EbolaOpt Documentation

Release 0.1

Jesse Ault, Alta Fang, Yile Gu, Sandra Sowah

January 15, 2015

This software package determines 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 uses a stochastic compartmental model to simulate the spread of the disease. Given resource constraints and epidemic information, it fits 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 determines how a fixed quantity of resources can be allocated in order to have the greatest positive impact on controlling an Ebola epidemic.

For the most recent version of EbolaOpt, visit our github page at https://github.com/altafang/ebolaproject.

An html version of this documentation (if you are not looking at it already) is at http://www.princeton.edu/~alta/ebolaproject/.

CHAPTER

ONE

CONTENTS

1.1 Background and References

1.1.1 Ebola Epidemic Modeling

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 funeral procedures; and (6) individuals removed from the chain of transmission, R, due to recovering from the disease or being buried. The dynamics for the population of these types of individuals are modeled through the coupled ordinary differential equations (ODEs) given in the following system:

$$\frac{dS}{dt} = -\frac{\beta_I SI + \beta_H SH + \beta_F SF}{N}$$

$$\frac{dE}{dt} = \frac{\beta_I SI + \beta_H SH + \beta_F SF}{N} - \alpha E$$

$$\frac{dI}{dt} = \alpha E - [\gamma_H \theta_1 + \gamma_I (1-\theta_1)(1-\delta_1) + \gamma_D (1-\theta_1)\delta_1] I$$

$$\frac{dH}{dt} = \gamma_H \theta_1 I - [\gamma_{DH}\delta_2 + \gamma_{IH}(1-\delta_2)] H$$

$$\frac{dF}{dt} = \gamma_D (1-\theta_1)\delta_1 I + \gamma_{DH}\delta_2 H - \gamma_F F$$

$$\frac{dR}{dt} = \gamma_I (1-\theta_1)(1-\delta_1) I + \gamma_{IH}(1-\delta_2) H + \gamma_F F$$

These governing ODEs consider the populations of each compartment to be continuous. Solving this type of system is deterministic. A given solution can be found for given inputs. However, we know that the populations are not continuous. Instead, they are always discrete, and they are always integers. Thus, while a deterministic formulation of the problem may result in a solution that represents the most probable outcome, we know that many outcomes are actually possible. Thus, for this type of problem a stochastic formulation of the equations can be more appropriate. With this approach, the interpretation of the coefficients in the equations is changed. In the deterministic formulation, these coefficients are thought of as reaction "rates". However, in the stochastic formulation they are interpreted as

reaction "probabilities per unit time". Using a stochastic formulation, we can solve for not only the most probable outcome of the outbreak, but we can also solve for standard deviations and confidence intervals to understand the worst- and best- case scenarios. In order to solve the stochastic formulation of the basic equations, we utilize Gillespie's first reaction method [3]. In order to solve for a single trajectory for the future compartmental values, the Gillespie algorithm can be briefly summarized as follows:

- Initialize the compartment populations and reaction constants.
- For each possible reaction (infection, death, hospitalization, etc...), calculate the probability that it will be the next reaction.
- Generate random numbers to determine the next reaction using the probabilities and the time interval to next reaction.
- Update and repeat until the final time.

For more details, see [3]. Using these steps, a single trajectory is solved for. By generating many trajectories, mean trajectories and standard deviations can be generated with increasing accuracy as the number of trajectories is increased. Thus, this solution algorithm may be effectively thought of as a dynamic Monte Carlo method.

1.1.2 Resource Allocation Optimization

In order to perform our interventions study, we first determine which parameters may be affected through interventions. We recognize three parameters that could possibly be affected through the use of interventions. First, the transmission rate at hospitals may be reduced by purchasing personal protective equipment and other supplies. Secondly, investing in drugs and treatments may reduce the fatality rate of hospitalized patients. Finally, through contact tracing and community awareness a larger percentage of those infected can be taken to the hospital.

One of the largest uncertainties in our resource allocation comes from assigning cost functions to our resources. For example, how can a given amount of personal protective equipment affect the probability that someone will become infected in the hospital. Such a specific relationship would be difficult to find in practice apart from experimental data. Such data would need to be specific to each country, and furthermore specific to the current socio-economic state of the country. In general, such data is not available, and so we assume specific relationships for our cost function with the hope and understanding that with the right information these software tools could provide even more accurate results. We assume that beta_H and delta_2 increase and theta_1 decreases linearly as the resources allocated to each of them increases. For example, if we give x amount of resources to purchasing personal protective equipment for hospitals, then beta_H will decrease by a given amount y. If instead we gave 2x, then beta_H would decrease by 2y. Thus, given a fixed quantity of resources, we determine how to most efficiently allocate those resources among these three interventions. 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.

1.1.3 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. Gillepsie DT. "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions." Journal of Computational Physics. (1976)
4. Maarleveld. "StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes." PLoS One. (2013)
5. Zaric et al. "Resource allocation for epidemic control over short time horizons." Mathematical Biosciences. (2001)

6. Kasaie et al. "Simulation optimization for allocation of epidemic-control resources." IIE Transactions on Healthcare Systems Engineering (2013)

1.2 Installation

1.2.1 Dependencies

EbolaOpt requires:

- gcc4.8 or higher
- python2.7
- numpy
- scipy
- matplotlib
- PyQt (installs as PyQt4) for the GUI

A unix-based operating system (e.g. Linux or Mac OS X) is assumed.

