Spatial Outbreak Detection Analysis Tool:

A system to create sets of semi-synthetic geo-spatial clusters

by

Christopher A. Cassa

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

August 30, 2004

Author____
Department of Electrical Engineering and Computer Science
August 30, 2004

Certified by____
Peter Szolovits
Thesis Supervisor

Accepted by____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Syndromic Cluster Creation Tool:

A system to create sets of semi-synthetic geo-spatial clusters

by

Christopher A. Cassa

Submitted to the
Department of Electrical Engineering and Computer Science

August 30, 2004

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer [Electrical] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Syndromic surveillance systems, especially software systems, have emerged as the leading outbreak detection mechanisms. Early outbreak detection systems can assist with medical and logistic decision support. One important concern for effectively testing these systems in practice is the scarcity of authentic outbreak health data. Because of this shortage, creating suitable geotemporal test clusters for surveillance algorithm validation is essential. Described is an automated tool that creates artificial patient clusters by varying a large variety of realistic outbreak parameters. The cluster creation tool is an open-source program that accepts a set of outbreak parameters and creates artificial geospatial patient data for a single cluster or a series of similar clusters. This helps automate the process of rigorous testing and validation of outbreak detection algorithms. Using the cluster generator, single patient clusters and series of patient clusters were created – as files and series of files containing patient longitude and latitude coordinates. These clusters were then tested and validated using a publicly-available GIS visualization program. All generated clusters were properly created within the ranges that were entered as parameters at program execution. Sample semi-synthetic datasets from the cluster creation tool were then used to validate a popular spatial outbreak detection algorithm, the M-Statistic.

Thesis Supervisor: Peter Szolovits
Title: Director, Clinical Decision Making Group, MIT Computer Science and Artificial Intelligence Laboratory

Research Supervisor: Kenneth Mandl
Title: Research Director, Center for Biopreparedness at Children's Hospital Boston

Chapter 1: Introduction to Syndromic Surveillance

Many new suites of disease outbreak detection systems, also known as syndromic surveillance systems, are emerging in health-care facilities and centers for disease analysis throughout the world. [1-2, 7-9] Syndromic systems, especially software systems, have surfaced as leading outbreak detection mechanisms as there is an increasing yield of data available for analysis. Concern about bioterrorism threats is increasingly prevalent, so development of early outbreak detection to assist with medical and logistic decision support is increasingly important. The fundamental goal of syndromic surveillance systems is to be able to detect a small number of increased disease cases of one type of outbreak in a shorter time-frame than would be likely detected by acute physicians or medical administrators. [7] Several groups have quantified a range of detection goals, and some groups are attempting to discover small outbreaks (around fifteen extra visits or cases) within three days of the first abnormal visit to a clinic or emergency department. [11]

Evolution and Development of Syndromic Surveillance Systems

Syndromic surveillance systems have evolved a great deal since their emergence in the late 1990s. There are numerous types and implementations of these real-time outbreak detection systems that rely on a multitude of data sources. [25] Data streams range from sources as basic as primary-care location visit records and emergency department admission records to over-the-counter medication sales, web-based medical system visits, public and private-sector health-hotline calls, and even orange juice sales. [14-16] These sources have been selected based on the premise that the earliest signs of a potential outbreak are aberrations in the expected numbers of visits or relevant purchase volumes at the time of outbreak analysis.

In order for surveillance systems to be most useful, they should produce alerts in as timely a fashion as possible. Near real-time data acquisition and analysis is quickly becoming a standard practice. One example of a real-time syndromic surveillance system is the Automated Epidemiologic Geotemporal Integrated Surveillance (AEGIS) system, [1] software that tracks patient emergency department visits using live data streams from northeastern region hospitals. Patient address data is automatically geocoded and chief complaints are encoded into syndromic categories shortly after the patient has been admitted to a participating emergency department.

Software systems to conduct syndromic surveillance often utilize large existing hospital or retail databases to establish baseline parameters and covariate values. Those baseline measurements are used to find expected visit ranges for a specific time frame, which are then compared against directly observed values in a recent time-frame. Significant deviations from expected values occur with specific likelihood values. If computed likelihood values are low enough (those values that would correspond to a high confidence in abnormality of patient distributions,) then outbreak flags are raised. These so-called red flags should cause medical informatics professionals to further analyze a potentially emerging situation.

To increase the efficacy of surveillance systems, some organizations that track detailed patient data have chosen to group patients into syndrome categories. Several surveillance systems have found increases in sensitivity of outbreak detection and data analysis if the datasets being analyzed are grouped by syndrome. [2, 24] The national

ESSENCE project [9] has developed a categorization system that utilizes a subset of ICD-9 (International Classification of Diseases, 9th Revision) diagnostic codes for each syndrome that they use for syndrome grouping. A list of the ESSENCE syndrome groups is shown in Table 1.   Another system, the Realtime Outbreak and Disease Surveillance (RODS) project [8], has created a free-text Complaint Coder (CoCo) which takes a patient's chief complaint and assigns it to a syndromic category using a Bayesian classification scheme. Both of these systems then conduct surveillance analysis on separate groups of patients with similar syndromes.

Table 1: Syndrome Groups and Diagnoses from the ESSENCE project.
Source: http://www.geis.ha.osd.mil/GEIS/surveillanceactivities/ESSENCE/essenceinstructions.asp

Syndrome Groups	Representative Diagnoses
Respiratory	cough, pneumonia, upper respiratory infection
Gastrointestinal	vomiting, diarrhea
Neurological	meningitis, botulism-like symptoms
Dermatologic	Hemorrhagic (petechaie, bruising)
Dermatologic	infectious (vesicular rashes)
Fever	(unspecific fever, sepsis)
Coma	(coma, sudden death)

There has also been a good deal of discussion regarding hardware implementations for real-time surveillance, but they appear to be more difficult than software solutions in the short term. Effective physical detection of threats is much harder because it requires a greater infrastructure and expense. Software detection systems also generally rely on information that is already encoded into computer systems while hardware detection systems generally derive their data from new sources.

Data availability and variability issues create difficulty with temporal data analysis in a real-time fashion, but these systems have the potential to provide the sensitivity to detect a variety of public health outbreaks. [10] Temporal data filtering and smoothing is an expanding area of biosurveillance research which promise to improve many of the basic data variability concerns that interfere with optimal outbreak detection.

A number of biosurveillance systems have also started to integrate some notion of the spatial clustering of patients in area neighborhoods (and aberrations from normal levels of clustering) into their outbreak detection techniques. Two prominent techniques in the field that provide values of aberration in spatial clustering from normal spatial distributions are the M-Statistic [13] (described in more detail later) and the Spatial and Space-Time Scan Statistic, SaTScan. [11] The M-Statistic technique uses the deviation of the current distribution of inter-point distances (the distances between each patient and every other patient) from the distribution that would normally be expected, as a metric for spatial closeness. SaTScaN uses either a Poisson-based model, a Bernoulli model, or a space-time Permutation model, using user-provided patient visit data as the source of the underlying expected distribution.

Geocoding of patient addresses

All addresses are represented by latitude-longitude pairs in the cluster creation tool and in the implemented spatial detection algorithm. This standard was chosen because it is the most general and it avoids ambiguities that may arise from duplicate addresses or artificial boundaries such as zip codes. Patient addresses are geocoded into this format and then patients are categorized by their chief complaints (which are almost always recorded and coded by medical institutions.) [20-23] This allows doctors or medical practitioners to determine whether there is a correlation between patient location, time of visit, and symptoms. If this information is gathered for all patients at all times, it is easier to detect an outbreak when it occurs, because the outbreak data can be compared against expected baseline data from similar time periods. [25]

Calculating Sensitivity and Specificity of Outbreak Detection Systems

The quality of a detection system can only be assessed by observing its behavior on test cases of interest. To do so, we need to find or create valid test clusters of patients with disease and then measure how well our system is able to detect those patients. Detection efficacy is usually measured in terms of the sensitivity and specificity of the detector.

This project focuses on both problems by creating valid artificial test clusters that vary under a large variety of realistic parameters and then detecting those test clusters using a spatial detection technique. In order to create realistic detection mechanisms for large datasets, those datasets need to be tested using realistic physical outbreak data. A cluster creation tool that creates simple geo-temporal clusters of artificial patient data is part of this work and is freely available for download (see Appendix B). The current implementation is described here and its use in validating the real-time M-Statistic spatial scanning algorithm is described in Chapter 4.

Datasets for benchmarking performance

Datasets for evaluating the performance of algorithms used in outbreak detection may be measured using authentic data, synthetic data, or combinations of the two. Two kinds of purely authentic datasets are possible. One is genuine syndromic data that is contemporaneous with a known specific outbreak. The outbreak could be a large-scale event such as a winter influenza surge. [14] Alternatively, it could be a more circumscribed event such as a diarrheal outbreak. [15] The dataset would contain the background of ordinary disease or symptom occurrence and the signal of the actual outbreak. A second type of authentic dataset is a hybrid one containing background from a regional surveillance system spiked with cases from a known outbreak. This approach was taken when over-the-counter medication sales data were spiked with an outbreak based on the Sverdlosk incident. [16] Alternatively, one can also construct a hypothetical baseline and impose and inject actual or simulated signals. While this approach is valid, there is little need to simulate background activity, given the readily available abundance of real signal streams from surveillance systems.

