 1),(0.01, 1),(0.01, 1),(0.01, 1),(0.01, 1), (100, 7000000001)))

# if status == "SusFit":
#	data = open('DeterministicPHASEI_EBOLAsus.csv', 'w')
# elif status == "IncFit":
#	data = open('DeterministicPHASEI_EBOLAinc.csv', 'w')
# elif status == "InfFit":
#	data = open('DeterministicPHASEI_EBOLAinf.csv', 'w')
# elif status == "ImmFit":
#	data = open('DeterministicPHASEI_EBOLAimm.csv', 'w')
# elif status == "DeaFit":
#	data = open('DeterministicPHASEI_EBOLAdea.csv', 'w')
```

```
# else:
#	print "ERROR: I DON'T KNOW WHAT IT IS BUT SOMETHING IS NOT RIGHT."

# data.write("Infectivity, InfectionRate, Mobility, IncubationRate, Mortality, PercentageImmunity, RecoveryRate, Population, ResidualSum")

# i = 0
# trials = 10000
# for n in range(trials):
#	print "\n"
#	print n
#	guess = np.random.rand(8)
#	guess[7] = random.randint(100, 7000000001)
#	Combination = minimize(DeterministicEPIMOD1, guess, method = 'L-BFGS-B', bounds = ((0.001, 0.99),(0.001, 0.99),(0.001, 0.99),(0.001, 0.99),(0.001, 0.99),(0.001, 0.99),(0.001, 0.99), (100, 7000000001)))
#	if Combination.fun <= 1:
#		i += 1
#		print "%i candidates found!" %i
#		I = Combination.x[0]
#		InfP = Combination.x[1]
#		Mob = Combination.x[2]
#		IncP = Combination.x[3]
#		Mor = Combination.x[4]
#		Imn = Combination.x[5]
#		RecP = Combination.x[6]
#		Population = Combination.x[7]
#		Residual = Combination.fun
#		data.write("\n" + "%f, %f, %f, %f, %f, %f, %f, %f, %f" %(I, InfP, Mob, IncP, Mor, Imn, RecP, Population, Residual**2))
```

```
"""
DETERMINISTIC EPIDEMIC MODEL
Uma Wu & LiQing Wang

Discussed With: Michael Gelbart

~Ver.8 Updates~
-Began implementation of Spearmint
-Removed normalization
---------------------------------------------------------------------------

BLOCK DIAGRAM

########                                 #############
#IMMUNE#<---                              #SUSCEPTIBLE#
########    --|------RecP------>#############
          ^    | -Imn----------------        |Mob ^RecP
      Imn|     |                               |       v       |
############<------------I---############
#INFECTIOUS#<--------IncP----#INCUBATING#
############-----InfP------> ############
   |Mor
   v
######
#DEAD#
######

#####PARAMETERS#####
PImmune = Pre-Immunity (percentage of population immune to disease at initial time)
I = Infectivity (percentage of people that move from INCUBATING to INFECTIOUS without IncP
[OVERRIDE])
InfP = Length of Infectious Period (rate of individuals moving from INFECTIOUS to INCUBATING)
Mob = Mobility (rate of individuals moving from SUSCEPTIBLE to INCUBATING)
IncP = Length of Incubation Period (rate of individuals moving from INCUBATING to
INFECTIOUS)
Mor = Morbidity (percentage of infectious individuals that move from INFECTIOUS to DEAD)
Imn = Immunity (percentage of recovered individuals that move from INFECTIOUS to IMMUNE)
RecP = Length of Recovery Period (rate of individuals moving from INFECTIOUS and
INCUBATING to SUSCEPTIBLE)
#Note: Incubating refers to both the disease being dormant and the disease displaying
nonconsequential symptoms.

#####INITIAL VALUES#####
Pop = Population
```

```
T = Time of Simulation in Days
IMMUNE = Pop*PImmune
SUSCEPTIBLE = Pop - Pop*PImmune
INCUBATING = 0
INFECTIOUS = 1
DEAD = 0

#####DIFFERENTIAL EQUATIONS FROM BLOCK#####
IMMUNE = IMMUNE + INFECTIOUS*Imn + INCUBATING*Imn
SUSCEPTIBLE = SUSCEPTIBLE + INFECTIOUS*RecP - SUSCEPTIBLE*Mob
INCUBATING = INCUBATING + SUSCEPTIBLE*Mob + INFECTIOUS*InfP - INCUBATING*Imn - INCUBATING*I - INCUBATING*IncP
INFECTIOUS = INFECTIOUS + INCUBATING*I + INCUBATING*IncP - INFECTIOUS*Imn - INFECTIOUS*InfP - INFECTIOUS*Mor
DEAD = DEAD + INFECTIOUS*Mor