If you do not already have python2.7, numpy, scipy, matplotlib, and PyQt the Anaconda distribution (https://store.continuum.io/cshop/anaconda/) is a convenient way to install them.

1.2.2 How to Install

Download EbolaOpt-0.1.tar.gz from the releases/ directory of our github repository at https://github.com/altafang/ebolaproject and go into the directory into which you downloaded it. Then execute:

```
tar -xvf EbolaOpt-0.1.tar.gz
cd EbolaOpt-0.1/
python setup.py install
```

If you do not have root permissions, you can do the following instead of "python setup.py install":

```
python setup.py install --home=$HOME
export PYTHONPATH=$PYTHONPATH:$HOME/lib/python
```

Extra tips for installation on Mac OS X

If your built-in version of gcc is out of date, you can update it by installing gcc4.9 using MacPorts:

```
sudo port install gcc49
```

Now change the g++ and gcc symbolic links to point to the new version:

```
sudo mv /usr/bin/gcc /usr/bin/gcc_old
sudo ln -s /opt/local/bin/gcc-mp-4.9 /usr/bin/gcc
sudo mv /usr/bin/g++ /usr/bin/g++_old
sudo ln -s /opt/local/bin/g++-mp-4.9 /usr/bin/g++
```

1.2.3 Testing

Now you should be ready to use EbolaOpt. To test, execute:

```
cd ebolaopt/tests/
python test_optimization.py
```

You can also test that installation worked by going into any directory (except directories that contain a directory called ebolaopt), and at the python command prompt, execute:

```
import ebolaopt
ebolaopt.optimize()
```

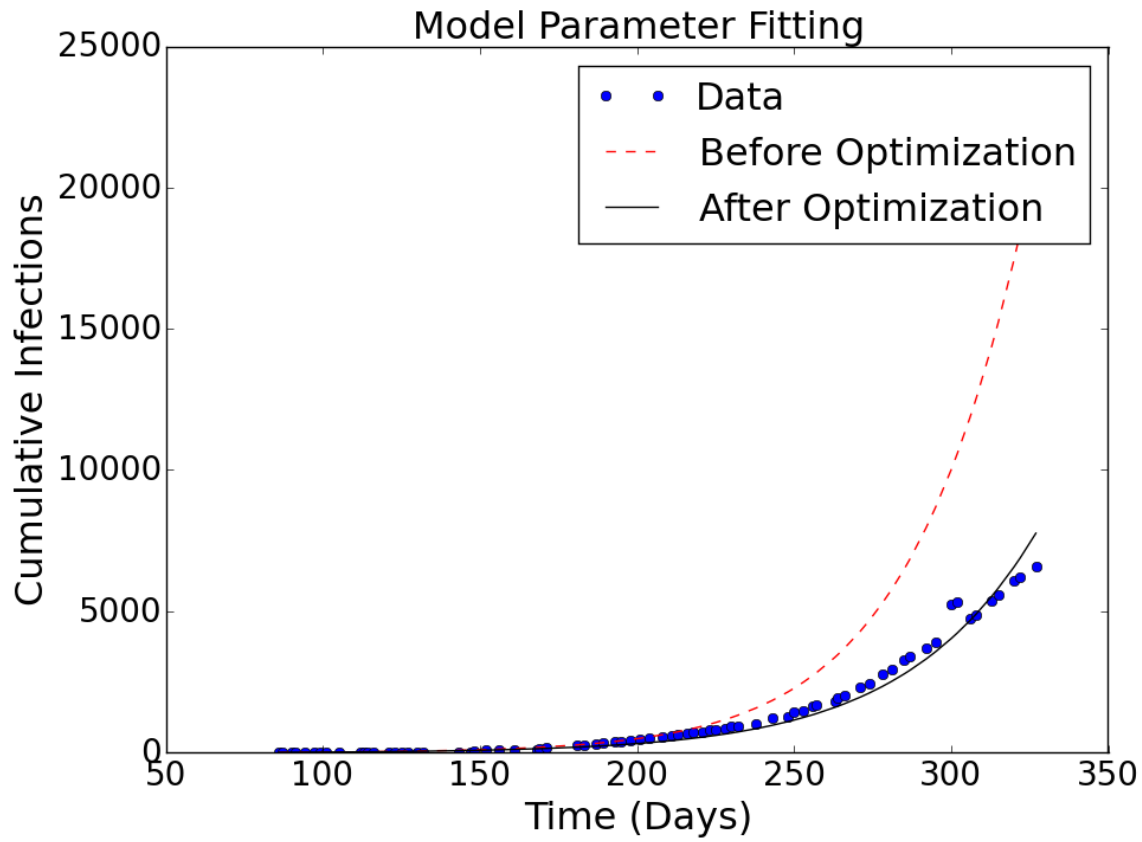
1.3 Getting Started

1.3.1 A Simple Example

Using EbolaOpt can be easy since there are default values for all inputs. Thus, running the following python code:

```
from ebolaopt import optimize
alloc, cost = optimize()
```