The approach that we explore in detail is to superimpose simulated signal onto authentic baseline. This tactic offers the opportunity to explore the impact of controlled variations of the signal characteristics. Broadly speaking, there are two main approaches to creating this simulated signal. One could produce multistage, multivariate mathematical models to produce the signal. Alternatively, it is also possible to define a series of parameters enabling the generation of a controlled feature-set simulated signal. For example, a complex mathematical model [18] might begin with a particular form of aerosolized anthrax being dispersed under a certain set of atmospheric conditions over a specific geographic region with a well-characterized population demographic. The number of susceptible individuals might be estimated and their subsequent behaviors modeled. The resulting impact on the syndromic surveillance data set, be it retail sales, primary care visits or emergency department visits, could be projected. The difficulty with this type of approach to creating simulated signals is that such detailed models require a great deal of hand-crafting. Furthermore, many different models might be needed to provide realistic signals corresponding to various plausible scenarios of outbreak or attack.

Any model that we choose to generate data will make certain assumptions about the simulated events that it models, and the resulting data may be misleading either because the assumptions are inappropriate to certain kinds of events or because the models may fail to take important aspects of the event into account. An example of inappropriateness may be an assumption that all bioterrorist outbreaks lead to an initially exponential rise in observed cases; though reasonable for infectious diseases, this would be inappropriate for non-infectious toxic agents. An example of an overly-simple model might assume that all observed cases occur simultaneously at some specific time after exposure.

We have tried to find a compromise between taking on the overly difficult task of accurately modeling many different types of outbreaks and accepting too simplistic a generative model that fails to simulate the dynamics of actual outbreaks of interest. In our model, we can vary the number of cases in an outbreak, the temporal pattern with which they appear in the data, and the sizes and locations of spatial clusters within which they appear. We do not attempt, however, to include much more detailed behaviors that would require modeling notions such as wind spread or varying susceptibility of different populations.

Chapter 2: Parameterizing an Outbreak

Background noise can be spiked with additional cases configured as spatial or temporal clusters, describable as a *controlled feature set*. A variety of adjustable parameters, described below, enable manipulation of the simulated outbreaks. Optimally, a training dataset should be modeled and the artificial outbreak signal should be injected into a validation dataset, though there may not always be sufficient data to do so. If there is not, the artificial outbreak signal may be injected into the same data used for training. There are many components in a controlled feature set that are relevant in describing a specific type of outbreak. Some of these components are specifically related to the temporal distribution of patients, some are specifically related to the spatial distribution of patients, and some only rely on the total number of patients that arrive.

Spatial features

A semi-synthetic data cluster has a spatial controlled feature set, which includes a number of important spatial components. The spatial relationship among the synthetic patient cases, which are represented as geocoded coordinates (latitude and longitude) is important – some primary components are whether the points are added in a uniform fashion, a random fashion, and what the overall physical shape of the cluster pattern will be. The cluster may also be described in terms of a maximum cluster radius (or other possible maximum distance in a non-circular outbreak), the density of the distribution of cases within that radius, and also by the relative location (or the 'angle') from a fixed point such as a hospital or urgent-care facility. Simulating spatial clusters raises additional challenges as well, including the identification of realistic locations for simulated cases, based on the spatial features of a region--such as the locations of housing and of bodies of water.

Outbreak Duration

This describes the number of days that a specific geo-temporal outbreak signal should span. It is useful to execute simulations over a range of outbreak durations and a number of factors might influence the range chosen. Different agents can cause outbreaks of varying lengths -- while a surge in influenza activity might last several weeks, an outbreak of meningitis in a college dorm might only last for several days. Furthermore, the temporal window used by the detection system may have substantial impact on how outbreaks of different magnitudes are detected. If the detection window were based, for example, on a sliding moving average of seven days, two or three day long outbreaks will be smoothed out; under certain conditions this smoothing may dilute the signal. Conversely, outbreaks gently trending upward in numbers might not be detected with a shorter sliding window.

Outbreak spacing

An efficient way to measure outbreak detection performance and the factors that influence it is to spike a data stream with many individual outbreaks. Generally, the more outbreaks presented to a model-based system, the more accurately the system's detection performance can be characterized. In order to maximize the number of simulated outbreaks in the dataset, one can introduce multiple non-overlapping outbreaks in a single dataset (e.g. a five day outbreak beginning on day 1, another five day outbreak beginning on day 11, another on day 21, etc.). The outbreaks are then removed and replaced by a different set of non-overlapping outbreaks and again presented to the system (e.g. days 2, 12 and 22). For measurement purposes, it is critical to ensure that all individual outbreaks are temporally isolated -- meaning that any response to the previous outbreak has been completely eliminated from the system before the next outbreak is encountered. For individual outbreaks to be temporally isolated, it is necessary for the time-filtering window, if it is larger than one day, to be smaller than the number of days between injected outbreaks. Such temporal isolation is critical for accurate measurements of detection performance, though it will not directly address the ability of the system to detect overlapping outbreaks. By shifting the outbreaks in time, the outbreaks are affected by different regions of noise. By spacing outbreaks throughout the year, the effect of seasonal changes in the background on outbreak detection can be measured as well. Understanding the effects of different regions of background noise cannot be accomplished without the use of simulation.

Outbreak temporal progression

The time course of an outbreak spreading through a population can follow any one of many paths, effectively producing a signature shape, related to an epidemic curve. For example, a highly infectious disease such as smallpox could spread exponentially over time, while a point-source exposure, not contagious person to person, such as an anthrax release, would be unlikely to grow exponentially. Several canonical shapes of temporal progression may be used in simulations to characterize the detection performance of surveillance systems. Flat outbreaks introduce a fixed number of extra visits per day for the duration of the outbreak -- for example, [10,10,10,10,10] extra visits for a five-day outbreak. Linear outbreaks introduce a linearly increasing number of extra visits per day over the course of the outbreak -- for example, [5,10, 15, 20, 25] extra visits over a five-day outbreak. Exponential outbreaks introduce an exponentially increasing number of extra visits per day over the course of the outbreak -- for example [2,4,8,16,32] extra visits over a five-day outbreak. Sigmoid shaped outbreaks mirror epidemiological phenomena where the number of affected individuals rises exponentially at first, then slows down until it plateaus at a new fixed level -- for example [2,4,8,12,14] extra visits over the course of a five-day outbreak. Alternatively, a model of more complex shape, described by a multinomial, such as the Sverdlosk [5] outbreak, might be desirable.

Outbreak magnitude

Because the minimum detectable size of an outbreak is often of interest, outbreak detection performance should be tested over a range of signal magnitudes; detection performance may vary substantially depending on these magnitudes. This variability is primarily due to the changes in signal-to-noise ratio that result from different outbreak sizes. For small outbreaks that are at or near the “noise floor” of the model -- the usual level of random variability in the model’s predictions -- the detection performance is typically very poor, because it is hard to distinguish outbreaks from the random noise of the model. As the relative size of the outbreaks increases, identifying the outbreaks in the presence of the noise becomes easier. Once the outbreak magnitude is large enough such that the noise does not effectively mask it all, the outbreak detection performance of the system typically plateaus at or near perfect detection.

In order to identify an appropriate range of outbreak magnitudes for the simulations, it is worthwhile to characterize the error, or noise profile of the model. To do so, the daily forecast errors of the model, defined as the forecast value minus the actual value for each day, must be calculated. The error profile can be visualized by plotting a histogram of these daily forecast errors, and standard deviation of the error distribution. As a general rule of thumb, outbreak magnitudes should range from near zero to at least twice the standard deviation of the forecast error. As an example, consider a model of emergency department visits with mean of 140 visits per day, and an error profile with a standard deviation of 20 visits. In this case it is helpful to run simulations of outbreaks ranging in magnitude from 0 to 40 visits per day. This range can be sampled in intervals of 5, yielding the following set of outbreak magnitudes [0,5,10,15,20,25,30,35,40].

It is important to note that the error profile of a model could vary through the year due to seasonal differences in the variability of the signal. For example, respiratory visit rates could vary much more unpredictably in the winter than in the summer. In such cases, it may be useful to construct separate error profiles for different seasons in order to tailor the detection test to each season.

