Mobile Health Surveillance: The Development of Software Tools for Monitoring the Spread of Disease

Albert Gerovitch[1] and Andrew Gritsevskiy[2] and Gregory Barboy[3]

Abstract— **Disease spread monitoring data often comes with a significant delay and low geospatial resolution. We aim to develop a software tool for data collection, which enables daily monitoring and prediction of the spread of disease in a small community. We have developed a crowdsourcing application that collects users' health statuses and locations. It allows users to update their daily status online, and, in return, provides a visual map of geospatial distribution of sick people in a community, outlining locations with increased disease incidence. Currently, due to the lack of a large user base, we substitute this information with simulated data, and demonstrate our program's capabilities on a hypothetical outbreak. In addition, we use analytical methods for predicting town-level disease spread in the future. We model the disease spread via interpersonal probabilistic interactions on an undirected social graph. The network structure is based on scale-free networks integrated with Census data. The epidemic is modeled using the Susceptible-Infected-Recovered (SIR) model and a set of parameters, including transmission rate and vaccination patterns. The developed application will provide better methods for early detection of epidemics, identify places with high concentrations of infected people, and predict localized disease spread.**

INTRODUCTION

The Center for Disease Control provides centralized repositories for data about disease spread in the US. Though well verified and with much detail, this data has a number of deficiencies. First, the data is not localized. Analysts and researchers can see how a disease progresses on a nation-wide or city-wide level, yet they cannot see how a disease travels across a more specific area such as a town or a small village. Besides that, the data that the Center for Disease Control releases reports of the outbreak with a delay of more than two weeks. When working towards stopping the spread of a disease in order to avoid an epidemic, these two factors are crucial, since localized data can help scientists pinpoint the source of a disease, and speed of data acquisition can go towards stopping the disease earlier. In addition, providing a more detailed forecast will allow individuals monitoring disease spread in their neighborhood to take corresponding preventative measures (such as avoidance of crowded spaces) in a more timely manner. The software suite we created

[1] A. Gerovitch is currently a student at Natick High School in Natick, Massachusetts `aliksg9 at gmail.com`

[2] A. Gritsevskiy is currently a student at Lexington High School in Lexington, Massachusetts `agritsevskiy at gmail.com`

[3] G. Barboy is currently a student at Needham High School in Needham, Massachusetts `grr2bar at gmail.com`

works towards providing highly localized data, minimizing the delay, and having the data available for daily use and analysis. We have developed an interactive web application, called Strii, that collects actual data from users on their disease status—if the person is sick or healthy at the time of login. Currently, due to a lack of a sufficient user base in order to obtain real data, we generate simulated data. The goal of Strii is to use this obtained, or generated, individual data to create a local-level disease spread map, to predict the future disease spread in a named city, and to forecast its spread to nearby cities, all using our software suite. A user can enter a town of interest, prompting our program to obtain that town's census data, analyse it together with collected health information, and return the currently observed, past, and predicted future spread of disease in that town back to the user. The results can be visualized to provide a better presentation of findings. In addition, our software suite provides analysis capability, allowing to estimate disease spread or containment under different interventions, different interpersonal interactions, and various severities of the disease. In our forecasting algorithm, we generate human networks based on three models: random, small-world, and scale-free. In these networks, nodes represent people, and edges represent friendships. To further improve the accuracy of our model, we use Census data to build a more realistic representation of a human network and its interactions. We use the SIR (Susceptible-Infected-Recovered) disease spread model to predict and visualize the spread of disease given a number of parameters, including transmission rate and vaccination patterns. Our software suite can take in a variety of parameters, which it then uses to simulate disease to predict its spread. These parameters include many intrinsic factors, such as the number of initially infected people, the duration of the recovery, and even the number of teenagers, who can be curfewed in order to prevent them from interacting with infected nodes. Due to the diversity of our parameters aimed at reflecting actual scenarios, our model, and therefore simulation, becomes more realistic. The paper is organized as follows: Chapter 1 describes the forecasting algorithm that we implemented in order to predict disease spread on a town level. Chapter 2 discusses parameters that we introduce in simulations, and the interactive tool that we developed allowing for analysis of parameters' influence on disease spread simulation results. Chapter 3 discusses an approach for creating crowdsourcing web application for health status data collection, and the currently developed simulated data stream that we use in our analysis. Finally, in Chapters 4 and 5 we present the results of our simulations, a set of visualizations produced by our software tool, and discuss further steps.