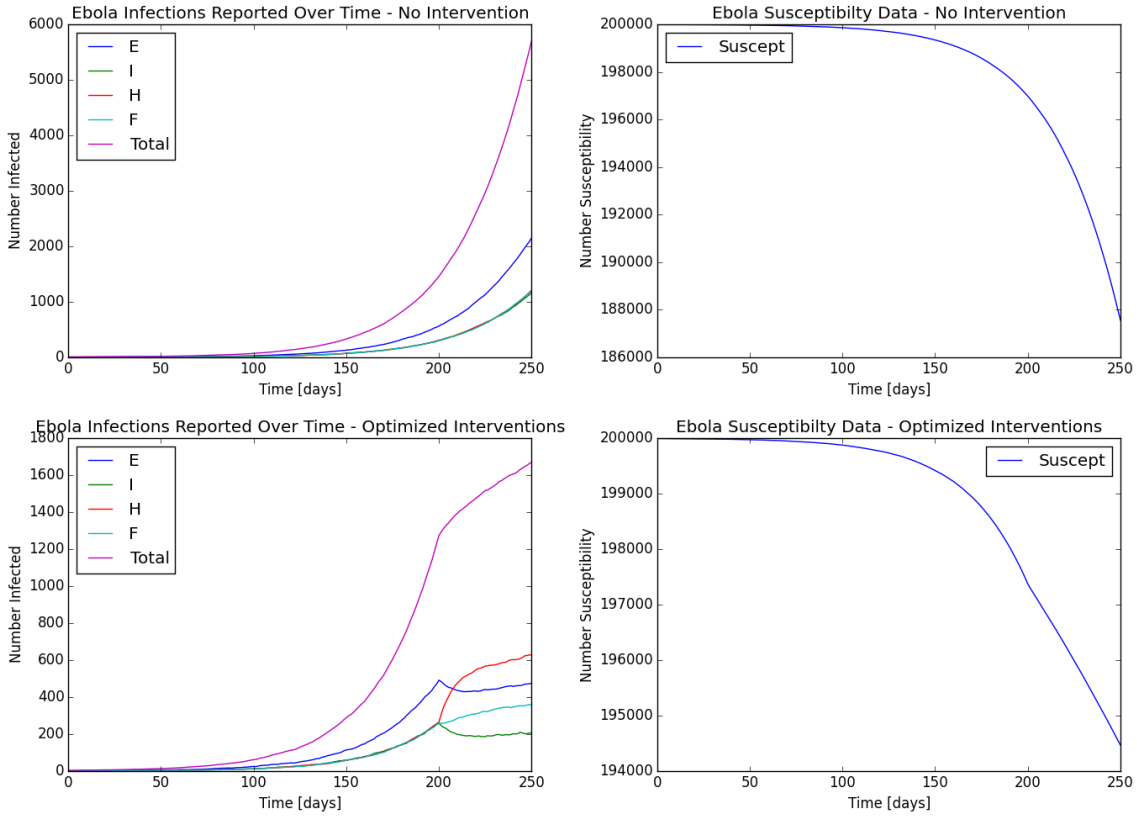
should give a plot of the model fit of the cases data:



After closing this plot, output that looks like the following will be printed out (output below is edited for brevity):

```
    beta_H  delta_2 theta_1 deaths
0   33.33%  33.33%  33.33%  5092.60
1   133.33% 33.33%  33.33%  5092.60
2   33.33%  133.33% 33.33%  2324.20
3   33.33%  133.33% 133.33% 1913.65
4   -57.74% 157.74% -0.00%  55232.25
5   74.13%  14.20%  11.67%  3998.00
...
Optimal resource allocation and cost:
    beta_H  delta_2 theta_1 deaths
    64.36%  21.64%  14.00%  4182.75
```

A plot of the simulation results will also be generated and saved to a file named "out.png":



Furthermore, the data used to generate these final plots are saved to files called "out.csv" and "out_noiv.csv", whose contents look like (truncated for brevity):

```
t (days), S(avg), S(std dev), E(avg), E(std dev), I(avg), I(std dev), H(avg), H(std dev), F(avg), F(s
0.00, 199997.00, 0.00, 0.00, 0.00, 3.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
1.25, 199996.65, 0.67, 0.35, 0.67, 2.50, 0.63, 0.35, 0.51, 0.15, 0.33, 0.00, 0.00
2.50, 199996.05, 1.05, 0.75, 0.80, 2.15, 0.80, 0.65, 0.51, 0.35, 0.46, 0.05, 0.18
...
```

"out_noiv.csv" gives the epidemic trajectory if no interventions are applied, while "out.csv" gives the epidemic trajectory if optimized interventions are applied.

1.3.2 Specifying More Inputs

Suppose you would like to specify the total resource budget as well as the costs and effects of different interventions. This can be done through a csv file. First, create a file called "constraints.csv" containing the following:

```
"total","5000"
"time","150"
"beta_H","60","-0.01"
"delta_2","500","-0.05"
"theta_1","200","0.01"
```

Here, the value after "total" is the total resource budget and the value after "time" is the time at which interventions are introduced. Finally, the values after each intervention name correspond to their costs and effects respectively.