"""

import numpy as np
from scipy.optimize import minimize
import scipy.optimize

def main(job_id, params):
    Parameters = [params["I"], params["InfP"], params["Mob"], params["IncP"], params["Mor"], params["Imn"], params["RecP"]]
    return -StochasticEPIMOD(Parameters)

def DeterministicEPIMOD(Parameters, T = 365, Pop = 7000000000):
    dt = 0.1
    IMMUNE = np.zeros(int((T+1)/dt))
    SUSCEPTIBLE = np.zeros(int((T+1)/dt))
    INCUBATING = np.zeros(int((T+1)/dt))
    INFECTIOUS = np.zeros(int((T+1)/dt))
    DEAD = np.zeros(int((T+1)/dt))

    #PARAMETER VALUES (INITIAL VALUES)
    I = Parameters[0] #0.1 #0.011 is the rate of infection from data
    InfP = Parameters[1] #0.12 #max 0.12
    Mob = Parameters[2] #11.0/30.0 #THIS SHOULD DEPEND ON THE NUMBER OF INFECTED INDIVIDUALS SOMEHOW AHHH
    IncP = Parameters[3] #7.0/15.0
    Mor = Parameters[4] #0.07      #0.096 is the death rate from data
    Imn = Parameters[5] #0.397 #1.0 - Mor
    RecP = Parameters[6] #0.468 is the recovery rate from data
```

```
	#INITIAL VALUES
	IMMUNE[0] = 0 #1213 #Pop*PImmune
	INCUBATING[0] =    1 #1516
	INFECTIOUS[0] = 0 #286
	DEAD[0] = 0
	SUSCEPTIBLE[0] = Pop - IMMUNE[0] - INCUBATING[0] - INFECTIOUS[0] - DEAD[0]


	for t in range(1, int(float(T)/dt+1)):
		SUSCEPTIBLEin = INFECTIOUS[t-1]*RecP*dt + INCUBATING[t-1]*RecP*dt
		SUSCEPTIBLEout = SUSCEPTIBLE[t-1]*(INCUBATING[t-1]/float(Pop))*Mob*dt
		SUSCEPTIBLE[t] = SUSCEPTIBLE[t-1] + SUSCEPTIBLEin - SUSCEPTIBLEout

		INCUBATINGin = SUSCEPTIBLE[t-1]*(INCUBATING[t-1]/float(Pop))*Mob*dt +
INFECTIOUS[t-1]*InfP*dt
		INCUBATINGout = INCUBATING[t-1]*Imn*dt + INCUBATING[t-1]*I*dt +
INCUBATING[t-1]*IncP*dt + INCUBATING[t-1]*RecP*dt
		INCUBATING[t] = INCUBATING[t-1] + INCUBATINGin - INCUBATINGout

		INFECTIOUSin = INCUBATING[t-1]*I*dt + INCUBATING[t-1]*IncP*dt
		INFECTIOUSout = INFECTIOUS[t-1]*Imn*dt + INFECTIOUS[t-1]*InfP*dt +
INFECTIOUS[t-1]*Mor*dt + INFECTIOUS[t-1]*RecP*dt
		INFECTIOUS[t] = INFECTIOUS[t-1] + INFECTIOUSin - INFECTIOUSout

		DEADin = INFECTIOUS[t-1]*Mor*dt
		DEADout = 0
		DEAD[t] = DEAD[t-1] + DEADin - DEADout

		IMMUNEin = INFECTIOUS[t-1]*Imn*dt + INCUBATING[t-1]*Imn*dt
		IMMUNEout = 0
		IMMUNE[t] = IMMUNE[t-1] + IMMUNEin - IMMUNEout

	return -np.log(DEAD[t])

def Normalize(Parameters):
	return np.sum(Parameters) - 1

###############EXECUTING SIMULATION####################

#print minimize(DeterministicEPIMOD, [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1], method = 'L-BFGS-B',
bounds = ((0.01, 1),)*7)
#print scipy.optimize.fmin_slsqp(DeterministicEPIMOD, [0.3, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1],
bounds = ((0.01, 1),)*7, f_eqcons = Normalize)
```

```
#returns the seven best parameters if not printed

data = open('DeterministicPHASEII2.csv', 'w')
data.write("Infectivity, InfectionRate, Mobility, IncubationRate, Mortality, PercentageImmunity, RecoveryRate, PercentageDead")

trials = 50000
for n in range(trials):
	print n
	guess = np.random.rand(7)
	Combination = minimize(DeterministicEPIMOD, guess, method = 'L-BFGS-B', bounds = ((0.01, 0.99),)*7)
	I = Combination.x[0]
	InfP = Combination.x[1]
	Mob = Combination.x[2]
	IncP = Combination.x[3]
	Mor = Combination.x[4]
	Imn = Combination.x[5]
	RecP = Combination.x[6]
	Dead = np.exp(-DeterministicEPIMOD(Combination.x))/7000000000.0
	data.write("\n" + "%f, %f, %f, %f, %f, %f, %f, %f" %(I, InfP, Mob, IncP, Mor, Imn, RecP, Dead))