I. Materials & Methods: Disease Spread Forecasting

A. Network Modelling

The forecasting algorithm involves simulating how the disease spreads through a sample town. We modeled the spread of disease through probabilistic contacts on the social network using the widely accepted SIR model. In order to achieve sensible results, it was therefore important to create a network of people that is realistic. The network represented of a town where every citizen was a node on an undirected multigraph, and every friendship was an edge between two nodes. Although adding people to the graph did not pose a challenge, the algorithm for interconnecting them was not trivial. In deciding how to connect the nodes of the multigraph, three networks were examined: The Random, Small World, and Scale-Free Networks. Finally, we augmented the developed structure with information from Census data about each individual town in the United States.

1) Random Network: The Random network is assigned a minimum and maximum amount of friends any specific node can have. After that, for each node, a random number of connections between the maximum and minimum is created. Each node can connect to any other node in the graph (Barabási and Bonabeau).

2) Small World Network: The Small World Network is similar to the Random Network; however, in the Small World network, each nodes is numbered, and every node n can only befriend nodes in the range n+k and n-k.[1] The result is a network with localized connections. Every person is friends with only the people in close proximity. The point of this network was to simulate a large town where people are less familiar with people who live farther away, i.e., on the other side of the town (Göpfert and Robert).

3) Scale-Free Network: The Scale-free network is a network constructed using the preferential attachment algorithm (BarabÃąsi and Bonabeau). N nodes are added to the network one by one. However, every new node is more likely to befriend older nodes than newer ones. As a result, the older nodes become "hubs" with large amounts of connections, and newer nodes obtain only a few connections. This results in the degree distribution of this network following a power-law distribution. This network proved to be the most realistic out of the three, since in most communities, some people are very well connected and know a lot of people, while others have less connections, as opposed to the general connectional equality of the other networks (Göpfert and Robert).

4) Scale-Free Network built using Census Data: In order to make the networks more realistic, the concept of households and census data was introduced. Census data of a specific user-entered town was taken from American Fact Finder, and N nodes were added to the graph where N stands for the population of that town. After this, using data on households, the people on the graph are divided into families and are all interconnected by edges, since people who live in one family interact very often and are thus equivalent to "friends" in terms of probabilistic disease spread. Age data from the Census is also taken into account. After dividing people into households, ages are distributed among the family members. This is an often overlooked, yet vitally important factor, since people of different ages can be either very susceptible, or very resilient to disease ("A Weekly Influenza Surveillance Report Prepared by the

[1] These calculations are done by using the modulus of the total amount of people in the network, so the first and last nodes in the graph have edges connecting them to nodes in both ranges *n+k* and *n-k*.

Influenza Division"). Finally, when all of the census data is taken into account, all of the nodes are interconnected using the scale-free algorithm, in order to simulate friendships.