Suppose you also would like to specify the cases data. Create a file called "cases.csv" containing data with the following format (truncated for brevity):

```
"Date","Guinea","Liberia","Nigeria","Sierra Leone","Spain","United States"
"2014-03-22",49,,,,,
"2014-03-24",86,,,,,
"2014-03-25",86,,,,,
"2014-03-26",86,,,,,
"2014-03-27",103,8,,6,,
```

Next, run the following in python:

```
from ebolaopt import optimize
alloc, cost = optimize(constraints_file="constraints.csv", \
                       data_file="cases.csv", t_final=300, \
                       country="Liberia", out_iv_file="liberia_iv.csv", \
                       out_noiv_file="liberia_noiv.csv", \
                       figure_file="liberia.png", \
                       valid_interventions=["beta_H", "theta_1"])
```

This tells EbolaOpt to read the constraints from "constraints.csv", do model fitting of the data in "cases.csv" under the "Liberia" column, run simulations until 300 days, only consider the interventions beta_H and theta_1 when computing the resource allocation optimization, and output simulation results to files named "liberia_iv.csv", "liberia_noiv.csv", and "liberia.png".

1.3.3 Interventions

Currently EbolaOpt supports 3 interventions, listed in the following table:

Parameter	Intervention	Effect
beta_H	PPE & other supplies	Contact rate for hospitalized cases
delta_2	Hypothetical drug	Fatality rate of hospitalized patients
theta_1	Contact tracing & community awareness	Fraction of infected cases diagnosed & hospitalized

You can print this table out from inside of python with the following:

```
from ebolaopt.constraints import constraints_help
constraints_help()
```

1.3.4 Run Simulation Without Optimization

Suppose you just want to run the simulation once for a particular resource allocation. This can be done in the following manner:

```
from ebolaopt import run_simulation
cost = run_simulation([1, 0, 0])
```

In this example, we run the simulation assuming all resources are dedicated to beta_H. Note that the values in the allocation array correspond to the valid intervention names in alphabetical order.

1.4 Graphical User Interface (GUI) Option

The graphical user interface wraps the analysis into a GUI for users do not want to use the command line option. The GUI is compatible for use on both Windows and Mac OS X computers with a slight variation on the look and feel of active main window. The default window options are already included and the only requirement to run the simulation is to import the applicable constraint and data files which have to be specified as .csv files. Since the GUI is created in PyQt, it is important to have PyQt installed in addition to the basic python installation. PyQt installs with the latest version as the default (current default version is PyQt4), and can be installed through the Anaconda distribution.

1.4.1 How to Launch

In python, run:

```
from ebolaopt.Ebola_Simulator import run
run()
```

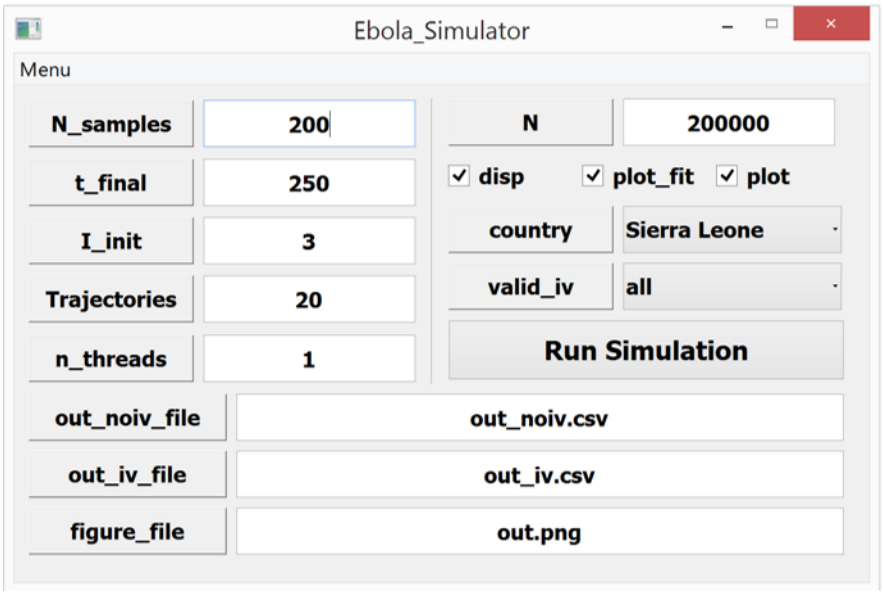
This will launch the GUI, in which you can easily specify your inputs and then run the optimization. Any printed outputs will be printed to where you launched the GUI from.

