

Optimizing Interventions in the Ebola Epidemic Using a Compartmental Model

Final Report
APC 524, Fall 2014

Developers

Jesse Ault
Alta Fang
Sandra Sowah
Yile Gu

Project Objective

The software package we propose will determine how to optimally allocate a given, finite number of resources during an Ebola virus outbreak in order to minimize the spread of the disease and contain the outbreak. This software package will use a stochastic compartmental model [1] to simulate the spread of the disease. Given resource constraints and epidemic information, it will fit the model parameters to the epidemic data and use that model to forecast the effect of certain interventions on the trajectory of the disease. By modeling the effects of different intervention distributions, the software will determine how a fixed quantity of resources can be allocated in order to have the minimal death count due to Ebola epidemic.

The software allows user to interact through either a friendly command line interface or a GUI interface. User needs to provide 1) a time series of cumulative Ebola infection case count in a csv file, 2) total population of the corresponding region, and 3) total resources and cost functions for each intervention in a csv file. Then, the program can calculate model parameters, optimal allocation resources, and projections of Ebola spread with or without the allocations (the allocations can be either user specified or the optimal ones from calculations). User has the option to generate figures to inspect the goodness of model fit, as well as the time series of the disease spread. Installation is needed for one part of the code; for details about the installations, please refer to the user manual.

Compartmental Model Overview

A compartmental model [1-2] has been proposed and used to study the Ebola outbreaks in the Democratic Republic of Congo (1995), in Uganda (2000), and in Sierra Leone and Liberia (2014). In this compartmental model, individuals are classified as: (1) susceptible individuals (S) who can be infected by the Ebola virus following contact with infectious cases; (2) exposed individuals (E) who have been infected by the Ebola virus but are not yet infectious or symptomatic; (3) symptomatic and infectious individuals in the community (I); (4) hospitalized Ebola cases (H) who are infectious; (5) dead Ebola cases (F) who may transmit the disease during funerals; and (6) individuals removed from the chain of transmission (R, cured or dead and buried). The dynamics for the population of these types of individuals are modeled through coupled ordinary differential equations (ODEs) with stochastic components, as shown in Figure 1 below.

These governing ODEs consider the populations of each compartment to be continuous. However, the populations are rather always discrete as integers. To encounter this problem, the previous studies [1-2] suggested that these simulations could be performed using Gillespie's first reaction method [3]. We implemented the method in C++. Details about the method would be discussed in the section of Design Process.

$$\begin{aligned}
\frac{dS}{dt} &= -\frac{\beta_I SI + \beta_H SH + \beta_F SF}{N} \\
\frac{dE}{dt} &= \frac{\beta_I SI + \beta_H SH + \beta_F SF}{N} - \alpha E \\
\frac{dI}{dt} &= \alpha E - [\gamma_H \theta_1 + \gamma_I (1 - \theta_1)(1 - \delta_1) + \gamma_D (1 - \theta_1) \delta_1] I \\
\frac{dH}{dt} &= \gamma_H \theta_1 I - [\gamma_{DH} \delta_2 + \gamma_{MH} (1 - \delta_2)] H \\
\frac{dF}{dt} &= \gamma_D (1 - \theta_1) \delta_1 I + \gamma_{DH} \delta_2 H - \gamma_F F \\
\frac{dR}{dt} &= \gamma_I (1 - \theta_1)(1 - \delta_1) I + \gamma_{MH} (1 - \delta_2) H + \gamma_F F
\end{aligned}$$

Figure 1: Compartmental model [1] for Ebola spread.

Several parameters in the model as in Figure 1 can be tuned to reflect the levels of prevention or intervention:

Parameter	Intervention	Effect
β_H	Personal protective equipment and other supplies	Contact rate for hospitalized cases
δ_2	Hypothetical drug	Fatality rate of hospitalized patients
θ_1	Contact tracing and community awareness	Fraction of infected cases diagnosed and hospitalized

In this project, it is assumed that β_H , δ_2 increases and θ_1 decreases linearly as the resource allocated to each of them increases. Given a fixed quantity of resources, we determine how to most efficiently allocate those resources amongst these three intervention methods to minimize the number of death from the Ebola disease. This is done by incorporating the epidemic model into an optimization cost function and performing constrained optimization to calculate the most efficient resource allocation that will minimize the expected number of deaths.