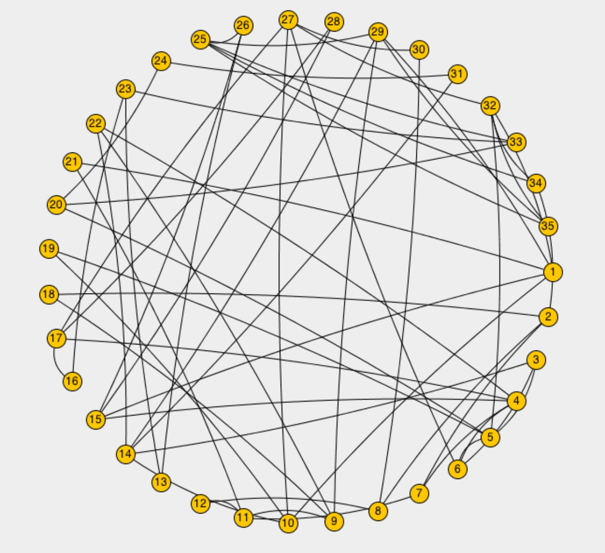


Fig. 1.1. A random network

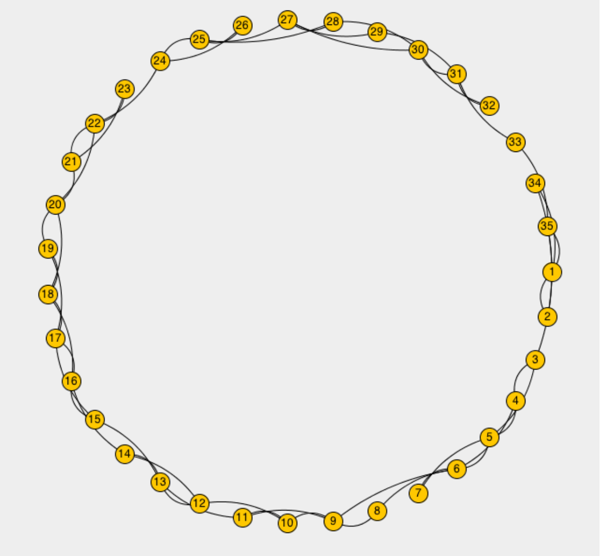


Fig. 1.2. A small world network. Note the "local connections" that characterize the network.

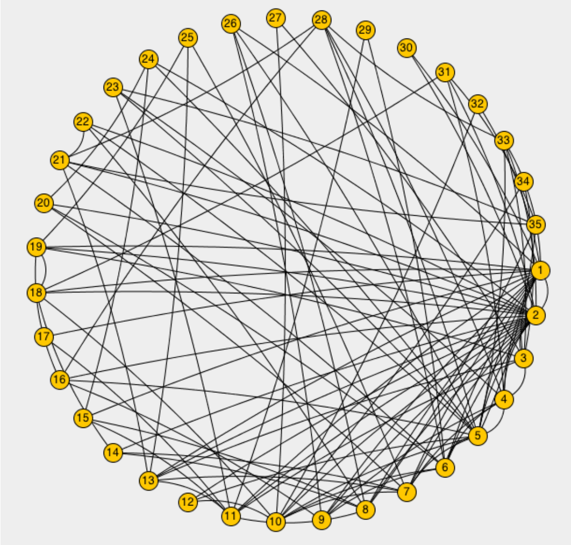


Fig. 1.3. A scale-free network. Note the scale-free characteristic "hubs" visible on the right side of the network.

II. Materials & Methods: Simulation Approach and Parameter Estimation

In our simulation, disease spread is modeled using the SIR algorithm. At every time t, each node is in of three states: Susceptible, Infected, or Recovered. A Susceptible state means that the node can be infected from a connection, representing a person that has not been sick yet; an Infected state means that the node can transmit the infection to a connected node, representing an infected person that is contagious; and a Recovered state represents a node that does not receive or transmit infections, representing a person that has developed an immunity to the disease. For the purposes of utilizing this approach, our program's input consists of several parameters, such as the probability of being infected by a friend, the number of days needed to recover from an infected state (a), or the probability (%) of transmitting the disease through a connection (n).

```
For each infected person:
    1. For each friend, infect that friend
       with a probability of n.
    2. Add a day to the counter.
    3. If the counter is equal to a, set
       this person's state to recovered.
```

This loop over each person represents a "day" of our simulation. One simulation ends when when number of infected people on a given day is equal to 0. Since the interactions between nodes are probabilistic, we take the average of many simulations when testing the effect of a certain parameter. **Fig 2-1** shows the difference between one simulation and 100 simulations. For actual research purposes, the number of simulations may reach the millions.

A. Parameters

A crucial part of our research was studying the effects of parameters on the spread of the epidemic, in order to see how various demographic patterns, types of interactions, and vaccination strategies affect the outcome. In our software suite, the user has the ability to alter any of 14 unique parameters, which will be discussed below.

1) Network Type: Defines type of the graph, which can be either small-world, random, or scale-free. In the current simulation we use the scale-free network model, as we found it to be the most realistic.

2) Number of people: Defines the number of nodes in the graph. We use Census data downloaded from American FactFinder as a proxy for network structure.