# TESTING PURPOSES ONLY
# guess = np.random.rand(7)
# print scipy.optimize.fmin_slsqp(DeterministicEPIMOD, guess, bounds = ((0.01, 1),)*7, f_eqcons = Normalize)
```

```
"""
STOCHASTIC EPIDEMIC MODEL
Uma Wu & LiQing Wang

Discussed With: Noah Bayless, Michael Gelbart
---------------------------------------------------------------------------
#####PATIENT INFORMATION#####
xLoc = range(0, x_Dim)
yLoc = range(0, y_Dim)
Status = IMMUNE, SUSCEPTIBLE, INFECTIOUS, INCUBATING, or DEAD
tExist = time elapsed since start of simulation
tInc = time incubating
tInf = time infectious
tRec = time recovering

#####PARAMETERS#####
PImmune = Pre-Immunity (Probable Percentage of Population that is Pre-Immune to Disease,
float)
I = Infectivity (Percentage Probability of Displaying Symptoms When Infected, float)
InfP = Length of Infectious Period (Time Range in which the Disease is Infectious, array)
Mob = Mobility (Percentage Probability of Contracting the Disease from an Adjacent Diseased
Individual, float)
IncP = Length of Incubation Period (Time Range in which the Disease does not produce
Symptoms (In an Infective Individual), array)
Mor = Morbidity (Percentage Probability of Infective Individuals Dying from the Disease, float)
Imn = Immunity (Percentage Probability of Individuals of Gaining Immunity After Recovery,
float)
RecP = Length of Recovery Period (Time Range in which the Disease is Present in an Individual,
array)
"""

########################DEFINING PATIENT
CLASS####################################

import random
import matplotlib.pyplot as plt
import numpy as np

class Dossier(): #Creates an object (like list or array) that has its own modules
	def __init__(self, x, y, PImmune, IncP, InfP, RecP): #Automatically run as soon as object is
created
		#x and y - int
		#PImmune - float
		#IncP, InfP and RecP - lists
```

```
        self.xLoc = x #Stores the inputted x location in the object
        self.yLoc = y #Same as above

        vaccination = random.random()
        if vaccination <= PImmune: #If the "roll" is below the PImmune probability
            self.Status = "IMMUNE" #The Individual becomes immune
        else: self.Status = "SUSCEPTIBLE" #If not, the individual becomes susceptible

        self.IncLen = random.choice(IncP) #Picks a incubation rate out of the IncP list
        self.InfLen = random.choice(InfP) #Same as above for infectious rate
        self.RecLen = random.choice(RecP) #Same as above for recovery rate
        self.IncState = False #Sets incubation state as false
        self.InfState = False #Same as above for infectious rate
        self.RecState = False #Same as above for recovery rate


    def Incubation(self, I, Imn):
        self.IncLen -= 1 #When the function is called, it first decreases the incubation period by 1
        if self.IncLen >= 0: #While the incubation period is more than 0, the IncState is set to True
            self.IncState = True
        else: self.IncState = False #Otherwise, the IncState is False. In other words, the patient stops incubating.

        if self.IncState == False: #Once the patient finishes incubating..
            luck = random.random()
            if luck <= I: #If their "roll" is less than Infectivity...
                self.Status = "INFECTIOUS" #They become Infectious
            else:
                antibody = random.random() #If their "roll" is more than Infectivity...
                if antibody <= Imn: #If their "roll" is smaller than the Immunity
                    self.Status = "IMMUNE" #They become Immune.
                else: self.Status = "SUSCEPTIBLE" #Otherwise, they become susceptible.


    def Infection(self, Mor): #Same concepts as incubation.
        self.InfLen -= 1
        if self.InfLen >= 0:
            self.InfState = True
        else: self.InfState = False

        if self.InfState == False:
            hopesndreams = random.random()
            if hopesndreams <= Mor:
                self.Status = "DEAD"
            else: self.Status = "INCUBATING"


    def Recovery(self, Imn): #Activated when patient goes into incubation or infection
        self.RecLen -= 1
        if self.RecLen >= 0:
            self.RecState = True
        else: self.RecState = False

        if self.RecState == False:
            antibody = random.random()
            if antibody <= Imn:
                self.Status = "IMMUNE"
            else: self.Status = "SUSCEPTIBLE"

    def Copy(self):
        new_pat = Dossier(self.xLoc, self.yLoc, 0, range(2,7), range(2,7), range(2,7))
        #^These input values doesn't matter cuz we're gonna update it anyway
        new_pat.Status = self.Status
        new_pat.IncLen = self.IncLen
        new_pat.InfLen = self.InfLen
        new_pat.RecLen = self.RecLen
        new_pat.IncState = self.IncState
        new_pat.InfState = self.InfState
        new_pat.RecState = self.RecState
        return new_pat