Metrics for detection performance

Sensitivity and specificity

There is always a tradeoff between sensitivity and specificity and the ability to detect outbreaks must be balanced against the cost of false alarms. [4] For evaluation purposes, it can be useful to hold sensitivity or specificity constant when plotting the other against another variable, such as outbreak magnitude, or outbreak duration. For example, specificity may be held constant while plotting sensitivity vs. outbreak magnitude. For each outbreak magnitude, the alarm threshold should be tuned until the desired number of false alarms -- and thus the desired specificity -- is achieved. The rationale for tuning the alarm threshold to a desired specificity for the outbreak detection algorithm is rooted in the fact that a certain level of false alarms is acceptable to the system so that it will be optimally sensitive with a specific number of user-specified alerts. At this point the resulting sensitivity under these conditions is measured. This process is repeated for each outbreak magnitude, ultimately yielding a plot of sensitivity vs. outbreak magnitude with specificity fixed. The likelihood of not having an alarm when there is no signal (specificity) can be measured simply by running the model on the baseline data without inserted artificial outbreaks.

Overall outbreak detection vs. outbreak day number

Because the outbreaks presented to the system typically will be longer than one day, sensitivity and specificity can be measured either in terms of detection of specific outbreak days or of the overall outbreak. Using the outbreak-day approach, each day is considered a separate, independent case -- if a particular five-day outbreak is detected on three of the days, but missed on the other two days, there are three successes (true positives) and two failures (false negatives). Similarly, each of the intervening non-outbreak days is considered independently when calculating false positive and true negative rates.

Using the overall outbreak detection approach, each outbreak is viewed as a single entity; if the outbreak is correctly detected on any one of the outbreak days (logical OR), the system has produced a true positive. An alternative criterion for a true positive is that the outbreak was correctly detected on a majority of the outbreak days. When reporting the overall outbreak sensitivity ("The system detected x% of all the outbreaks presented to it."), it is very helpful, in conjunction to present full sensitivity and specificity statistics are reported using the "outbreak-days" approach.

Receiver operator characteristic (ROC) curves

The tradeoff between sensitivity and specificity is well portrayed by ROC curves, which plot sensitivity vs. [1 – specificity]. For tests that have no diagnostic value, the ROC curve is a straight line along the diagonal of the plot. Plots of tests with higher diagnostic value have the line curved away from the middle of the plot. The area under the ROC curve can thus be used as a measure of the diagnostic value of a test. [12] The diagnostic value of two tests can be compared by comparing the areas under their respective ROC curves.

Chapter 3: Creating a Cluster Generation Tool

A cluster generation tool that creates single clusters and related series of synthetic patient clusters was created that allows facilitates efficient spatiotemporal outbreak detection testing. The tool is designed to create clusters of patients with similar syndromes in a user-specified distribution. The cluster data points will be added to a larger database (with many patients over a long period of time) and then each dataset can be tested using detection algorithms to determine whether that detection models can uncover a specific type of artificial cluster.

The cluster generation tool can also generate various sets of clusters that range in value over a single parameter to rigorously test detection algorithms. The generator can also be easily altered to create additional types of clusters that follow other spatial and temporal distributions.

GIS Datapoints and Earth Surface Measurements

It was necessary to calculate several fundamental GIS measurements to create the basic geospatial data engine for the cluster generator. Artificial patient datapoints are to be added following several parameters, many of which require the creation of a relationship between Earth surface measurements in meters and degrees of latitude and longitude. The number of degrees of latitude and longitude varies per number of meters of distance measured on the Earth's surface at each latitude and longitude. GIS conversions that would properly calculate these distances were necessary to create.

An important consideration was picking an appropriate distance conversion model for the calculations. Sphere equations break down significantly at small distances, but the Haversine formula is correct at almost all populated points on the Earth's surface. At several Longitude and Latitude pairs the Haversine formula was tested for accuracy. At each Latitude-Longitude, the distance between two datapoints was measured and then compared to the result that Microsoft Corporation's MapPoint GIS mapping tool produced for the same measurement.

Using the Haversine Formula

To use the Haversine calculation, the Earth has radius R, and the locations of two points in spherical coordinates (longitude and latitude) have names lon1,lat1 and lon2,lat2. In psuedocode, the Haversine Formula [3] is calculated with the following code:

```
  dlon = lon2 - lon1
  dlat = lat2 - lat1
  a = (sin(dlat/2))^2 + cos(lat1) * cos(lat2) *
(sin(dlon/2))^2
  c = 2 * atan2(sqrt(a), sqrt(1-a))
  d = R * c
```

This system gives mathematically and computationally exact results. The intermediate result c is the great circle distance in radians. The great circle distance d will be in the same units as R.

Distance Measurement Methods

Programmatic methods were written to solve a number of other geocoding basic needs, generally involving latitude-longitude datapoints and physical Earth surface distances. Inside the main GIS class, we have three primary methods to handle these basic data conversions. The first is a method to find the distance between to latitude-longitude points, which uses the specific latitude-longitude of the first datapoint to create a ratio of degrees per physical unit of distance (meters were used) in each direction (N-S latitude and E-W longitude). The ratios are calculated by first creating artificial datapoints in each direction that are a 0.05 degrees to the north and to the east, and the corresponding physical distances are calculated using the Haversine Formula (discussed above) and then the ratios are computed (using the artificial datapoint distance divided by the calculated physical distance.)

A second method was created to find the second latitude-longitude point that is a certain physical distance, measured at a specific angle, from a first latitude-longitude datapoint. The angle was measured from the Euclidian x-axis and increased in a counter-clockwise form. The distance from the first point was entered in meters, and the output was a second GIS datapoint that was related to the first point and the parameters entered.

Finally the methods were created that found the number of degrees latitude-longitude per unit of physical distance, in meters (in each respective direction.) These methods were then used in the above two methods for those basic calculations.

Basics of Date Algorithms Implemented in Cluster Generator

Cluster data points in a real outbreak are added at different rates, depending on a number of factors including the type of outbreak and the type of people it affects. One of these factors is the temporal growth pattern of the cluster. Three date algorithms (temporal growth algorithms) were implemented to model the ways in which a disease might grow in time: a random growth-pattern, a linear growth-pattern, and an exponential growth-pattern.

The random, linear, and exponential models were chosen as a base set because they were the simplest, most widely-applicable models. The random algorithm simulates a disease which randomly affects the population. An example of such an outbreak is the recent Anthrax attacks where several people in the United States were affected in a scattered random pattern. The linear algorithm models an outbreak which increases at a constant rate, which may be more applicable for a non-communicable infection that may be spreading spatially with respect to time. The exponential algorithm is probably the most realistic algorithm for some communicable infections, because at first a disease will affect only a few people and then grow exponentially at a rate determined by the motility of the population, the incubation period of the disease being studied, the population susceptibility rate, and the transmission rate for that disease. The rate of growth for the linear and exponential algorithms can be adjusted with a multiplier, based on those factors for how the disease spreads.

When designing the system, the date-assignment algorithms were separated from other parts of the implementation in the rest of the system so that it would be easy to add more date algorithms to model different types of outbreaks.

The algorithms are described in Fig. 1, below. For the random algorithm, a random number is generated between 1 and the number of days in the cluster. This produces a random date distribution. For a linear distribution, we divide the day value (1, 2, 3, etc.) by the sum of the day values and multiply it by the total number of points to determine the fraction of the total points that occur on that day. This is also scaled by a multiplier to alter the rate of linear growth. Similarly for the exponential distribution, we divide $e^{(\text{multiplier*day number})}$ by the sum of all of the $e^{(\text{multiplier*day number})}$ day values and multiply by the number of points to determine the number of points that occur on that day. Figures 2 and 3 show examples of linear and exponential growth-pattern probability distribution estimations.

Random:

$$\text{Number\_of\_points\_in\_cluster} = (\text{rand\_\#})(\text{mod\_\#\_of\_days})$$

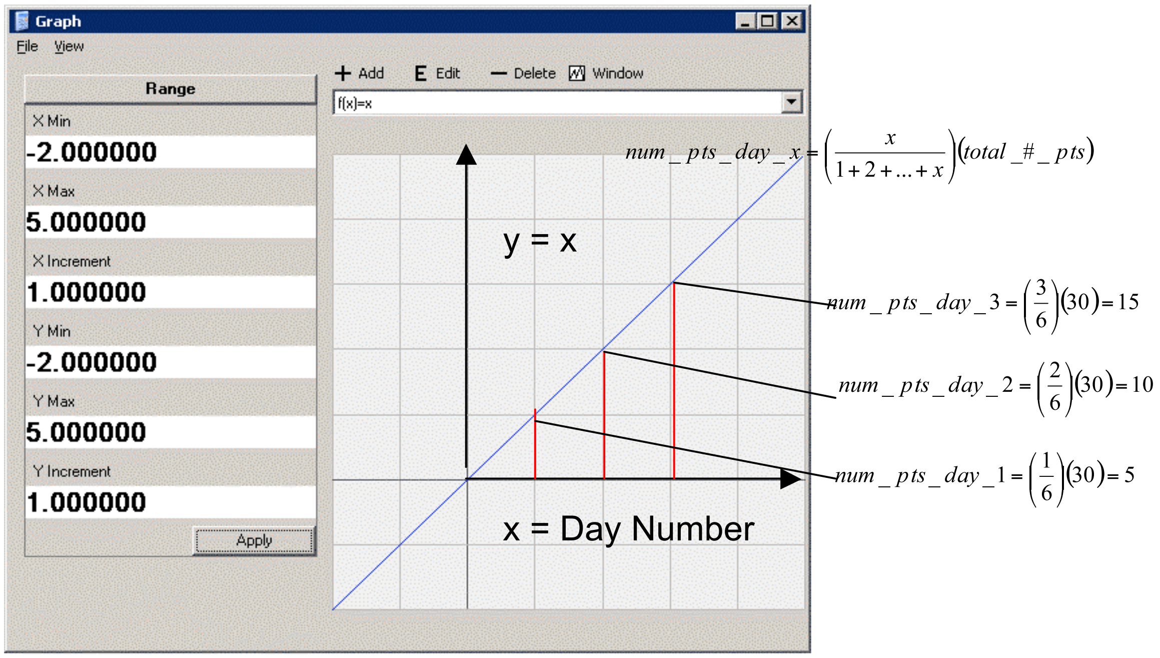
Linear:

$$\text{Number\_of\_points\_in\_cluster} = \left(\frac{\text{day\_value}}{\text{total\_day\_values}}\right)\left(\text{total\_number\_of\_points}\right)\left(\text{multiplier}\right)$$