1.4.2 Schematic of the User Main Window

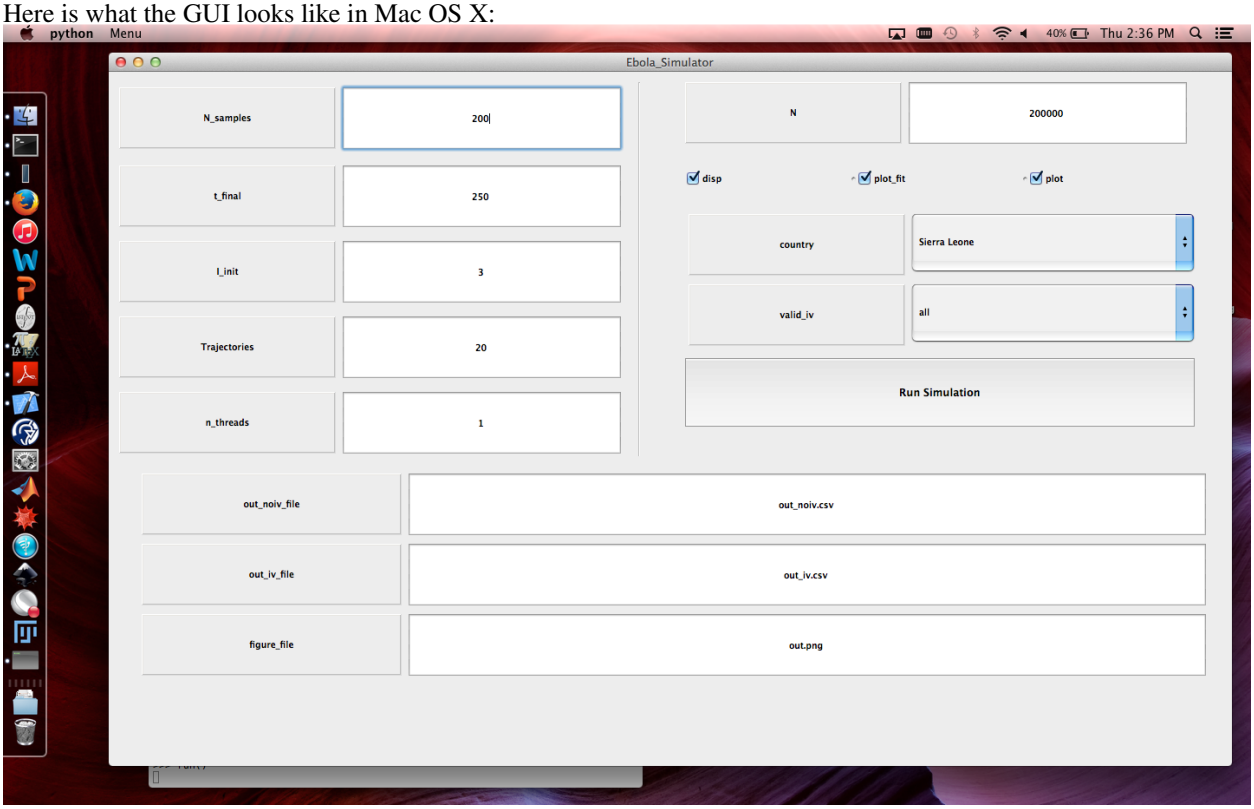
`Source Code: Ebola_Simulator.py`

The Windows and Mac OS versions of the GUI are shown below. Due to restrictions between platforms, the main window is fixed in geometry specification to allow ease of portability from one machine platform to the other. In order to run the simulation successfully, the data file and constraints file must be imported using the menu options defined.

Here is what the GUI looks like in Windows:



Here is what the GUI looks like in Mac OS X:



Note that on Mac OS X, selecting "run simulation" may have the additional side effect of opening a large blank grey box, but just ignore that.

1.4.3 Description of Inputs

- N_samples: number of times to sample the stochastic run to query results for generating the output
- t_final: limit of time series data, in days

- I_init: initial value for the number of infected cases in the community
- trajectories: number of times the stochastic simulation is run for consistency and stability
- n_threads: number of processors to use, OpenMP Parallelization
- N: size of the total population of susceptible persons
- disp: whether to print every step of the optimization
- plot_fit: whether to plot data fitting figure
- plot: whether to generate the final plot in a pop-out window
- country: which country's data to fit
- valid_iv: which interventions are valid to optimize over
- out_noiv_file: output filename, no interventions applied, format=.csv
- out_iv_file: output filename, interventions applied, format=.csv
- figure_file: name of the figure file to save

Note if plot_fit is True, you have to close the plot in order for the optimization to proceed.

1.5 Functions

1.5.1 calc_interventions

`Source Code: tools.py`

Returns a *ModelParams* object. Calculates the values for the interventions to use in the analysis based on the parameters called for computation. Constraint information is taken from *MyConstraints* and *OrigParams* is an instance of *StochLib.ModelParams* that is taken to generate ModifiedParams.

Input Arguments:

alloc = an array containing specified values for the resource allocation to be implemented
OrigParams = *ModelParams* object holding parameters before interventions are applied to the simulation model
MyConstraints = *Constraints* object holding constraints information

Output:

ModifiedParams = new *ModelParams* object holding parameters after interventions are applied to the simulation model

1.5.2 calc_needed_resources

`Source Code: run_simulations.py`

Returns an array listing of needed_resources. The function checks to make sure that the total given is not so large that optimization is pointless.
Arguments:

Input Arguments:

MyConstraints = *Constraints* object holding constraints information

OrigParams = *ModelParams* object holding parameters before interventions are applied to the simulation model

Output:

needed_resources = array of resource allocation

1.5.3 constraints_help

`Source Code: constraints.py`

Prints the intervention options. Describes the meanings of the parameters applied. Takes no arguments.

1.5.4 fit_params

`Source Code: modelfit.py`

Returns an object containing the list of parameters for a specific country.