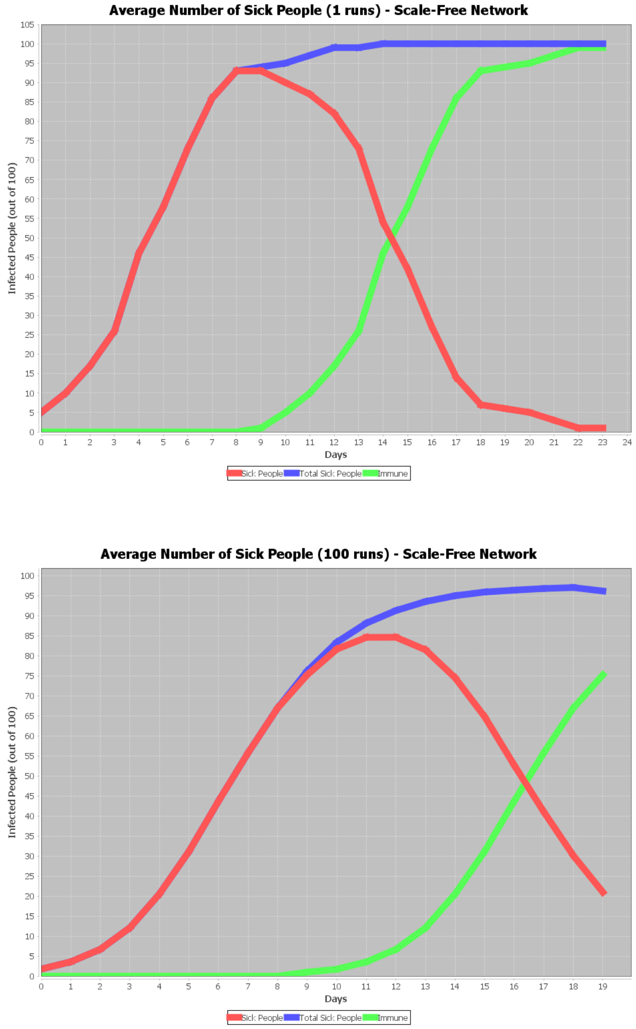


Fig. 2.1. Two simulation results with identical parameters. The one on the left is a single simulation, whereas the one on the right is the average of 100 simulations. If we were to look at just the single simulation, we would have a skewed perception of the disease, with the peak at around day 8, whereas the average peak tends to be around day 11-12.

3) Friend Range: For the random and small-world networks, the user gives the minimum and maximum number of friends that a person can have. When constructing the network, the numbers of friends each person has are randomly distributed within this range.

4) Probability of Random Connection (Small-World): For a small-world network, this is the probability of a connection forming that does not follow the rule of befriending only nearby nodes.

5) Town Simulation: If this is turned on, the user enters the name of a town or city in the USA. Then, the network is constructed by using household and age data, as described in the *Network Modelling* section.

6) Recovery Days: This is the number of days needed for a person to recover from the disease. Currently the default value is set for a common flu average length of recovery identified from literature, which is 3 days. In the current simulations we estimate this parameter through data. We fit the performance of the network with different values for recovery, and find the value providing the best fit.

7) Probability of Transmission: This is the probability that the disease will be transmitted through a single interaction. We estimate this parameter by finding the best fit to actual data. The input is the network of people and the data from the first few days of the disease, and the output is the probability of disease transmission that best matches the results. We run simulations for a number of values for transmission probability looking for the best fit **(Fig 2-2)**. The transmission value providing the smallest integrated variation from the actual results is selected to be used in the simulations.

8) Initial Number of Infected People: Defines number of people at the start of simulations that are in an infected state. In the web application, this information comes from the last day of the Strii data. Presently, Strii data are simulated.

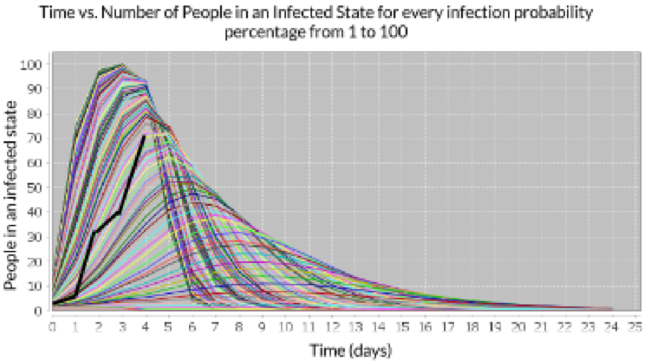


Fig. 2.2. Two simulation results with identical parameters. The one on the left is a single simulation, whereas the one on the right is the average of 100 simulations. If we were to look at just the single simulation, we would have a skewed perception of the disease, with the peak at around day 8, whereas the average peak tends to be around day 11-12.