Exponential

$$\text{Number\_of\_points\_in\_cluster\_on\_that\_day} = \left(\frac{\text{e} \wedge (\text{multiplier} * \text{day number})}{\text{total day values}}\right)\left(\text{total\_number\_of\_points}\right)$$

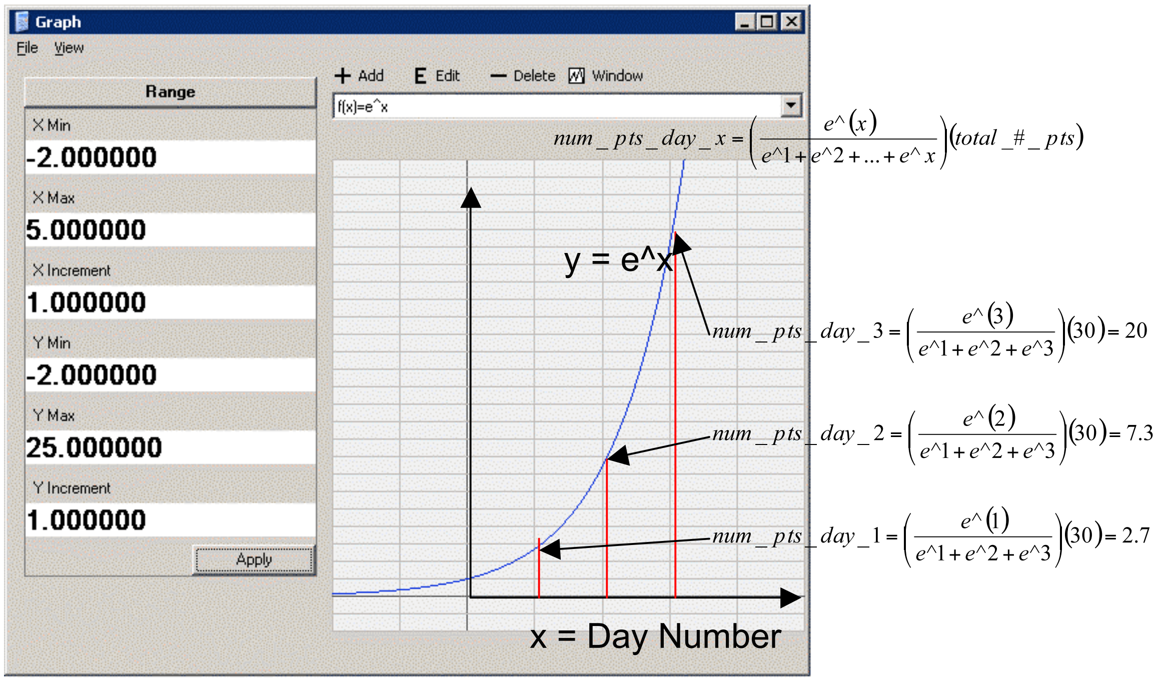
Figure 1: Three date algorithms have been implemented in the Cluster Creation program: Random, Linear, and Exponential.


Graph
File View
Range
+ Add
E Edit
— Delete
Window
Apply
y = x
x = Day Number
$num\_pts\_day\_x = \left(\frac{x}{1+2+\ldots+x}\right)(total\_\#\_pts)$
$num\_pts\_day\_3 = \left(\frac{3}{6}\right)(30) = 15$
$num\_pts\_day\_2 = \left(\frac{2}{6}\right)(30) = 10$
$num\_pts\_day\_1 = \left(\frac{1}{6}\right)(30) = 5$

f(x)=x

X Min -2.000000
X Max 5.000000
X Increment 1.000000
Y Min -2.000000
Y Max 5.000000
Y Increment 1.000000

Figure 2: An example of the linear date algorithm estimation for thirty points spanning three days.


Graph
File View
Range
+ Add
E Edit
— Delete
Window
Apply
y = e^x
x = Day Number
$num\_pts\_day\_x = \left(\frac{e\hat{}(x)}{e\hat{}1+e\hat{}2+\ldots+e\hat{}x}\right)(total\_\#\_pts)$
$num\_pts\_day\_3 = \left(\frac{e\hat{}(3)}{e\hat{}1+e\hat{}2+e\hat{}3}\right)(30) = 20$
$num\_pts\_day\_2 = \left(\frac{e\hat{}(2)}{e\hat{}1+e\hat{}2+e\hat{}3}\right)(30) = 7.3$
$num\_pts\_day\_1 = \left(\frac{e\hat{}(1)}{e\hat{}1+e\hat{}2+e\hat{}3}\right)(30) = 2.7$

f(x)=e^x

X Min -2.000000
X Max 5.000000
X Increment 1.000000
Y Min -2.000000
Y Max 25.000000
Y Increment 1.000000

Figure 3: An example of the exponential date algorithm estimation for thirty points spanning three days.

Parameters for Cluster Creation Tool

Input Parameters: Single Cluster

The following are the current input parameters for creating a single patient cluster (Table 1). Parameters can be added or deleted by updating the GUI and modifying the GenerateCluster function in the cluster generator class.

Table 2: Parameters that can be altered when creating a single cluster.

Outbreak Parameter	Description of Outbreak Parameter
Cluster ID Number	User specified reference or identification number for each cluster. Within each cluster, every point will have its own identification number which will range from 0 to n-1, where n is the number of cluster points.
Number of Points in the cluster	Number of patients or points in the generated cluster.
"Reference Point" GIS Location	The latitude-longitude coordinates of the reference point coordinate, which could be a hospital or a primary care facility, for example.
Maximum cluster radius	The distance of the outermost point in the cluster from the center of the cluster.
"Angle" from the Hospital	The angle of the cluster from the hospital with respect to the latitude/longitude of the reference GIS location, measured counter-clockwise from due east as zero degrees, using unit circle convention.
Distance from the hospital	The distance of the cluster center point from the hospital.
Numbers of Days the Cluster should span	The number of days from when the first person shows symptoms to when the last person does.
Date Algorithm	This specifies which of the three (additional are are possible) to choose.
Description and output filenames	The user can specify where the cluster data and user-specified cluster description will be written.

Parameters for Cluster Creation Tool

Input Parameters: Sets of Clusters

The tool can generate a series of cluster which vary in one of five ways, which are described below. The user specifies the number of clusters to create and selects which parameter to vary. He also specifies a minimum and maximum value for which to vary the parameter, which will also be described below.

- *Number of points in the cluster* – The total number of points in the cluster will be varied over all the clusters.
- *Maximum distance from cluster center point (radius)* – A number of clusters will be created, all in the same location, but each with a different radius.
- *Number of days of cluster duration* – This varies the number of days that each cluster spans.
- *Angle around the hospital* – A number of clusters will be created around the hospital with the angle varied.
- *Distance from hospital* – The clusters will be along the same line at different distances from the hospital.