##############################DEFINING FUNCTIONS####################################

def Spread(Mob, NBR):
    if NBR.Status == "INFECTIOUS":
        contagion = random.random()
        if contagion <= Mob: #If the "roll" is lower than mobility, then the patient is infected by the virus
            return True
        else:
            return False
    return False

def Plot(Patient, Population):
    for P in range(1, Population+1):
        x = Patient[P].xLoc
        y = Patient[P].yLoc

        if Patient[P].Status == 'SUSCEPTIBLE':
            colour = "white"
        elif Patient[P].Status == 'IMMUNE':
            colour = "blue"
        elif Patient[P].Status == 'INFECTIOUS':
            colour = "red"
        elif Patient[P].Status == 'INCUBATING':
            colour = "yellow"
        elif Patient[P].Status == 'DEAD':
            colour = "black"

        if colour == "white":
            plt.plot(x, y, "wo")
        elif colour == "blue":
            plt.plot(x, y, "bo")
        elif colour == "red":
            plt.plot(x, y, "ro")
        elif colour == "yellow":
            plt.plot(x, y, "yo")
        elif colour == "black":
            plt.plot(x, y, "ko")
    plt.show()

def DeepCopy(Old): #Takes in dict, returns deep copy
    New = dict()
    for i in range(1, len(Old)+1):
        New[i] = Old[i].Copy()
    return New

########################FITTING
FUNCTIONS######################################
if __name__ == "__main__":
    SusFit = False
    IncFit = False
    InfFit = False
    ImmFit = False
    DeaFit = False

    if SusFit:
        SUSCEPTIBLEquery = []
        with open("SUSCEPTIBLEfit.csv", "r") as SUSCEPTIBLEfit:
            for line in SUSCEPTIBLEfit:
                    SUSCEPTIBLEquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = SUSCEPTIBLEquery
        status = "SusFit"

    if IncFit:
        INCUBATINGquery = []
        with open("INCUBATINGfit.csv", "r") as INCUBATINGfit:
            for line in INCUBATINGfit:
                    INCUBATINGquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = INCUBATINGquery
        status = "IncFit"

    if InfFit:
        INFECTIOUSquery = []
        with open("INFECTIOUSfit.csv", "r") as INFECTIOUSfit:
            for line in INFECTIOUSfit:
                    INFECTIOUSquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = INFECTIOUSquery
        status = "InfFit"

    if ImmFit:
        IMMUNEquery = []
        with open("IMMUNEfit.csv", "r") as IMMUNEfit:
            for line in IMMUNEfit:
                    IMMUNEquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = IMMUNEquery
        status = "ImmFit"

    if DeaFit:
        DEADquery = []
        with open("DEADfit.csv", "r") as DEADfit:
            for line in DEADfit:
                    DEADquery.append(tuple([int(x) for x in line[:-1].split(",")]))
        preset = DEADquery
        status = "DeaFit"

    """
    INPUTS:
        time: the current time value
        y: the current value of the function
        query: the list of points (t, value) to fit against
    OUTPUTS:
        dy: the deviation from the query
        """

        fit = False
        if fit == True:
                def Fit(time, y, query = preset):
                        for i in query:
                                if time == i[0]:
                                        dy = y - i[1]
                                else:
                                        dy = "NONE"
                        return dy
else:
        fit = False


############################INITIALIZE
SIMULATION################################
def main(job_id, params):
        Parameters = [params["I"], params["InfP"], params["Mob"], params["IncP"],
params["Mor"], params["Imn"], params["RecP"]]
        return StochasticEPIMOD(Parameters)


def StochasticEPIMOD(Parameters, xDimension = 20, yDimension = 20, tElapsed = 365, flux =
True, move_range = 0.5, Plot = True, Print = True, fit = fit):
        #INITIAL VALUES
        Population = xDimension*yDimension

        #PARAMETERS
        PImmune = 0.0
        I = Parameters[0] #Percentage
        InfP = range(int(Parameters[1])-3, int(Parameters[1])+4)
        Mob = Parameters[2] #Percentage
        IncP = range(int(Parameters[3])-3, int(Parameters[3])+4)
        Mor = Parameters[4] #Percentage
        Imn = Parameters[5] #Percentage
        RecP = range(int(Parameters[6])-3, int(Parameters[6])+4)