Input Arguments:

data_file = data file of cases vs. time for various countries

country = specified country based on Ebola data,

N = value, size of the total population of susceptible persons

Keyword Arguments:

plot_fit = False (Default), plots data fitting figure in window

= True, plotting option is ignored

1.5.5 get_data_path

`Source Code: __init__.py`

Returns the *path* directory of the built-in data. It is used to generate the *path* directory of the constraints and cases default files.

Input Arguments:

path = name of the file in the data directory

Output:

path = full path of the data file

1.5.6 parse_data

`Source Code: modelfit.py`

Returns an array listing of the day and cases associated with that particular day for a specific country. The function takes the directory listing of the file path of the raw data csv file and extracts the number of cases versus time for a given country. Requirement: country name should match the string in the .csv file.

Input Arguments:

filename = input file with country string header, must be in .csv format
country = specified country based on Ebola data

Outputs:

days = array listing for the specific day containing data on the number of cases
cases = array listing containing the number of cases reported

1.5.7 plot_output

`Source Code: plot.py`

Generates a figure plot in a window. Data from the simulations run for the optimized option with interventions and the no intervention simulation is plotted for comparison.

Keyword Arguments:

out_noiv_file = output file: no interventions applied, *format=.csv*
out_iv_file = output file: interventions applied, *format=.csv*
figure_file = output figure file, *format* = *.png*

1.5.8 print_heading

`Source Code: tools.py`

Prints the header line saying what the interventions are.

Input Argument:

MyConstraints = *Constraints* object holding constraints parameters

1.5.9 print_output

`Source Code: tools.py`

Prints a formatted output display to the screen for the runs displaying the resource allocation and costs.

Input Arguments:

alloc = an array containing specified values for the resource allocation implemented

cost = value, number of deaths

Keyword Arguments:

linenum = empty string (Default), line number printing is ignored

= number string, prints line numbers for the table output display on the screen

1.5.10 optimize

`Source Code: __init__.py`

Returns optimized *final_cost* with interventions applied to the model. A combined optimized analysis (using **optimize_with_setup** and **setup_model**) is then performed and the generated values are sent to the output files.

Keywords Arguments:

disp = True (Default), prints every step of the optimization

= False

out_noiv_file = output file: no interventions applied, *format=.csv*

out_iv_file = output file: interventions applied, *format=.csv*

figure_file = output figure file, *format = .png*

plot = True (Default), generates the final plot in a pop-out window

= False

n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization

data_file = data file of cases vs. time for various countries

plot_fit = True (Default), plots data fitting figure in window

= False, plotting option is ignored

N_samples = value; number of times to sample the stochastic run to query results for generating the output

trajectories = value; number of times the stoachstic simulation is run for consistency and stability

constraints_file = constraints filename of a csv file with total budget and intervention time and costs

N = value, size of the total population of susceptible persons

valid_interventions = array listing of all interventions applicable, Default = 'all'; other options: eg. ["theta_1", "delta_2"]

I_init = value; initial values for the number of infectious cases in the community

S_init = value; initial values for the number of susceptible individuals

H_init = value; initial values for the number of hospitalized cases

F_init = value; initial values for the number of cases who are dead but not yet buried

R_init = value; initial values for the number of individuals removed from the chain of transmission

E_init = value; initial values for the number of exposed individuals

country = specified country based on Ebola data, Default = "Sierra Leone"

= other options: "Liberia", "Guinea"

t_final = value; limit of time series data calculated in days

Output:

final_cost = value, death metric of associated cost (number of dead people) after optimized simulation

1.5.11 optimize_with_setup

`Source Code: __init__.py`

Returns the *xmin* and *final_cost*. This function computes the *final_cost* values after optimization has been performed based on the parameters given from setup_model.

Input Arguments:

alloc = an array containing specified values for the resource allocation implemented

params = a tuple of selected Ebola parameter objects specific to the *country* option selected

Keyword Arguments:

disp = True (Default), prints every step of the optimization

= False

out_noiv_file = output file: no interventions applied, *format=.csv*

out_iv_file = output file: interventions applied, *format=.csv*

n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization

plot = True (Default), generates the final plot in a pop-out window

= False

figure_file = output figure file, *format* = *.png*

Output:

xmin = value, optimized resource allocation/distribution

final_cost = value, death metric of associated cost (number of dead people) after optimized simulation

1.5.12 run_no_interventions

`Source Code: run_simulations.py`

Returns *cost* when there have been no interventions applied to the model. A stochastic analysis is then performed using the input arguments given and the result generated is the cost associated with a specific intervention applied.

Input Arguments:

OrigParams = object of parameters before interventions are applied to the simulation model

StochParams = object containing parameters for stochastic modelling

out_file = name of simulation output file

Keyword Argument:

n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization

Output:

cost = value, death metric of associated cost (number of dead people) after optimized simulation

1.5.13 run_optimization

`Source Code: run_simulations.py`

Returns the optimized *final_cost* and resource allocation associated with the *final_cost*. This function computes the *final_cost* values after optimization has been performed based on the parameters given from *StochParams*. Error handling is performed for values that correspond to cases where optimization is not needed.