9) Initial Number of Vaccinated People: Defined number of people at the start of simulation that are in a *recovered* state. Our program also can introduce a given number of people who are immune to the disease into the simulation. These people are determined from the last day of Strii data, or, in our case, simulated Strii data.

10) Probability of Vaccinating from Connection: The probability that a person, given at least one of its connections is infected, will get a vaccine.

11) Percentage of the population that are "teenagers": We assumed that the number of interactions of young people is significantly above average, e. g. they go to high school. We therefore decided to allow selective preventive measures just for these people, e.g. school closure, or quarantining (A parent telling their child to miss a day of school due to sickness). We called nodes with high number of connections "teenagers". This parameter reflects what percentage of the population are "teenagers" and therefore are subject for isolation preventive measures . For example, if the user entered 10, the 10% of nodes with the highest amount of connections would gain "teenager" status.

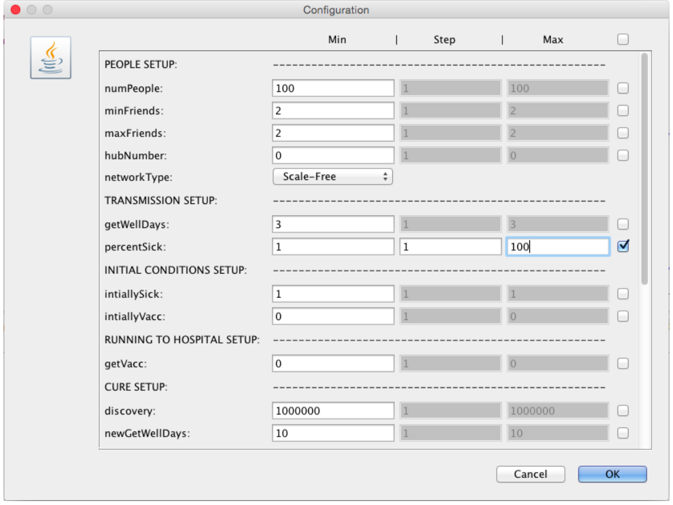
12) Daily probability of a "teenager" being "curfewed": Each day, with a certain probability, each "teenager" can get "curfewed" for a certain number of days determined by parameter 13. If a "teenager" node is under a "curfew", it does not interact with its friends, meaning that it cannot get infected or, if it is sick, infect its friends.

13) Duration of the curfew: The number of days that the curfew lasts for.

14) Number of times the simulation runs: As the nature of simulations is probabilistic, we can offer multiple runs to better approximate the average and deviations of each individual simulation. The results are averaged before being returned to the user.

B. Parameter Analysis

As part of our software suite, we let the user analyse the effect of a single parameter on the entire epidemic (**Fig 2-3**). For example, Fig **2-4** shows the results of analysis of the effect of the probability of infection on the duration of the disease. As discussed later, we can see a correlation that a smaller probability of infection relates to a smaller amount of total incidences of the disease.



	Min	Step	Max	□
PEOPLE SETUP:	------------------------	------------------------	------------------------	
numPeople:	100	1	100	□
minFriends:	2	1	2	□
maxFriends:	2	1	2	□
hubNumber:	0	1	0	□
networkType:	Scale-Free			
TRANSMISSION SETUP:	------------------------	------------------------	------------------------	
getWellDays:	3	1	3	□
percentSick:	1	1	100	☑
INITIAL CONDITIONS SETUP:	------------------------	------------------------	------------------------	
intiallySick:	1	1	1	□
intiallyVacc:	0	1	0	□
RUNNING TO HOSPITAL SETUP:	------------------------	------------------------	------------------------	
getVacc:	0	1	0	□
CURE SETUP:	------------------------	------------------------	------------------------	
discovery:	1000000	1	1000000	□
newGetWellDays:	10	1	10	□

Fig. 2.3. A part of our software suite, which allows the user to analyse the effect of a single parameter on the outcome of the disease. The "Min" column and "Max" columns are the ranges for testing, and the "step" is the testing subdivision. For example, with the displayed setup, the program would run for every probability of infection transmission from 1% (*min*) to 100% (*max*), analysing each 1% (*step*).