        #PATIENTS
        Patient = dict()
        ID = 0
        Pos2Pat = np.zeros((yDimension, xDimension))
        for x in range(xDimension):
                for y in range(yDimension):
```

```
            ID += 1
            Patient[ID] = Dossier(x, y, PImmune, IncP, InfP, RecP)
            Pos2Pat[y,x] = ID

    #GENERATING PATIENT ZERO
    x = xDimension/2
    y = yDimension/2
    for P in range(1, ID+1):
        if Patient[P].xLoc == x and Patient[P].yLoc == y:
            Patient[P].Status = "INFECTIOUS"

    Record = DeepCopy(Patient)

    SUSCEPTIBLEtrend = []
    INCUBATINGtrend = []
    INFECTIOUStrend = []
    IMMUNEtrend = []
    DEADtrend = []

    residuals = []

    ###########################SIMULATION BEGINS####################################
    for t in range(tElapsed):
        # Plot(Patient, Population)
        for P in range(1,ID+1):
            #STATUS PRIORITY: DEAD = IMMUNE > INFECTIOUS = INCUBATING > SUSCEPTIBLE
            if Record[P].Status == "SUSCEPTIBLE":
                n = 8
                x = Patient[P].xLoc
                y = Patient[P].yLoc
                while Patient[P].Status != "INFECTIOUS" and Patient[P].Status != "INCUBATING" and n > 0:
                    if n == 8:
                        infection = Spread(Mob, Record[Pos2Pat[y,(x+1)%xDimension]]) #Right
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 7:
                        infection = Spread(Mob, Record[Pos2Pat[y,(x-1)%xDimension]])
```

```
#Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 6:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,x]])
#Above
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 5:
                        infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,x]])
#Below
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 4:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,(x+1)%xDimension]]) #Upper Right
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 3:
                        infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,(x-1)%xDimension]]) #Lower Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 2:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,(x-1)%xDimension]]) #Upper Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
```

```
                            Patient[P].Status = "INFECTIOUS"
                        else: Patient[P].Status = "INCUBATING"
                elif n == 1:
                    infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,(x+1)%xDimension]]) #Lower Right
                    if infection == True:
                        infection = random.random()
                        if infection <= I:
                            Patient[P].Status = "INFECTIOUS"
                        else: Patient[P].Status = "INCUBATING"
                n -= 1

        elif Record[P].Status == "INFECTIOUS":
            Patient[P].Recovery(Imn)
            if Patient[P].RecState == True: #If the patient is still infectious...
                Patient[P].Infection(Mor)

        elif Record[P].Status == "INCUBATING":
            Patient[P].Recovery(Imn)
            if Patient[P].RecState == True: #If the patient is still incubating...
                Patient[P].Incubation(I,Imn)

        elif Record[P].Status == "IMMUNE": #Removed from population
            pass
        elif Record[P].Status == "DEAD": #Removed from population
            pass


    #UPDATE MOVING
    if flux == True:
        for x in range(xDimension):
            for y in range(yDimension):
                if Patient[Pos2Pat[y,x]].Status != "DEAD":
                    move = random.random()
                    if move <= move_range:
                        swap = random.randint(1, 4)
                        if swap == 1: #switch with above
                            oriID = Pos2Pat[y, x]
                            newID = Pos2Pat[(y+1)%yDimension, x]
                            Patient[oriID].yLoc = (y+1)%yDimension
                            Patient[newID].yLoc = y
                            Pos2Pat[y,x] = newID
                            Pos2Pat[(y+1)%yDimension, x] = oriID

                        elif swap == 2: #switch with below
                            oriID = Pos2Pat[y, x]
                            newID = Pos2Pat[(y-1)%yDimension, x]
                            Patient[oriID].yLoc = (y-1)%yDimension
                            Patient[newID].yLoc = y
                            Pos2Pat[y,x] = newID
                            Pos2Pat[(y-1)%yDimension, x] = oriID

                        elif swap == 3: #switch with left
                            oriID = Pos2Pat[y, x]
                            newID = Pos2Pat[y, (x-1)%yDimension]
                            Patient[oriID].xLoc = (x-1)%xDimension
                            Patient[newID].xLoc = x
                            Pos2Pat[y,x] = newID
                            Pos2Pat[y, (x-1)%yDimension] = oriID