Output Files from Cluster Creation Tool

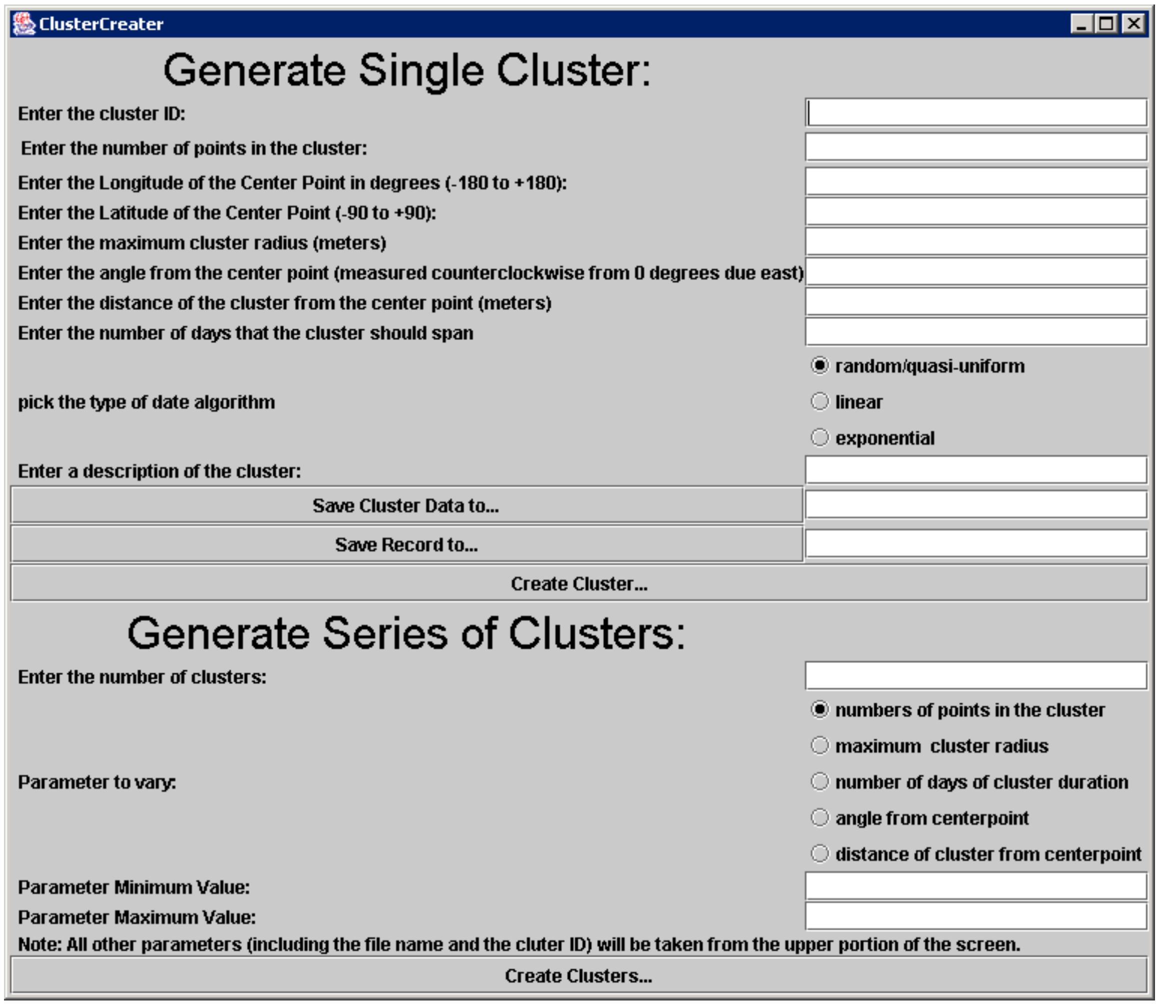
The Cluster Creation Tool creates at least two output files every time it is run. These file names are specified by the user in the input parameters. The program creates a data file for every cluster that is created and a record file describing the cluster. The data file contains the cluster point ID # (assigned numerically from 0 to the number of points minus 1), the longitude and latitude of the cluster point, and the relative date of the cluster point. Figure 4 displays a sample output file. When generating a series of clusters, the program automatically generates n files with appended identifiers. For example, when creating 10 clusters, the program would generate 10 files of the output file name each appended with a number from 0 to 9.

```
0,-71.10793353860446,42.360580952101984,4
1,-71.10616263466369,42.3636042625572,3
```

Figure 4: Sample data file output.

Cluster Generator User Interface

The user interface, displayed below in Figure 5, for the cluster generator uses the Java Swing toolkit to allow users to quickly select desired parameters for cluster creation. The user enters appropriate values into the text boxes for the type of cluster creation (or series cluster creation methods) and then selects using radio button the desired time-course for cluster entries. Users may also select where in the file system to save the output files from the cluster generation. There are two distinct output files for which the user should select a destination and a filename. Once the user has made the appropriate selections, it is possible to create a single cluster or a series of clusters by clicking the appropriate button. The user interface will alert the user whether the clusters were created successfully, and if they were not created successfully, an alert window will appear which describes the error or exception that was thrown from the generator program.


ClusterCreater

Generate Single Cluster:

Enter the cluster ID:

Enter the number of points in the cluster:

Enter the Longitude of the Center Point in degrees (-180 to +180):

Enter the Latitude of the Center Point (-90 to +90):

Enter the maximum cluster radius (meters)

Enter the angle from the center point (measured counterclockwise from 0 degrees due east)

Enter the distance of the cluster from the center point (meters)

Enter the number of days that the cluster should span

pick the type of date algorithm: [✓ random/quasi-uniform] linear exponential

Enter a description of the cluster:

Save Cluster Data to...

Save Record to...

Create Cluster...

Generate Series of Clusters:

Enter the number of clusters:

Parameter to vary: [✓ numbers of points in the cluster] maximum cluster radius number of days of cluster duration angle from centerpoint distance of cluster from centerpoint

Parameter Minimum Value:

Parameter Maximum Value:

Note: All other parameters (including the file name and the cluter ID) will be taken from the upper portion of the screen.

Create Clusters...

Figure 5: Cluster Generator Graphical User Interface

Results of Cluster Generation

Two sample scenarios for use of the cluster creation tools are described below. The first example demonstrates the use of the cluster creation tool to create a single artificial patient datapoint cluster, while the second demonstrates the use of the series of patient clusters creator.

Sample Program Input and Output:

Example 1: Creating a Single Patient Cluster:

The first example is the creation of a single artificial patient datapoint cluster. This first cluster includes patient reports of an outbreak that spans five days with the following parameters:

Cluster ID:		442 (randomly assigned by user)
Number of Points in Cluster:		30
Centerpoint:		(MIT)
	Longitude:	-71.09516
	Latitude:	42.35666
Cluster Radius:		600 m
Angle from centerpoint:		90
Distance from centerpoint:		1600 m
Number of Days:		5
Cluster Description:		Linear time-growth cluster North of MIT

This linear time-growth cluster has been placed approximately 1 mile (1600m) due North of MIT, which is described by the latitude-longitude datapoint (-71.09516, 42.35666).

Example 1: CSV Text Output Sample:

The CSV (comma-separated output file, described above,) for the single cluster example is partially described by the portion of the sample output below. Eleven points are listed, but thirty are actually enumerated in the output file. It is of value to notice that the dates are enumerated using the linear algorithm and so there are two points on the first day, four on the second day, six on the third day, continuing in the file to complete enumeration of the thirty patient points and five days spanned.

```
0,-71.09600452536358,42.37455407329273,1
1,-71.10149672138236,42.365894560466806,1
2,-71.0954755413253,42.373890954435524,2
3,-71.08968377859539,42.37242100053542,2
4,-71.09281946336338,42.36955324904336,2
5,-71.09564524977307,42.371694560897,2
6,-71.09345472615571,42.370979504450304,3
7,-71.09983295495935,42.369683605959985,3
8,-71.09781606117451,42.37282397457113,3
9,-71.09685871099056,42.37540065852763,3
10,-71.0921214185705,42.37216701505921,3
... (to point with ClusterID 29)
```

Example 1: Microsoft MapPoint GIS Mapping Output for Single Cluster:

The output csv file was next imported into a GIS analysis tool created by Microsoft, called MapPoint 2002. The map below, Fig. 5, illustrates the faithful creation of the set of points in the single patient data cluster. The date of each patient in the artificial outbreak cluster is described by the color of the patient circle.

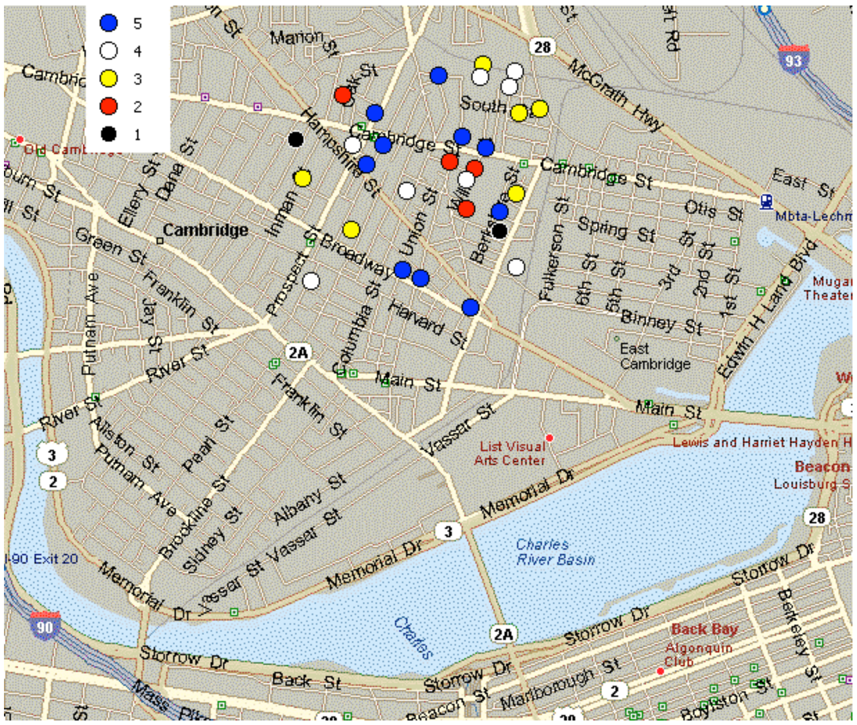


Figure 6: A single linear time-growth cluster north of MIT.