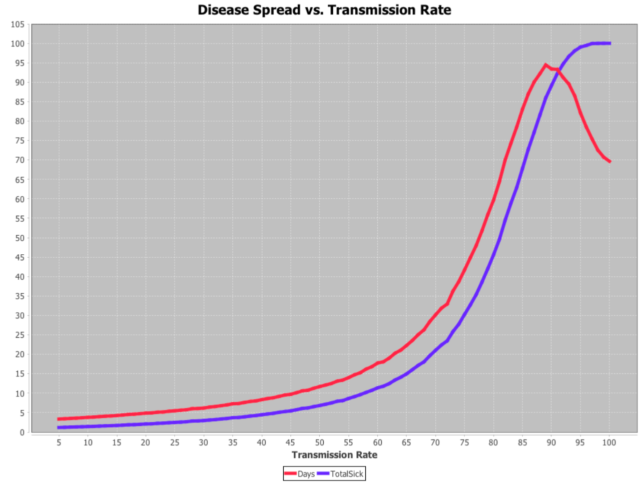
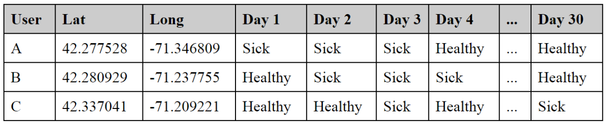


Fig. 2.4. A graph of how the probability of getting infected affects the total number of sick people and length of the overall epidemic. Here, disease spread is shown as a function of transmission rate. It is evidently positively correlated with the total number of sick people. For the duration of the disease, the situation is more complex. For very small percentages, the duration is obviously low. For very large percentages, the duration is also very low, as most people instantly get sick, and all are immune within days. This will be discussed in more detail in our *Results* section.

III. Materials & Methods: Strii Crowdsourcing Application Collecting Health Status Data

A. Data Collection Software

Our online data collection software aims to collect updated localized health status data from individuals accessing our tool. When creating their account, the user's exact location is determined using the HTML5 Geolocation API (Popescu). Then, on a daily basis, a user can log in and enter their health status. This creates a database with users and their recorded data. As shown in **Fig 3-1**, each row in the database corresponds to a user, and contains their geospatial information, as well as 30 days of health information.



User	Lat	Long	Day 1	Day 2	Day 3	Day 4	...	Day 30
A	42.277528	-71.346809	Sick	Sick	Sick	Healthy	...	Healthy
B	42.280929	-71.237755	Healthy	Sick	Sick	Sick	...	Healthy
C	42.337041	-71.209221	Healthy	Healthy	Sick	Healthy	...	Sick

Fig. 3.1. Example of the data stored in the Strii database. For each user, the application saves user's initial location, and the last 30 days of health status data, whether user indicated if he/she was sick or healthy. Strii is also able to missing input.

B. Simulated Data

Currently, since we did not get a chance to transfer Strii to the cloud and reach out to a wide audience for real data, we wrote a robust program to simulate the collected data. Given some number of days, this program generates users and simulates interactions and disease spread. For our purposes, it uses a recovery period of 3 days and infection rate of 15%. An important feature of our program is that it is able to account for unentered data, since people do not always enter data every day. This allows it to repair missing information, which would be crucially important for the success of our future website, as human-entered data.

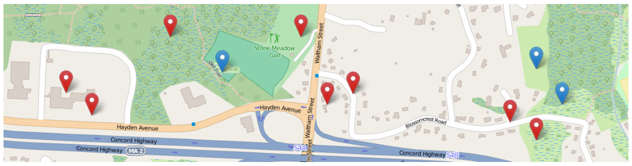


Fig. 3.2. An image of our map with simulated data points. Red means sick, blue means healthy.

C. User Interaction Algorithm

Below, we outline our process of data collection, subsequent analysis and forecasting, and presentation of results that we have developed. After opening the website in their favorite web browser, the user enters city name or zip code of interest in a sleek, modern search bar. If this city has already been requested by a different user, the forecast, which was stored from the last request, is immediately displayed atop the map. Otherwise, our Java backend starts a new background thread, where it obtains city information using the United States Census API, runs the requested simulation, and displays the results. Due to the small number of data points we have in relation to the total city populations, we use the scalability of social networks in order to run the simulations. For example, if we have 100 users in a given town, we map their locations and health statuses proportionally throughout the actual city of 25000 people. Then, using the method discussed in the Parameters subsection of Chapter 1, we find the optimal disease spread percentage to match the historical data, and forecast the continuation of the infection on the census-based city network using our backend simulation software. We save the results in the database, and present the current case distribution and forecast to the user.