In other words, we minimize $f(x_1, x_2, x_3)$ subjected by the constraint that $\sum_{i=1}^3 x_i = 1$ and $x_i > 0$. $f(x_1, x_2, x_3)$ calculates the expected/averaged number of deaths corresponding to a given allocation of resources by running the stochastic simulations with many trajectories. x_i represents the fraction of total resource allocated to each of the three interventions.

Tools Used

The tools used in this project are shown below:

Purpose	Tool or resources
Visualization in Python	Matplotlib
Optimization in Python	Scipy
Documentation	Sphinx
Distribution	Distutils
Testing	Python unittest
Coupling between C++ and Python	Cython
Parallel computation	OpenMP
Version Control	Git
Profiling	Not sure

Design Process

Except for the part where Gillespie's first reaction method is implemented in C++, all the implementation is performed in Python. Figure 2 is the UML (unified modeling language) of the code.

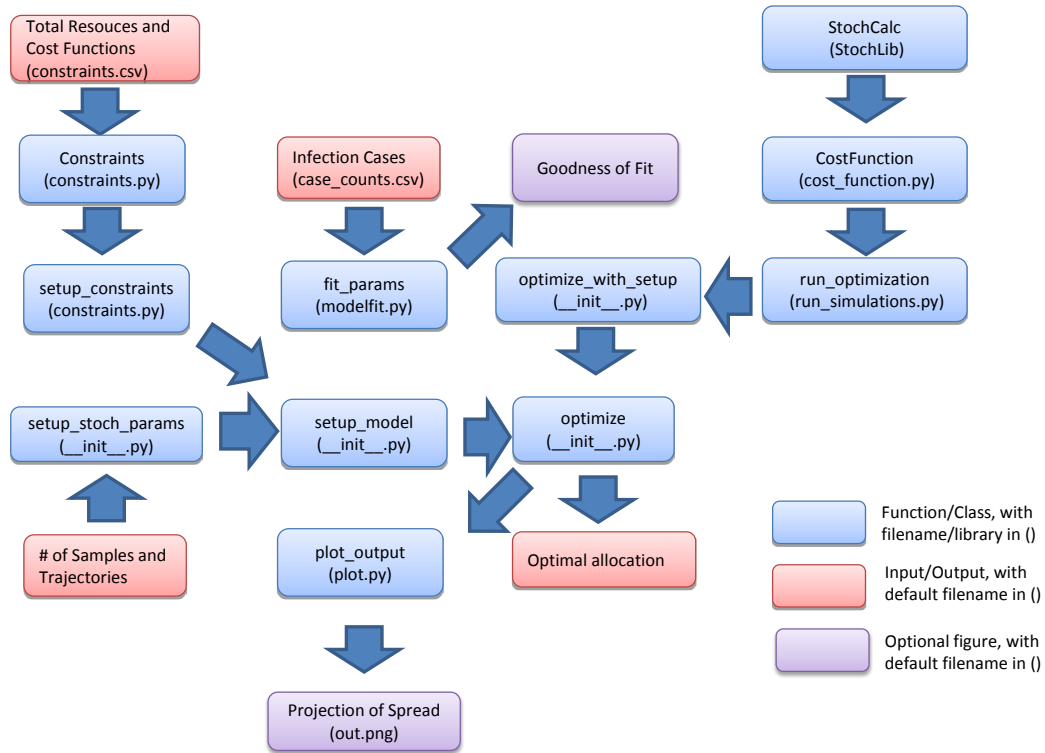


Figure 2: UML of the code

Several major functions in this UML are discussed below.

- fit_params

Functionality: In this function, inputs are the data of infection cases and respective population, and the outputs are the model parameters and optionally a figure comparing the data with the model.