                        elif swap == 4: #switch with right
                            oriID = Pos2Pat[y, x]
                            newID = Pos2Pat[y, (x+1)%yDimension]
                            Patient[oriID].xLoc = (x+1)%xDimension
                            Patient[newID].xLoc = x
                            Pos2Pat[y,x] = newID
                            Pos2Pat[y, (x+1)%yDimension] = oriID
    SUSCEPTIBLE = 0
    INFECTIOUS = 0
    INCUBATING = 0
    IMMUNE = 0
    DEAD = 0

    for P in range(1, ID+1):
        if Patient[P].Status == "SUSCEPTIBLE":
            SUSCEPTIBLE += 1
        elif Patient[P].Status == "INFECTIOUS":
            INFECTIOUS += 1
        elif Patient[P].Status == "INCUBATING":
            INCUBATING += 1
        elif Patient[P].Status == "IMMUNE":
            IMMUNE += 1
        elif Patient[P].Status == "DEAD":
            DEAD += 1

    SUSCEPTIBLEtrend.append(SUSCEPTIBLE)
    INCUBATINGtrend.append(INCUBATING)
    INFECTIOUStrend.append(INFECTIOUS)
```

```
        IMMUNEtrend.append(IMMUNE)
        DEADtrend.append(DEAD)

        if fit == True:
            if status == "SusFit":
                y = SUSCEPTIBLE
            elif status == "IncFit":
                y = INCUBATING
            elif status == "InfFit":
                y = INFECTIOUS
            elif status == "ImmFit":
                y = IMMUNE
            elif status == "DeaFit":
                y = DEAD

            diffs = Fit(t, y)
            if diffs != "NONE":
                residuals.append(diffs**2)

        Record = DeepCopy(Patient)

    if Plot:
        plt.plot(SUSCEPTIBLEtrend, "0.5", label = "SUSCEPTIBLE")
        plt.plot(INCUBATINGtrend, "y", label = "INCUBATING")
        plt.plot(INFECTIOUStrend, "r", label = "INFECTIOUS")
        plt.plot(IMMUNEtrend, "b", label = "IMMUNE")
        plt.plot(DEADtrend, "k", label = "DEAD")

        plt.legend(loc="best")
        plt.title("ebolaSTOinfSIM")
        plt.ylabel("Number of Individuals")
        plt.xlabel("Time (in days)")
        # plt.show()
        plt.savefig("ebolaSTOinfSIM2.pdf")

    if Print:
        print "SUSCEPTIBLE = %i" %SUSCEPTIBLE
        print "INFECTIOUS = %i" %INFECTIOUS
        print "INCUBATING = %i" %INCUBATING
        print "IMMUNE = %i" %IMMUNE
        print "DEAD = %i" %DEAD
        print "Percent Dead = %f" %(float(DEAD)/float(Population))

    return abs(sum(residuals))
```

```
        # PercentDead = float(DEAD)/float(Population)
        # return PercentDead

if __name__ == "__main__":
        StochasticEPIMOD([0.6834, 14, 0.0163, 19, 0.9512, 0.8861, 22])
```

```
"""
STOCHASTIC EPIDEMIC MODEL
Uma Wu & LiQing Wang

Discussed With: Noah Bayless, Michael Gelbart
-----------------------------------------------------------------------------
#####PATIENT INFORMATION#####
xLoc = range(0, x_Dim)
yLoc = range(0, y_Dim)
Status = IMMUNE, SUSCEPTIBLE, INFECTIOUS, INCUBATING, or DEAD
tExist = time elapsed since start of simulation
tInc = time incubating
tInf = time infectious
tRec = time recovering


#####PARAMETERS#####
PImmune = Pre-Immunity (PERCENTAGE PROBABILITY of Population that is Pre-Immune to Disease, float)
I = Infectivity (PERCENTAGE PROBABILITY of Displaying Symptoms When Infected, float)
InfP = Length of Infectious Period (TIME RANGE in which the Disease is Infectious, array)
Mob = Mobility (PERCENTAGE PROBABILITY of Contracting the Disease from an Adjacent Diseased Individual, float)
IncP = Length of Incubation Period (TIME RANGE in which the Disease does not produce Symptoms (In an Infective Individual), array)
Mor = Morbidity (PERCENTAGE PROBABILITY of Infective Individuals Dying from the Disease, float)
Imn = Immunity (PERCENTAGE PROBABILITY of Individuals of Gaining Immunity After Recovery, float)
RecP = Length of Recovery Period (TIME RANGE in which the Disease is Present in an Individual, array)
"""

from StochasticEPIMOD1 import Dossier
from StochasticEPIMOD1 import Spread
from StochasticEPIMOD1 import Plot
from StochasticEPIMOD1 import DeepCopy

import random
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize


```

```
def main(job_id, params):
	Parameters = [params["I"], params["InfP"], params["Mob"], params["IncP"], 
params["Mor"], params["Imn"], params["RecP"]]
	return -StochasticEPIMOD(Parameters)

def StochasticEPIMOD(Parameters, xDimension = 20, yDimension = 20, tElapsed = 365, flux = 
True, move_range = 0.5, Plot = False, Print = False):
	#INITIAL VALUES
	Population = xDimension*yDimension