Input Arguments:

OrigParams = object of parameters before interventions are applied to the simulation model
StochParams = object containing a list of parameters for stochastic modelling
MyConstraints = constraints object in a file of praters generated from the *Constraints* object
disp = True or False, whether to generates the plot profile in a pop-out window
out_file = name of simulation output file

Keyword Argument:

n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization

Output:

final_cost = value, death metric for computing associated cost (number of dead people) after optimized simulation

1.5.14 run_simulation

`Source Code: __init__.py`

Returns *final_cost* with/without interventions applied to the model based on an updated params listing. A combined optimized stochastic analysis (using **run_simulation_with_setup** and **setup_model**) is then performed using the input arguments given and specific intervention applied. A figure plot for the trends when interventions have been applied compared to when interventions are not applied is generated. The figure is then saved to an output file.

Input Arguments:

alloc = an array list containing specified values for the resource allocation to be implemented

Keyword Arguments:

out_noiv_file = "out_noiv.csv" (Default). Output file: no interventions applied, *format=.csv*
out_iv_file = "out_iv.csv" (Default). Output file: interventions applied, *format=.csv*
n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization
plot = True (Default), generates the plot profile in a pop-out window
= False
figure_file = output figure file, *format* = *.png*
data_file = data file of cases vs. time for various countries
plot_fit = True (Default), plots data fitting figure in window
= False, plotting option is ignored
N_samples = value; number of times to sample the stochastic run to query results for generating the output
trajectories = value; number of times the stoachstic simulation is run for consistency and stability

constraints_file = constraints filename of a csv file with total budget and intervention time and costs
N = value, size of the total population of susceptible persons
valid_interventions = array listing of all interventions applicable, Default = 'all'; other options: eg. ["theta_1", "delta_2"]
I_init = value; initial values for the number of infectious cases in the community
S_init = value; initial values for the number of susceptible individuals
H_init = value; initial values for the number of hospitalized cases
F_init = value; initial values for the number of cases who are dead but not yet buried
R_init = value; initial values for the number of individuals removed from the chain of transmission
E_init = value; initial values for the number of exposed individuals
country = specified country based on Ebola data, Default = "Sierra Leone"
= other options: "Liberia", "Guinea"
t_final = value; limit of time series data calculated in days

Output:
final_cost = value, death metric of associated cost (number of dead people) after optimized simulation

1.5.15 run_simulation_with_setup

`Source Code: __init__.py`

Returns *final_cost* with/without interventions applied to the model. An optimized stochastic analysis is then performed using the input arguments given and the result is generated. A figure plot for the trends when interventions have been applied compared to when interventions are not applied is generated. The figure is then saved to an output file.

Input Arguments:
alloc = an array list containing specified values for the resource allocation to be implemented
params = tuple of parameters (the output from *setup_model*)

Keyword Arguments:
out_noiv_file = "out_noiv.csv" (Default). Output file: no interventions applied, *format=.csv*
out_iv_file = "out_iv.csv" (Default). Output file: interventions applied, *format=.csv*
n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization
plot = True (Default), generates the plot profile in a pop-out window
= False
figure_file = output figure file, *format* = *.png*

Output:
final_cost = value, death metric for computing associated cost (number of dead people) after optimized simulation

1.5.16 run_with_interventions

`Source Code: run_simulations.py`

Returns *cost* when interventions have been applied to the model. A stochastic analysis is then performed using the input arguments given and the result generated is the cost associated with a specific intervention applied. An array printout of *MyConstraints* and resource allocation with cost values are generated for output into *out_file*.

Input Arguments:

alloc = an array list containing specified values for the resource allocation to be implemented
OrigParams = list of parameters before interventions are applied to the simulation model
StochParams = object containing a list of parameters for stochastic modelling
MyConstraints = constraints object in a file of praters generated from the *Constraints* object
out_file = name of simulation output file

Keyword Argument:

n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization

Output:

cost = value, death metric of associated cost (number of dead people) after optimized simulation

1.5.17 setup_constraints

`Source Code: constraints.py`

Returns an *MyConstraints* object to be used for subsequent analysis. Filters constraints with the *valid_interventions* keyword so that optimization is only run over the specified valid interventions.

Input Arguments:

filename = input file with parameters to be parsed, *format: .csv*
valid_interventions = array listing of all interventions applicable, Default = 'all'; other options: e.g. ["theta_1", "delta_2"]

Output:

MyConstraints = *Constraints* object holding the relevant parameters

1.5.18 setup_model

`Source Code: __init__.py`

Returns *params*, a tuple of selected parameters specific to the country option selected. The Ebola model chosen is then used for the deterministic and stochastic simulation.

Keyword Arguments:

data_file = data file of cases vs. time for various countries
plot_fit = True (Default), plots data fitting figure in window
= False, plotting option is ignored
N_samples = value; number of times to sample the stochastic run to query results for generating the output
trajectories = value; number of times the stoachstic simulation is run for consistency and stability
constraints_file = constraints filename of a csv file with total budget and intervention time and costs
N = value, size of the total population of susceptible persons
valid_interventions = array listing of all interventions applicable, Default = 'all'; other options: eg. ["theta_1", "delta_2"]
I_init = value; initial values for the number of infectious cases in the community
S_init = value; initial values for the number of susceptible individuals
H_init = value; initial values for the number of hospitalized cases
F_init = value; initial values for the number of cases who are dead but not yet buried
R_init = value; initial values for the number of individuals removed from the chain of transmission
E_init = value; initial values for the number of exposed individuals
country = specified country based on Ebola data, Default = "Sierra Leone"
= other options: "Liberia", "Guinea"
t_final = value; limit of time series data calculated in days

Output:
params = a tuple of selected Ebola parameter objects specific to the *country* option selected

1.5.19 setup_stoch_params

`Source Code: run_simulations.py`

Returns an object *StochParams*. This function initializes the parameters for optimization run from the Stochpy library of parameters generated from the stochastic computation previously done. All paramaters defined here are consistent with the *Legrand, J. et al (2006)* paper.