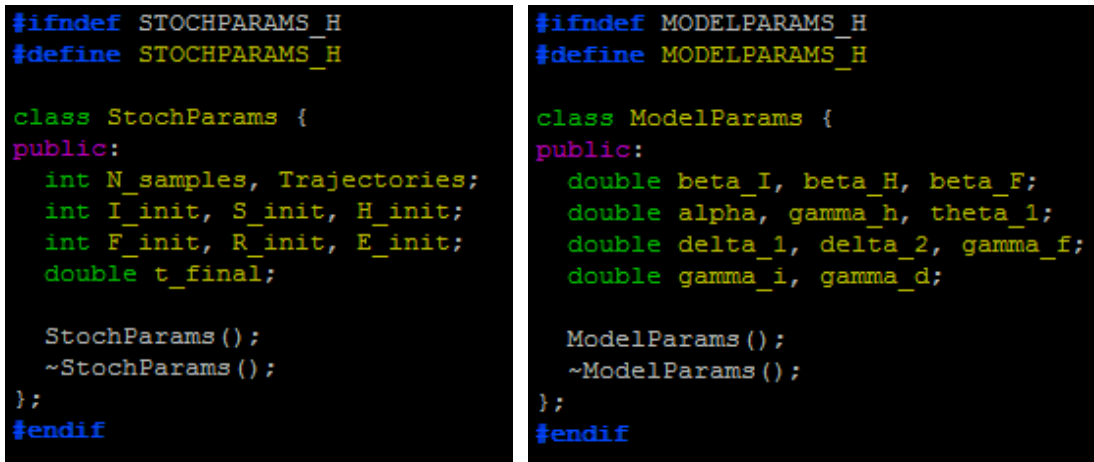
Specifics: A deterministic version of the compartment model is implemented as a function named SIROde. The model parameters are initialized using the values from literature [2]. COBYLA method from Scipy is used to minimize the difference between the model predictions and data, with appropriate constraints on model parameters. The model parameters calculated are then stored in the object OrigParams, which is then used in other functions.

- StochCalc

Functionality: Gillespie’s first reaction method [3] is implemented which calculates many trajectories. For each trajectory, time series of population in each compartment is obtained. According to the Gillespie algorithm [3], for each trajectory, the following steps are followed:

- 1. Initialize the populations and reaction constants.
- 2. For each possible reaction, calculate the probability it will be the next reaction.
- 3. Generate random numbers to determine the next reaction using the probabilities and the time interval.
- 4. Update and repeat until the final time.

Specifics: C++ solver written to utilize only the Gillespie algorithm for the specific rate equations of our compartmental model. OpenMP is included to allow the user to optionally run the code on parallel. Cython is used to call the C++ StochCalc solver from our Python optimization loop. Two classes are used to hold all the parameters for the solver as shown in Figure 3. Python equivalent classes are passed between the Python and C++ codes.



```
#ifndef STOCHPARAMS_H
#define STOCHPARAMS_H

class StochParams {
public:
    int N_samples, Trajectories;
    int I_init, S_init, H_init;
    int F_init, R_init, E_init;
    double t_final;

    StochParams();
    ~StochParams();
};
#endif
```

```
#ifndef MODELPARAMS_H
#define MODELPARAMS_H

class ModelParams {
public:
    double beta_I, beta_H, beta_F;
    double alpha, gamma_h, theta_1;
    double delta_1, delta_2, gamma_f;
    double gamma_i, gamma_d;

    ModelParams();
    ~ModelParams();
};
#endif
```

Figure 3: Two classes that hold all the parameters for the solver in C++

- run_optimization

Functionality: This function uses the model parameters obtained from `fit_params`, and utilizes `StochCalc` to determine the expected number of deaths. Through optimization, it finds the optimal resource allocation.

Specifics: The COBYLA method from Scipy is used for minimization. The function to be minimized is represented by a `CostFunction` callable object that calls `StochCalc` for calculations and is initialized using parameters from `fit_params` as well as user input. To prevent unreasonable results, two things are checked before performing the optimizations. Specifically, the intervention time is checked to make sure that it is greater than the final time of the simulation. Also, the total resource budget is checked to ensure that it is not too much such that all of the interventions can be maximized. In either of these cases, no optimization is necessary and only one simulation is performed.