	#PARAMETERS
	PImmune = 0.0
	I = Parameters[0] #Percentage
	InfP = range(int(Parameters[1])-3, int(Parameters[1])+4)
	Mob = Parameters[2] #Percentage
	IncP = range(int(Parameters[3])-3, int(Parameters[3])+4)
	Mor = Parameters[4] #Percentage
	Imn = Parameters[5] #Percentage
	RecP = range(int(Parameters[6])-3, int(Parameters[6])+4)

	#PATIENTS
	Patient = dict()
	ID = 0
	Pos2Pat = np.zeros((yDimension, xDimension))
	for x in range(xDimension):
		for y in range(yDimension):
			ID += 1
			Patient[ID] = Dossier(x, y, PImmune, IncP, InfP, RecP)
			Pos2Pat[y,x] = ID

	#GENERATING PATIENT ZERO
	x = xDimension/2
	y = yDimension/2
	for P in range(1, ID+1):
		if Patient[P].xLoc == x and Patient[P].yLoc == y:
			Patient[P].Status = "INFECTIOUS"

	Record = DeepCopy(Patient)

	############################SIMULATION
BEGINS#####################################
	for t in range(tElapsed):
		for P in range(1,ID+1):
			#STATUS PRIORITY: DEAD = IMMUNE > INFECTIOUS = INCUBATING >
```

```
SUSCEPTIBLE
            if Record[P].Status == "SUSCEPTIBLE":
                n = 8
                x = Patient[P].xLoc
                y = Patient[P].yLoc
                while Patient[P].Status != "INFECTIOUS" and Patient[P].Status != "INCUBATING" and n > 0:
                    if n == 8:
                        infection = Spread(Mob, Record[Pos2Pat[y,(x+1)%xDimension]]) #Right
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 7:
                        infection = Spread(Mob, Record[Pos2Pat[y,(x-1)%xDimension]]) #Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 6:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,x]]) #Above
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 5:
                        infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,x]]) #Below
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 4:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,(x+1)%xDimension]]) #Upper Right
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 3:
                        infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,(x-1)%xDimension]]) #Lower Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 2:
                        infection = Spread(Mob, Record[Pos2Pat[(y+1)%yDimension,(x-1)%xDimension]]) #Upper Left
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    elif n == 1:
                        infection = Spread(Mob, Record[Pos2Pat[(y-1)%yDimension,(x+1)%xDimension]]) #Lower Right
                        if infection == True:
                            infection = random.random()
                            if infection <= I:
                                Patient[P].Status = "INFECTIOUS"
                            else: Patient[P].Status = "INCUBATING"
                    n -= 1

            elif Record[P].Status == "INFECTIOUS":
                Patient[P].Recovery(Imn)
                if Patient[P].RecState == True: #If the patient is still infectious...
                    Patient[P].Infection(Mor)

            elif Record[P].Status == "INCUBATING":
                Patient[P].Recovery(Imn)
                if Patient[P].RecState == True: #If the patient is still incubating...
                    Patient[P].Incubation(I,Imn)

            elif Record[P].Status == "IMMUNE": #Removed from population
                pass
            elif Record[P].Status == "DEAD": #Removed from population
                pass

        #UPDATE MOVING
        if flux == True:
            for x in range(xDimension):
                for y in range(yDimension):
                    if Patient[Pos2Pat[y,x]].Status != "DEAD":
                        move = random.random()
                        if move <= move_range:
                            swap = random.randint(1, 4)
                            if swap == 1: #switch with above
                                oriID = Pos2Pat[y, x]
                                newID = Pos2Pat[(y+1)%yDimension, x]
                                Patient[oriID].yLoc = (y+1)%yDimension
                                Patient[newID].yLoc = y
                                Pos2Pat[y,x] = newID
                                Pos2Pat[(y+1)%yDimension, x] = oriID

                            elif swap == 2: #switch with below
                                oriID = Pos2Pat[y, x]
                                newID = Pos2Pat[(y-1)%yDimension, x]
                                Patient[oriID].yLoc = (y-1)%yDimension
                                Patient[newID].yLoc = y
                                Pos2Pat[y,x] = newID
                                Pos2Pat[(y-1)%yDimension, x] = oriID

                            elif swap == 3: #switch with left
                                oriID = Pos2Pat[y, x]
                                newID = Pos2Pat[y, (x-1)%yDimension]
                                Patient[oriID].xLoc = (x-1)%xDimension
                                Patient[newID].xLoc = x
                                Pos2Pat[y,x] = newID
                                Pos2Pat[y, (x-1)%yDimension] = oriID