Input Arguments:
N_samples = value; number of times to sample the stochastic run to query results for generating the output
Trajectories = value; number of times the stoachstic simulation is run for a consistency and stability
I_init = value; initial values for the number of infectious cases in the community
S_init = value; initial values for the number of susceptible individuals
H_init = value; initial values for the number of hospitalized cases
F_init = value; initial values for the number of cases who are dead but not yet buried
R_init = value; initial values for the number of individuals removed from the chain of transmission
E_init = value; initial values for the number of exposed individuals
t_final = value; limit of time series data calculated in days

Output:
StochParams = object containing parameters for stochastic modelling

1.6 Class Objects

The class objects used for the Ebola Disease modelling were created to store data parameters. These objects support both attribute references and instantiation (valid attribute names, data attributes and methods).

1.6.1 Constraints

`Source Code: constraints.py`

An object to hold optimization parameters: total resources, resource costs and effects, and time to start interventions.

Inputs:

filename = name of constraints input file with parameters to be parsed, *format: .csv*

Attributes:

total = value, total budget based on the number of resources to allocate
t_interventions = value, time before/after *t_final* for simulations with/without interventions applied respectively
all_interventions = dictionary of interventions listing the associated cost and effects

1.6.2 CostFunction

`Source Code: cost_function.py`

A callable object that must be minimized as part of the optimization computation based on the interventions, associated costs, and resource allocation. Displays a print out of resource allocation and cost in real-time computation if disp=True.

Attributes:

OrigParams = parameters before interventions are applied to the simulation model
StochParams = object containing parameters for stochastic modelling
MyConstraints = *Constraints* object holding intervention information
disp = False (Default)
= True, prints the allocation and cost at each step of the optimization
n_threads = 1 (Default), Number of processors to use, OpenMP Parallelization
n = number assigned to each step during the printout generation

Inputs:

alloc = an array list containing specified values for the resource allocation to be implemented
out_file = "NONE" (Default), other option generates and output text file

Output:

cost = value, cumulative deaths resulting from the given resource allocation

1.6.3 ModelParams

`Source Code: ModelParams.h`

An object containing a list of parameters from the Ebola modelling data in *Legrand, J. et al (2006)* based on the country chosen for running the simulation. The parameters defined are consistent with the parameters defined in *Legrand, J. et al (2006)*.

Attributes:

double *beta_I* = value; transmission coefficient in the community
double *beta_H* = value; transmission coefficient at the hospital
double *beta_F* = value; transmission coefficient during funerals
double *alpha* = value; the inverse of the mean duration of the incubation period
double *gamma_h* = value; the mean duration from symptom onset to hospitalization
double *theta_1* = value; the percentage of infectious cases are hospitalized
double *delta_1* = value; ratio 1 computed in order that the overall case-fatality ratio is *delta*
double *delta_2* = value; ratio 2 computed in order that the overall case-fatality ratio is *delta*
double *gamma_f* = value; the mean duration from death to burial
double *gamma_i* = value; the mean duration of the infectious period for survivors
double *gamma_d* = value; the mean duration from hospitalization to death

1.6.4 StochParams

`Source Code: StochParams.h`

An object containing a list of parameters for stochastic modelling. The parameters defined are consistent with the parameters defined in *Legrand, J. et al (2006)*.

Attributes:

int *N_samples* = value; number of times to sample the stochastic run to query results for generating the output
int *Trajectories* = value; number of times the stoachstic simulation is run for a consistency and stability
int *I_init* = value; initial values for the number of infectious cases in the community
int *S_init* = value; initial values for the number of susceptible individuals
int *H_init* = value; initial values for the number of hospitalized cases
int *F_init* = value; initial values for the number of cases who are dead but not yet buried
int *R_init* = value; initial values for the number of individuals removed from the chain of transmission
int *E_init* = value; initial values for the number of exposed individuals
double *t_final* = value; limit of time series data calculated in days

1.7 Warnings and Pitfalls

1.7.1 Getting Different Results Each Time

On some computers, running the same optimization with the same inputs multiple times can lead to significantly different outputs. The reason for this is that calculations may be performed with very small precision errors that ultimately lead the model fitting to different results since the optimization necessary for model fitting is a local rather than global minimization. For more information, see: http://blog.nag.com/2011/02/wandering-precision.html

1.7.2 Interventions Worsen The Epidemic

In some situations, the simulations will demonstrate that applying interventions may actually worsen the epidemic and result in greater numbers of cases and deaths. This may arise when for example the death rate within the hospital is greater than outside, so that increased hospitalization leads to more deaths.