Testing

Unittest is used to test python scripts. All the test files are included in `tests` folder under `ebolaopt` folder.

`test_modelfit.py` is used to test function `fit_params`.

- test1: It tests that the code gives reasonable results for different countries.
- test2: It tests that the code can optionally gives figures comparing model predictions with history data.

`test_optimization.py` is used to test resource allocations.

- test1: It tests that the optimization would not run if the prevention time is after the final time.
- test2: It tests that the optimization would not run if too much total resource is present.
- test3: It tests that the optimization would work for different countries.
- test4: It tests that the boundary cases where all the resources is allocated to one particular prevention.
- test5: It tests that the parallelization capabilities of the code.
- test6: It tests that the optimal resource allocations obtained from the code here actually give better results than non-optimal resource allocations. More details about the results from this test are included in the section Simulation Results and Discussions.

Profiling

Gillespie's first reaction method is identified as the most computationally intense part of the code. To speed up this part of the code, the method is written in a problem specific C++ code, rather than using `StochPy`, a generic Python tool. As shown in table below as well as in Figure 4, significant speed-up can be achieved by choosing `StochCalc`, the C++ code written in this project, over `StochPy`. For both `StochCalc` and `StochPy`, the required run time increases linearly with the number of trajectories.

# Trajectories	Run Time		
	StochPy	StochCalc	Speedup Factor
1	13.3 s	0.189 s	70.4
5	52.5 s	0.478 s	109.8
10	123.1 s	1.007 s	122.2
50	672.6 s	6.410 s	104.9
100	1313 s	14.13 s	92.9

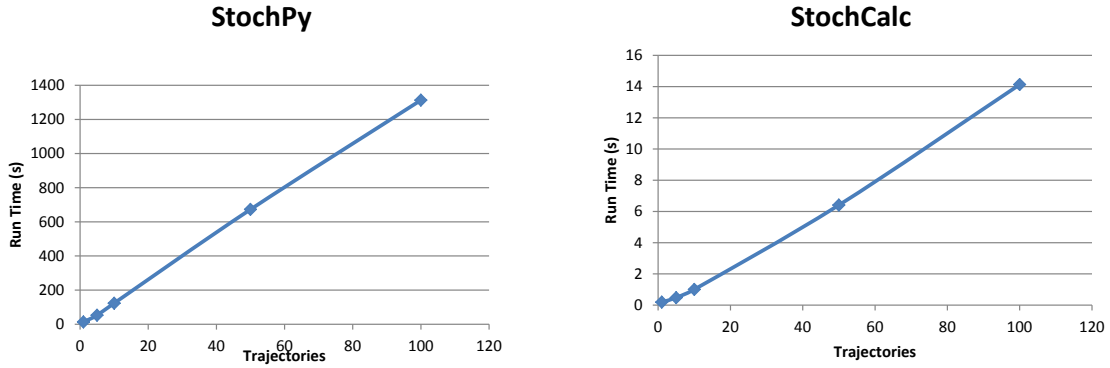


Figure 4: Comparisons of the run time of the Gillespie’s first reaction method between StochPy and StochCalc for various numbers of trajectories.

Simulation Results and Discussions

To demonstrate the usage of the software, we did a comprehensive study on the recent outbreak in the country of Sierra Leone (test6 in *test_optimization.py*). The data are obtained from Caitlin Rivers from Virginia Tech, who procure them from World Health Organization and WHO situation reports.

As demonstrated in Figure 5, the software is able to find the model parameters that fit the data well. In order to correctly calculate the optimal resource allocation to minimize number of deaths, the cost of resources for each of the three intervention parameters β_H , δ_2 and θ_1 needs to be obtained. However, despite much literature research, no concrete information on the subject is found. Here, we assume that:

- For every 60 units of resource spent, β_H is decreased by 0.01.
- For every 500 units of resource spent, δ_2 is decreased by 0.05.
- For every 200 units of resource spend, θ_1 is increased by 0.01.
- We have 10000 units of resource in total for allocation.