                            elif swap == 4: #switch with right
                                oriID = Pos2Pat[y, x]
                                newID = Pos2Pat[y, (x+1)%yDimension]
                                Patient[oriID].xLoc = (x+1)%xDimension
                                Patient[newID].xLoc = x
                                Pos2Pat[y,x] = newID
                                Pos2Pat[y, (x+1)%yDimension] = oriID
        SUSCEPTIBLE = 0
        INFECTIOUS = 0
        INCUBATING = 0
        IMMUNE = 0
        DEAD = 0

        for P in range(1, ID+1):
            if Patient[P].Status == "SUSCEPTIBLE":
                SUSCEPTIBLE += 1
            elif Patient[P].Status == "INFECTIOUS":
                INFECTIOUS += 1
            elif Patient[P].Status == "INCUBATING":
                INCUBATING += 1
            elif Patient[P].Status == "IMMUNE":
                IMMUNE += 1
            elif Patient[P].Status == "DEAD":
                DEAD += 1

        Record = DeepCopy(Patient)

    PercentDead = float(DEAD)/float(Population)
    return PercentDead

def Average(Parameters, tests = 1000):
    total = 0.0
    results = np.zeros(tests)
    for n in range(tests):
        results[n] = float(StochasticEPIMOD(Parameters))
        print n
    print np.mean(results)
    print np.std(results)
    print results
    return np.mean(results), np.std(results), results

data = open("StochasticEPIMOD1SARSresultsDEAD.txt", "w")

Combination = [0.2860000, 9, 0.256050, 5, 0.34989, 0.462340, 17]
average, stdev, results = Average(Combination)

data.write("Average: %f" %average)
data.write("Standard Deviation: %f" %stdev)
for i in range(len(results)-1):
    data.write("%f" %results[i])
```

Infectivity	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0.58794	0.65814	0.72177	0.7782	0.82672	0.86648	0.8965	0.91566	0.92258	0.91525
Infection Rate	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0.92235	0.92258	0.92223	0.9211	0.91895	0.91542	0.91003	0.90209	0.89061	0.87414
Mobility	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0.92314	0.91627	0.90518	0.88673	0.85485	0.79698	0.68413	0.4279	0	0
Incubation Rate	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0.57929	0.65022	0.71465	0.77197	0.82146	0.86229	0.89352	0.91401	0.92245	0.91696
Mortality	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0.9227	0.92123	0.91884	0.91513	0.9095	0.90095	0.88763	0.86564	0.82444	0.72152
Percentage Immunity	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0	0	0	0.00002	0.0466	0.09498	0.15879	0.24622	0.37337	0.57796
Recovery Rate	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Percentage Death	0	0	0	0.00002	0.20618	0.33827	0.46093	0.57964	0.69716	0.81512

PARAMETERS	STOCHASTIC	DETERMINISTIC
Initially Susceptible Population/ Total Population (Pop)	N/A: There was not enough computational power to simulate population as a parameter.	(PHASE I only) Total number of people in each compartment. Remains constant throughout simulation
Infectivity (I)	Percentage Probability of Population that Displays Symptoms When Infected	Percentage of Population that Displays Symptoms When Infected
Length of Infectious Period (InfP)	(+/-3) Possible Time Range in which the Disease is Infectious	Time in which the Disease is Infectious
Mobility (Mob)	Percentage Probability of Contracting the Disease from an Adjacent Diseased Individual	Number of Healthy Individuals that a Diseased Individual may Infect
Length of Incubation Period (IncP)	(+/-3) Possible Time Range in which the Disease does not produce Symptoms (In an Infective Individual)	Time in which the Disease does not produce Symptoms (In an Infective Individual)
Length of Recovery Period (RecP)	(+/-3) Possible Time Range in which the Disease is Present in an Individual	Time in which the Disease is Present in an Individual
Mortality (Mor)	Percentage Probability of Infective Individuals Dying from the Disease	Percentage of Infective Individuals Dying from the Disease
Immunity (Imn)	Percentage Probability of Individuals of Gaining Immunity After Recovery	Percentage of Individuals that Gain Immunity Upon Recovery

SARS Data points for Fitting	
153	78
158	98
159	103
165	144
166	154
186	461
187	478
188	495
189	506
202	666
204	689
210	750
211	754
216	772
221	784
228	799
229	799
231	804
238	809
239	810
242	811
243	812
244	812
249	812
252	812

Ebola Data Points for Fitting	
7	24
279	12713
312	13697
347	14487
382	15151
413	15854
446	16236
479	16470
510	16763