D. Web Based Interface

The ultimate goal of our project is to put this application completely in the cloud. Users would register, and be able to select how they feel using a simple, minimalistic menu. This data would be stored in an online database, and displayed atop a map on the website. A user would be able to request a prediction of the future spread of a disease in a certain town or location on the map, and the Java backend would simulate with the requested parameters on the cloud using the acquired data, and the user would see a result. This is quite expensive to implement, which means that at the moment, our web-based interface is online, but our simulations are run locally on a computer. One advantage to this web-based system is that many people would be able to access it simultaneously. This would allow us to reach out to a wider audience. Next, the web based simulation would allow real-time monitoring of the spread of the disease, with constant updates. This would allow users to see the spread of disease in their local towns, allowing them to take immediate preventative action. Lastly, the website would be freely available on mobile devices as well, and would take up relatively little battery life, as opposed to a full application, which we have now.

IV. Results and Discussion

As a result of the network construction algorithms and the simulation program, a disease forecaster was developed. The interaction of the simulated data and the program produced results such as the graph in **Figure 4-1** where the data before the vertical black line was taken from simulated user inputs, and the data after the black line showed the forecaster's prediction of how the disease would spread. The program produced therefore has the capability to predict the spread of disease in a realistic town, using very few days and data points as a base for prediction. The simulated user inputs can easily be replaced by inputs acquired through Strii in order to have the program working on modelling real diseases rather than simulated ones.

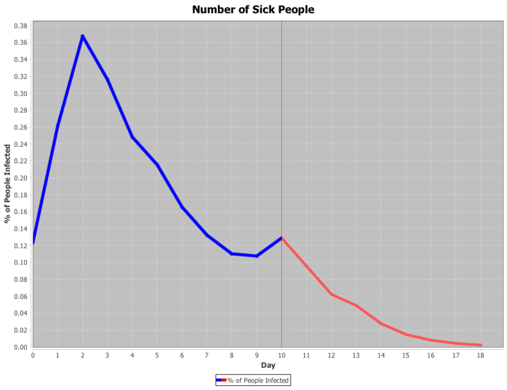


Fig. 4.1. This is the output graph of the forecaster program for Needham, MA. The data in blue before the vertical black line at day 10 shows the simulated user input received and scaled up to apply to the entire town. The data in red after the black line is the program's prediction for how the disease will continue to spread through the network.

Our suite also allows one to analyse how changes in some parameters affect the disease spread. The user can vary the selected parameter's value and observe the effect of changes on the length of epidemic and the total number of people affected by the disease, such as in **Figure 4-2**. These experiments allowed us to compare our simulations to experimental data and use more realistic parameters, yet it can also be used in the future in order to have the program suggest possible actions users can take in order to prevent disease spread.

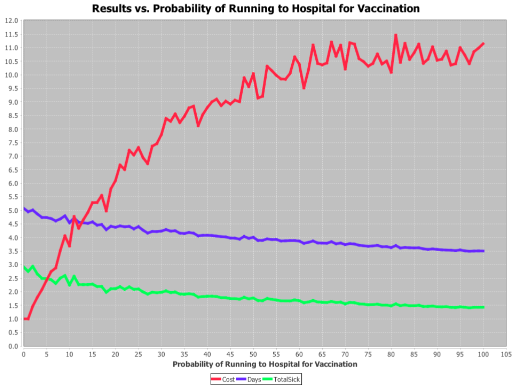


Fig. 4.2. This graph shows the impact of probability of people getting vaccinations on the spread of a disease. The parameter is the probability of running to the hospital for vaccination when one's friend catches the disease. The red line represents the cost in each scenario, the blue line represents the total length of the epidemic in days, and the green line represents the total number of people who got sick during the epidemic. In this scenario, the impact of more people getting vaccinated is small, and not worth the cost.

The developed application provides town-level information about disease spread for the user, and works for any location within the US. As an example, consider a hypothetical user from Lexington. As he logs into our web site, he is prompted to enter his/her health status. Currently, this is done through simulated users and data. For example, on December 20, 2014, Strii contained 46 users, 11 of which were identified by our program as those from Lexington. After a user prompts the program to analyse Lexington, the program returns a map of the geospatial distribution of Lexington users with their current health status, as depicted in **Figure 3-2** earlier in the paper. In addition, the program runs a predictive simulation, and the result of anticipated disease spread is also shown to the user, like in **Figure 4-1**. The user can also analyse the effects of vaccination or other interventions on the epidemic in this town. For example, **Figure 4-2**, shows the effect vaccinating an increasing number of people on the spread of the disease. The user can vary any parameter provided in our suite, as shown in **Figure 2-3**, to get a better understanding of possible scenarios and solutions.