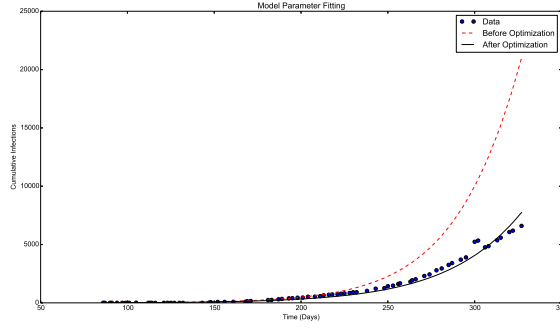


Figure 5: After optimization, the software is able to find model parameters within constraints that improve the model fit with the data for cumulative infections in Sierra Leone.

- The intervention starts at 200 days after the outbreak.

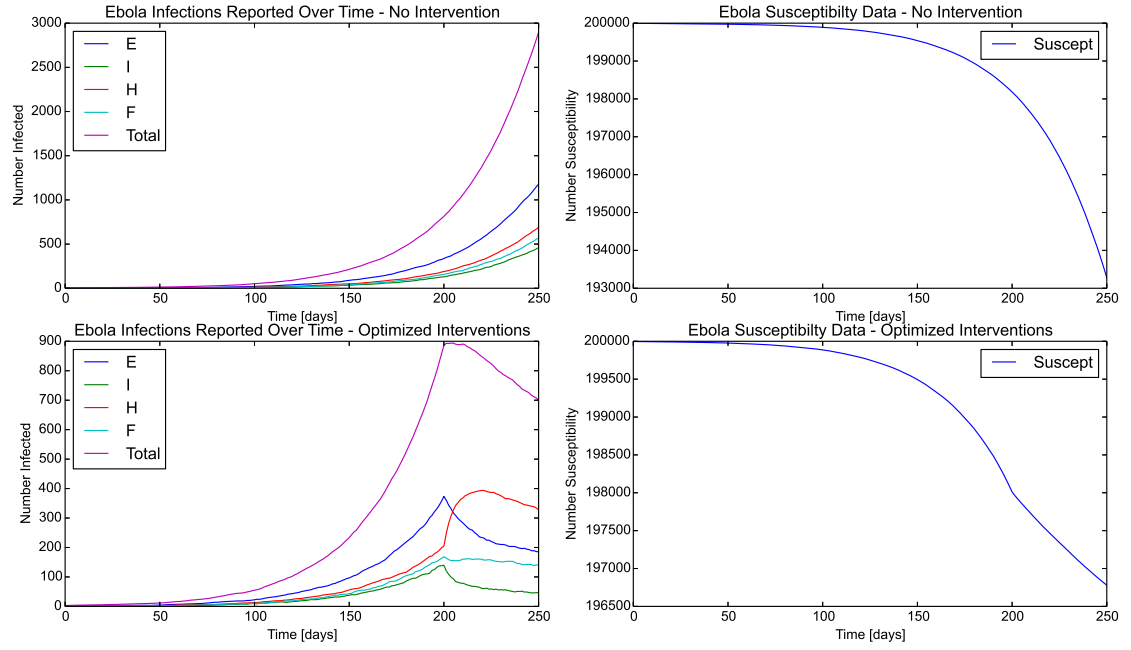
From the assumptions above, the software is able to find the optimal resource allocation. The model predictions of Ebola infections over time without and with interventions are shown in Figure 6(a). As shown in the figure, without the interventions, the infections grow exponentially. People in the Suspect compartment which represents the people who are free from Ebola decrease with increasing speed. With the interventions which start at the day of 200, the infections are checked and even decrease over time. People who are free from Ebola decrease with much lower speed in comparison.

To demonstrate the fact that the optimal resource allocation obtained from this software is indeed a favorable option, we use the same amount of resources but allocate all of them to β_H , and perform the same calculations to find the Ebola spread over time. The results are shown in Figure 6(b). After comparison between Figure 6(b) with Figure 6(a), it is shown that with the optimal resource allocation, there are less infection cases over time. More people would be free from Ebola. Thus, the optimal resource allocation calculated is indeed an effective allocation.