Example 2: Creating a Series of Clusters (Varying Angle)

Varied angle around centerpoint (MIT) and created 4 clusters.

Minimum angle:	0
Maximum angle:	270
Number of Clusters:	4
Centerpoint:	MIT (same LongLat point as above)
Cluster Radius:	400m
Distance from MIT:	3000m

In this example, the angle around MIT was varied in the series of four clusters that were generated. The series cluster generator created four files automatically, and each file was imported into MapPoint with a different color, and charted, shown below in Fig. 6.

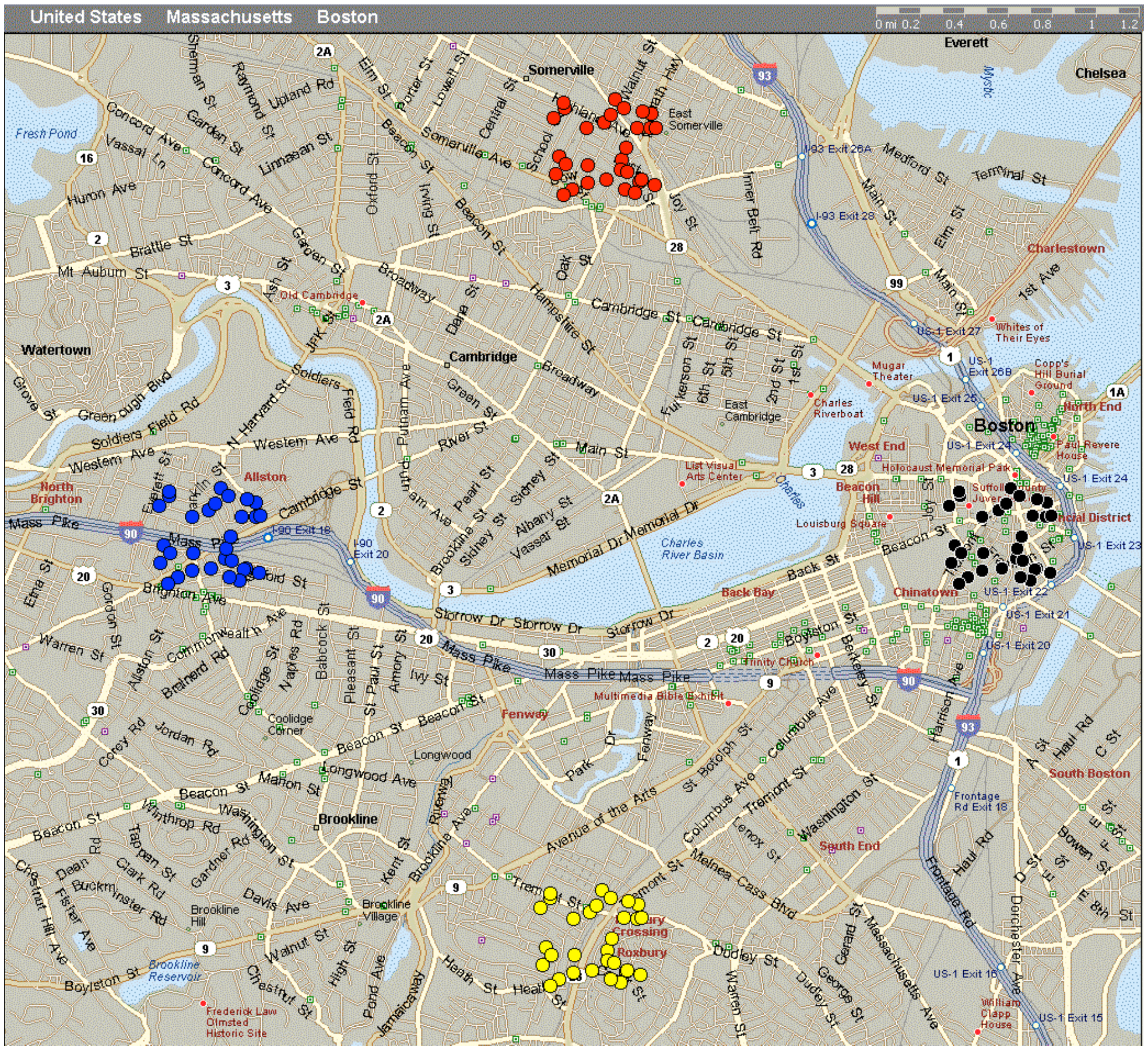


Figure 7: Creation of a series of four clusters about MIT (with the angle varied.)

Creating a Set of Outbreak Datasets for Spatial Detection Algorithm Validation

To rigorously evaluate spatial detection algorithms, it will be valuable to create a set that contains patient cluster datasets that span a reasonably-valid range of parameter values, as described in Chapter 2. A set of 360 artificial patient clusters were created using the cluster generator that varied over ranges of distance from the hospital, total number of patients in the cluster, size of patient cluster (or cluster density), relative angle with respect to the hospital,

Evaluating the Accuracy and Uniformity of Generated Clusters

Semi-synthetic datasets created by the cluster generator fall within a specified set of parameter-based boundaries. Cluster data points are created randomly within the domain defined by those parameters, so it is important to verify that the clusters are accurately and are close to uniformly generated.

To measure uniformity of generated clusters, ten test clusters were created with one hundred points in each cluster. The centroid of each set of cluster points was then calculated and compared to the specified centerpoint of that cluster. In every case, the cluster centroid was within five percent of the specified cluster radius, in distance, from the specified centerpoint. This result demonstrates that the datasets are uniform, on average, with a large number of points, as would be expected with a random distribution.

To measure the accuracy of the geocoding engine, 360 clusters were made around a single centerpoint, varying the angle evenly (1 degree added per cluster,) and they each had a cluster centerpoint that fell exactly on the circle that they should have formed. This same test was conducted at 5 random latitude-longitude locations and the same results were achieved.

Chapter 4: Implementation and Validation of M-Statistic Real-Time Spatial Detection

A spatial detection algorithm was implemented to evaluate the cluster generation program described in the previous chapter. The specific algorithm, the M-statistic, is an interpoint-distance based detection algorithm that analyzes patient distributions with single-syndrome, single-week sets of patients in an area. A rich treatment of the basic algorithm and the metrics used is provided in the original paper. [13]

To create a baseline set of values that would be expected to evaluate how spatially clustered each week's patients are, historical patient address data are analyzed. Four years of Children's Hospital Boston data were taken and seasonally separated for each syndrome. For each summer, for example, data for all patients with each syndrome were taken and combined. Each patient's distance to every other patient over the four summers was calculated in miles. With N patients in those four summers, there are a total of $N * (N - 1) / 2$ distances that are calculated.

All of those distances, in increasing order, were placed into 10 equally sized bins, such that each bin contained approximately 10 percent of the total distances. The cutoffs were then determined for those seasonal, specific syndrome bins, as distances in miles. The first bin, for example, started with 0 miles and the final bin ended with 100 miles, because the total distance from the hospital that was maximally allowable in the study is 50 miles. (Two patients can each be 50 miles from the hospital, on opposite sides of the hospital, for a total inter-point distance of 100 miles.)

After the baseline distribution values have been calculated, it is possible to evaluate how different the distribution of patients from a specific single week might be. For *M* patients that arrive at the emergency department in a specific week, the $M\,(M-1)\,/\,2$ inter-point distance values are calculated. Those values are then placed into the appropriate bins that were established in the baseline distribution calculation. Now, inside the bins, there is an uneven distribution – originally the bins had each contained ten percent of the total inter-point distances. At this stage, because all of the inter-point distances in this calculation only depend on the patients that have arrived in the ED in the present week, these distances will likely be equally distributed into the bins with ten percent in each.