V. Future Improvements

There are many areas of our project that we would like to improve in the future. First, we would like to improve accuracy of our forecast. Next, our website is not yet fully developed. We will work on improving the interface and adding more options for a more satisfying user experience. In addition, our program still runs locally on a computer, as we could not afford to make it accessible on the cloud by user request. In the future, we would like to upload our program to the JavaâĎć backend of our web site, to make everything run on the cloud and be accessible by millions of users. The future improved version of our app can be seen in **Fig. 5-1**. In this setup, Strii data of disease transmission and in-the-cloud forecasting software work together to present residents of the United States a website which shows how disease spreads, and how it is predicted to spread further, in their local city or town. This can then be connected to our existing software that analyses the effects of parameter change on disease spread and epidemic length in order to output suggested actions to the user.

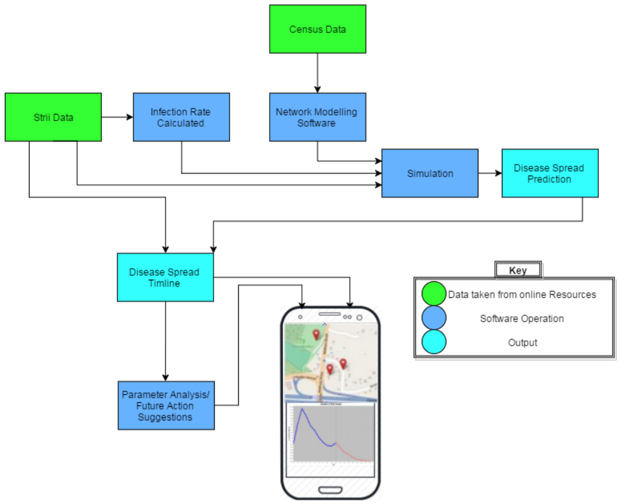

Census Data
Strii Data
Infection Rate Calculated
Network Modelling Software
Simulation
Disease Spread Prediction
Disease Spread Timline
Parameter Analysis/ Future Action Suggestions
Key
Data taken from online Resources
Software Operation
Output

Fig. 5.1. A map of how the final product should function. Using data collected with Strii, a simulation will run through a network that was constructed using census data. The result is a disease spread timeline which logs the existing and projected numbers for sick, infected, and recovered people by day. This timeline will then be used to create a map interface which allows the user to visually see the spread of the disease by day in a given town. Then, the data will be analysed again using parameter analysis in order to suggest to the user what actions they can take in order to prevent further spread of the disease.

ACKNOWLEDGEMENTS

Dr. Natasha Markuzon, for mentoring our project and leading us through the process

The MIT PRIMES program, and in particular Dr. Vyacheslav Gerovitch and Dr. Pavel Etingof, for enabling this research opportunity

Our parents, for being supportive in the process and offering their assistance where they could

REFERENCES

[1] Barabási, Albert-László, and Eric Bonabeau. "Scale-Free Networks." *Scientific American* May 2003: 60-69. Web.

[2] Göpfert, Martin C., and Daniel Robert. "The Web of Human Sexual Contacts." *Macmillan Magazines* 21 June 2001: 907-08. Nature. Web.

[3] "A Weekly Influenza Surveillance Report Prepared by the Influenza Division." *FluView*. Center for Disease Control and Prevention, n.d. Web.

[4] "Seasonal Influenza-Associated Hospitalizations in the United States."Influenza (Flu). Centers for Disease Control and Prevention, 24 June 2011. Web.

[5] United States of America. U.S. Department of Commerce. U.S. Census Bureau. N.p.: n.p., n.d. *2010 Census*. Web. 21 Sept. 2015. <http://www.census.gov/data/developers/data-sets/decennial-census-data.html>.

[6] Popescu, Andrei. "Geolocation API Specification." World Wide Web Consortium. Tim Berners-Lee, Jeffrey Jaffe, 24 Oct. 2013. Web. 17 Sept. 2015. <http://www.w3.org/TR/geolocation-API/>.