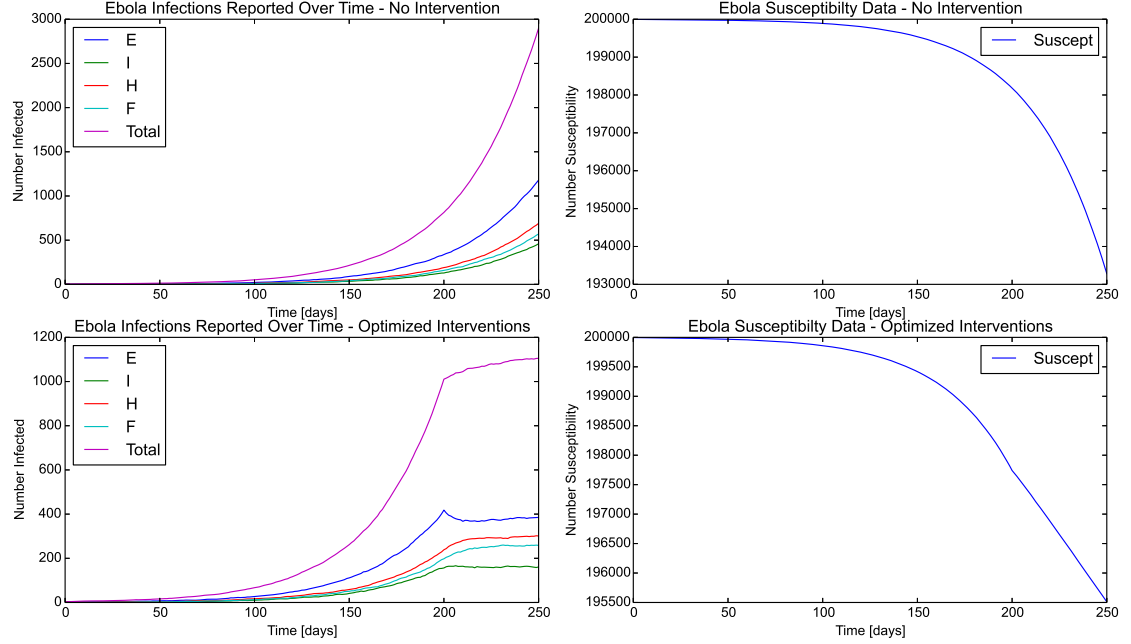
Lessons Learned from the Project

The project gave us an opportunity to experience writing a relatively large code as a team. From the project, we realized the importance of writing a code with good modularity. We also got to practice using github to put the codes together. In addition, we learned to think from the perspective of user when writing the code. This project also gave us a fresh look at how software could help solve important real-world problems.

Distribution of Labor



(a)



(b)

Figure 6: Comparison of Ebola spread over time between the case with no intervention and the one with intervention. In (a), the optimal resource allocation is used. In (b), all the resources is spent on β_H .

Developer	Primary Responsibility
Jesse Ault	Wrote code to solve the stochastic differential equations that model the Ebola outbreak. Interfaced C++ with Python using Cython, and allowed parallel computation through OpenMP.
Alta Fang	Wrapped the optimization around the stochastic solver and handle user inputs. Made the software distributable and installable through distutils.
Sandra Sowah	Analyzed outputs from the stochastic simulations as well as from the optimization, wrote code to visualize them, and made GUI with PyQt. Main contributor to the documentation.
Yile Gu	Wrote code that fits the raw data of an epidemic's history to a deterministic version of the model to generate model parameters which govern the stochastic equations. Wrote Python Unittests for testing. Main contributor to the final report.

References

- [1] Legrand et al. "Understanding the dynamics of Ebola epidemics" Epidemiol. Infect. (2007)
- [2] Rivers et al. "Modeling the impact of interventions on an Epidemic of Ebola in Sierra Leone and Liberia" (2014)
- [3] Gillespie DT. "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions". Journal of Computational Physics (1976)