The M-Statistic for the current week is then calculated using the formula below. The transpose of the vector observed minus expected values for each bin entry (ten entries for the ten bins) is taken and multiplied by the column vector product found by multiplying the inverse of the seasonal covariance matrix with the observed minus expected vector. The final value recovered from this multiplication is a single inner-product value

$$M = (obs - exp)^T \times [S^{-}(obs - exp)]$$

obs =1 x 10 matrix of normalized observed proportions

exp =1 x 10 matrix of normalized expected proportions

S = 10 x 10 variance-covariance matrix of the baseline proportions (calculated with data for 105 weeks)

T refers to the transpose of the matrix

S^{-} refers to the Moore-Penrose generalized inverse of the S matrix

Proportions were normalized as follows:

Proportion = bin freq ÷ (total for all bins x 100).   100 = maximum pair-wise distance

Detection Strategies

Outbreak detection systems must decide whether an outbreak has occurred in a relevant dataset. To determine whether there is cause to alert public health officials, several detection strategies can be employed. Each of these detection strategies accounts for a specific segment of the total domain of outbreak information that is derived from the M statistic test, including the spatial clustering 'm' value and the total number of patients, the 'n' value. It appears that the n*m product value is also a useful metric for detection of patient clustering, as it accounts for so-called spatio-temporal clustering, which uses both knowledge of the abnormality of spatial clustering and temporal aberrations. As a product, the value also increases the standard deviation of the distribution of possible achievable result values, thus potentially increasing the specificity of the test. Below is a table which describes the rules that are associated with each of the four detection strategies and section of the n, m domain they each occupy.

Table 3: M-Statistic Detection Strategies

Strategy	Description
N > 95th percentile, by season	Number of visits is too high, separate values for each season
M > 95th percentile, by season	M statistic is too high, separate values for each season
MN > 95th percentile	Calculate M × N, value is too high
N and MN rules	N is too high (top 0.5% distribution) OR M × N is too high (top 0.5% distribution) OR both N is high (>80%) and M × N is high (>80%)

These detection strategies were chosen as possible cutoffs because they provide an observed significance level, or p-value, of 0.05. The first three tests, N > 95th percentile, M > 95th percentile, and the product test, MN > 95th percentile, all have p-values of 0.05. These percentile cutoff values are determined by the specific statistical sampling that is done as the m-statistic baseline values are computed. [13] The composite test, which uses three possible rules, also represents a p-value of 0.05. Instead of simply being the tail-end of a normal distribution, these rule sets represent three separate piece-wise segments of the n, m domain that all sum to the a p-value of 0.05. These detection strategies are described here because they each represent the same observed significance level, however one detection strategy, the product MN > 95th percentile strategy, has a higher observed detection rate, so it will likely be used to a greater extent in practice. Here, we will only use the m-statistic detection rates of semi-synthetic datasets to validate the m-statistic's detection of those datasets, rather than to rigorously validate the m-statistic. A validation of the m-statistic itself should include a complete ROC curve, which describes the tradeoff between sensitivity and specificity of a specific test. If this is not included in a validation of a test, it would allow for a test that always detects a cluster in a semi-synthetic dataset, but also often detects a cluster in a dataset without one. However, this validation and ROC curve is out of the scope of this work, so we use the simpler detection rate analysis as an illustration of effective semi-synthetic dataset detection.

Test Clusters Used for M-Statistic Validation

A set of 252 synthetic test clusters were created to determine the detection rate for the M-Statistic test. Those synthetic clusters were combined with authentic emergency department data from Children's Hospital Boston (CHB). Each of the cluster datasets was combined with each of 204 weeks of CHB ED patient data to create a total of 51,408 semi-synthetic datasets for M-Statistic validation. The test sets contained a range of each of the possible cluster generator parameters, enumerated in Table 3, below.

Table 4: Test Cluster Parameter Values

Cluster Parameter	Values
Distance from Reference Point	5, 15, 50 miles
Radius of cluster	250, 500, 1000, 3000 meters
Cluster Magnitude	10, 25, 40 patients
Cluster Duration	1 day
Angle from Reference Point	36, 72, 108, 144, 180, 216, 252, 288, 324, 360 deg.

M-Statistic Detection Rates by Strategy

Seasonally, using 12,852 (one fourth of the 51,408 total) test clusters, each detection strategy was used and overall detection rates were observed as listed in Table 5, below. These results represent the percentage of test clusters that were detected using each of the strategies for each season of data. Some of these detection rates seem quite low, but that is because one third of the test clusters only added ten patients to the entire week of test data. This small of a perturbation to the overall weekly datasets did not appear to be uncovered well by the N detection strategy (which depends entirely on the number of visits over the week. There is a significant increase in the detection rates of spring and summer clusters using the N-based detection strategy. This is likely due to the strong seasonal decrease in hospital emergency room admissions during the spring and summer. The increase in seasonal admissions during the winter and fall decrease the ability to detect clusters purely on the number of emergency department admissions alone. This is due to the overall number of patients and the variability (and standard deviation) of those admission rates during the winter and fall being significantly higher than those in the spring and summer.

Table 5: M-Statistic Detection Rates for Various Strategies, by season

Cluster Detection Strategy	Percentage of test clusters that triggered alarms				
	All Seasons	Winter	Spring	Summer	Fall
N > 95th percentile	16.24	11.40	21.67	19.66	11.97
M > 95th percentile	49.13	43.61	49.42	55.35	48.01
M*N > 95th percentile	62.32	55.43	63.49	70.90	59.27
N and M*N Composite	55.83	66.60	49.61	55.01	52.52

The purely spatial clustering detection strategy (M-based detection) also varies seasonally. This is likely due to the nature of the type of detection strategy involved – while spatial clustering should be no more prevalent in the absence of actual outbreaks during any one season than another, it is likely that any cluster at all will stand out more when there are fewer patients. This is the case during the summer, when there are sometimes fewer than half the number of patients (on average by week number) than there are during some winter weeks. There is, however, a statistically significant decrease in the purely spatial detection strategy during the winter season. One hypothesis as to why this might be true is that seasonally, during winter, there were significant influenza outbreaks in the test locations in each winter dataset that was used. This type of outbreak would make the overall baseline clustering closeness threshold higher for the winter than in other seasons, because a contagious (and likely closely-clustered) outbreak adds many more small inter-point distances to the baseline M-statistic inter-point distance distribution.

Another potentially interesting finding is that the N and M*N composite-based test serves as the best detection strategy during the winter season, and also, as a strategy, has its highest detection rates during the winter. One potential reason for this is that the N and M*N composite detection strategy has as one option the rule ‘both N is high (>80 percentile)’ and ‘M × N is high (>80 percentile)’, which appears to be a useful indicator in cases where there is neither a significantly high N nor a significantly high M*N value. With an N-based or M*N-based strategy percentile value of higher lower than 95, neither of those detection strategies will pick up potential clusters that have questionable numbers of patients and questionable spatial clustering. Both of these together provide some motive for believing that there is a cluster present in a given dataset, and this is likely the case during the winter, when both N and M*N values are high, but not high enough to be detected using the other three alarm strategies.

Another controlled feature set component that can be used for separate analysis of the test cluster datasets is the size of the cluster in number of patients. The artificial test datasets add either 10, 25, or 40 patients to each of the weeks of patient data. Table 6 contains the detection rates for each of the three cluster sizes, for each detection strategy.

Table 6: Sensitivity to detect simulated clusters of three sizes

		Percentage of test clusters that triggered alarms				
Cluster detection strategy	# extra visits	All seasons	Winter	Spring	Summer	Fall
M > 95 %ile, by season	10	8.30	6.30	7.81	11.73	7.32
	25	57.41	47.09	60.34	65.52	56.36
	40	81.69	77.44	80.10	88.81	80.34
MN > 95 %ile	10	20.93	13.87	21.43	32.69	15.52
	25	74.69	65.35	76.61	84.70	71.82
	40	91.35	87.06	92.42	95.33	90.46
N and MN rules	10	14.86	35.22	6.43	9.56	8.97
	25	65.12	74.67	57.68	67.17	61.38
	40	87.50	89.91	84.71	88.30	87.21

In all cluster detection strategies, larger clusters are more easily detected, which should be expected, because larger clusters provide larger changes from the expected distributions of patients. The ability of M-Statistic detection using small sized clusters and only spatial data was very low during the winter, but detection was quite successful during the summer using only spatial data. This is likely due to the fact that there are a greater number of patients in the winter, so additional patients with no aberrant spatial properties will be less easy to locate. Another noticeable difference is that the spatial only (M-based detection) detection strategy has significantly lower detection rates than the other two detection strategies. This is likely due to the fact that the other two detection strategies incorporate temporal data much more significantly, so an increase in the number of patients added should increase the chance that temporal detection values would be higher than normal. This is confirmed by the increase of ten percent or more, seasonally. Also worth noting is that in all strategies and in all seasons, the M-Statistic algorithm detected artificially added clusters of size 25 in over 75% of cases.

Artificial clusters were added to authentic patient data at three different distances to the Children's Hospital Boston centerpoint: 5, 15 and 50 km. The detection rates for each of the M-Statistic strategies in each season are listed below in Table 7. Increases in detection rates were observed as the clusters were added farther away from the hospital. This is likely due to the fact that the areas farther away from the hospital are less densely-populated, so an increase in the number of patients there is more significant and visible to the system than is an increase in the number of patients in a more densely-populated neighborhood.

Table 7: Sensitivity to detect simulated clusters at three distances from the hospital

		Percentage of test clusters that triggered alarms				
Cluster detection strategy	Km from hospital	All seasons	Winter	Spring	Summer	Fall
M > 95 %ile, by season	5	33.20	27.68	30.19	43.76	31.11
	15	57.65	55.64	56.76	65.35	52.84
	50	63.38	54.35	69.55	61.90	67.31
MN > 95 %ile	5	49.33	41.82	50.58	60.26	44.42
	15	69.81	62.78	71.01	76.89	68.35
	50	73.41	67.51	74.40	80.13	71.40
N and MN rules	5	42.44	56.21	34.10	40.92	39.08
	15	62.87	72.02	57.47	62.91	59.43
	50	67.92	76.03	63.90	67.25	64.80

One final controlled feature set component was the radius of the cluster; the radius defines the size of the space into which the same number of points are placed. This is then indirectly a measure of cluster density, which should have an appreciable effect on detection using the M-Statistic's spatial strategies, because the spatial strategies measure the distribution of inter-patient distances. Decreasing the radius should increase the number of patients that are close together in this distribution, so it should be expected that smaller radius clusters should likely have higher detection rates. An increasing trend as the radius of the cluster is decreased (with a single detection strategy and season,) but the detection rate appears to plateau below a radius of size 500 m. This is somewhat unexpected, because the patients in a cluster with a 250 m radius are four times as densely packed as the patients in a cluster with a radius of 500 m. One possible explanation for this observation is that the M-Statistic uses only ten bins to discretize the overall inter-point distance distribution in a given set of patients, with distances ranging from two patients in the same place (zero miles apart) to patients up to 100 miles apart. Because there are only ten bins into which to store the overall distribution, there may be a significant loss of small-scale patient visit information using this technique. Even though the patients in the smaller clusters are four times as densely packed, they appear almost as densely packed to the M-Statistic algorithm, because most of those patients probably appear in the smallest size inter-point distance bin already. One example of bin cutoffs, for Children's Hospital Boston, GI, fall m-statistic:

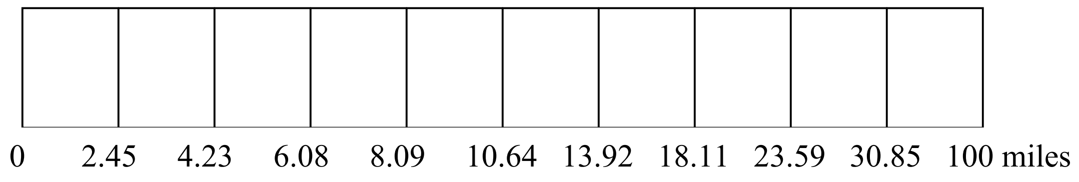


Table 8: Sensitivity to detect simulated clusters with four radius sizes

		Percent time alarm was triggered				
Cluster detection strategy	Radius in km	All seasons	Winter	Spring	Summer	Fall
M > 95 %ile, by season	250 m	54.75	48.10	55.56	61.47	53.67
	500 m	53.82	47.48	54.24	60.58	52.81
	1 km	51.73	45.72	51.81	59.05	50.18
	3 km	36.24	33.15	36.08	40.31	35.36

MN > 95 %ile	250 m	66.97	59.54	68.82	75.04	64.25
	500 m	66.41	59.10	67.67	74.96	63.68
	1 km	64.71	57.75	65.94	73.11	61.82
	3 km	51.21	45.32	51.53	60.51	47.33
N and MN rules	250 m	60.67	69.52	55.00	61.18	57.34
	500 m	59.93	69.12	54.10	60.33	56.55
	1 km	58.28	67.91	52.47	58.33	54.81
	3 km	44.44	59.87	36.88	40.21	41.38

Chapter 5: Discussion: Present Results and Future Development

Use of the Cluster Creation Tool

The current goal for use of the cluster creation tool is to evaluate multiple spatial detection algorithms using semi-synthetic datasets containing clusters that vary over all reasonable parameters and values. In order to make the testing of this algorithm realistic, the clusters that are created by the tool must include the most realistic possible situations for actual patient cluster creation.

Pertinent Syndromic Surveillance Future Objectives

A good deal of work has been done in the field of epidemiology to determine what the most pertinent and realistic patient models of syndromic spread are. Once the most realistic scenarios are assessed and mathematically described, they can be programmatically implemented and incorporated into the cluster creation tool. Several of the most promising scenarios are described below and will be likely implemented as improvements to the current tool.

Modeling the Super-Spreader Phenomenon

Another valuable model to create in the cluster tool would be to create distributions of multiple sub groups within a cluster group that each grow in time, following the so-called "super-spreader" phenomenon. In this example, one patient begins to spread (in time and space) a disease, which is then spread farther by super-spreaders, who each create their own geospatial and temporal patient cluster distributions. Those resulting distributions likely follow similar time courses and spatial growth patterns. This super-spreader phenomenon has been observed with several diseases and would be expected in the case of a syndrome with an incubation time or another set of realistic conditions that lead to this type of spread.

Nearest Neighbor Mapping

At present, the cluster generator produces patient addresses in unrealistic locations such as rivers and oceans. To address this problem, it is possible to map each point to its nearest neighbor in a database of patient addresses or in a database containing all physical addresses. One issue with nearest neighbor mapping is that some locations will end up heavily populated with inappropriate cluster points. For example, if a cluster is created in the ocean, all the cluster points will map to addresses along the coast line. This is an unrealistic model set of points because the user was attempting to create a single, circular cluster, and may receive a bimodal or oddly-shaped cluster as output. This problem may be alleviated by allowing no two points to map to the same point, or by deleting points which do not map closely to a physical address. More work needs to be done to determine if either of these are accurate descriptions of how a disease might spread.

Relating cluster density proportionally with population density

Another pertinent question for future development is whether artificial cluster density should match location population density proportionally. The benefits of this would be added realism in the test sets of clusters, because the number of patients that appears in an outbreak is very likely related to the number of people per unit of area in a certain location. Census data is publicly available at the census block group level that describes the population density for relatively small areas. This data could be harnessed to allow a user to specify a number of patients to add to a cluster per unit of population density. Using a metric such as number of patients to add to the cluster per population per square mile would allow clusters to be created with a dynamic number of patients that would probably be more appropriate and interpretable than just a user-specified number of people.

Cluster Location Distributions

For specific types of outbreaks, different geographic patient distributions have been observed. Similar to the addressed issues of time-evolution modeling of artificial patients in the cluster, the physical distribution of the patients in a cluster may be very valuable during analysis, because of the additional information that it may provide algorithms.

This additional analysis information could be harnessed by a tool that might interpret geographic cluster distributions and give some sort of parameter-set or value back to describe the orientation and/or distribution of the points. With this basic analysis of the distribution, specific algorithms could be chosen for analysis and there would also be an additional set of clues as to the possible sorts of illnesses that might be involved.

There are also considerations related to the types of geospatial distributions. It will be valuable to research how those clusters could be situated in real-life examples. Geographic possibilities are Gaussian, Linear, Exponential, teardrop, among others.

Limitations of Cluster Generator

An advantage of controlled feature set simulations is that any set of parameters can be chosen to mimic a known or theoretical outbreak. However, the use of any simulated data for benchmarking syndromic surveillance systems carries the risk of evaluating performance under unrealistic conditions. The controlled feature set simulation approach entails the explicit assumption that the historical data are pure noise and contain no signal. For bioterrorism-related events, this assumption is almost certainly true. However, it is quite possible, and even likely, that detectable outbreaks of naturally occurring infection are contained within the historical data.

Another limitation is that this approach does not account for processes occurring at the syndromic grouping stage, because artificial cases are injected directly into the data stream. When a case of true upper respiratory infection presents to the ED, it may or may not be correctly assigned to the proper syndromic group based on a chief complaint or ICD code. The approach could be modified, however, to introduce simulated cases earlier in the process, hypothetically presenting them to the syndromic classifier, enabling modeling of the accuracy of the syndromic grouping process. Also, in live syndromic surveillance systems, records representing specific events for a given day may be transmitted from the data sources at different points in time. Such time delays could be incorporated into the controlled feature set simulations. In the experiments described, several discrete parameter values are assigned. Another approach would be to use a method such as Monte Carlo simulation to redefine the model parameters over a smoother distribution of values. Importantly, the application of controlled feature set simulation to surveillance using multivariate data streams requires explicit assumptions about the relationships among the signal features across data sets.

Conclusion

The use of semi-synthetic datasets containing authentic background noise and outbreaks defined by a controlled feature set provides a valuable means for benchmarking the detection performance of syndromic surveillance systems. A cluster generator was implemented with the ability to quickly create semi-synthetic test clusters for use in evaluating detection rates of spatial detection algorithms under a variety of controlled feature set conditions.

Appendix A: Real-time M-Statistic Documentation

Real-Time M-Statistic Patient Spatial-Clustering Detection System

Calculates the real-time M-Statistic values for a specific hospital, syndrome, and date span against a baseline computed separately. An implimentation of the M-Statistic spatial cluster detection approach.

Libraries Imported

java.sql.*;

java.lang.Math.*;

java.lang.Object.*;

java.util.*;

java.io.*;

Jama.*;

The first five imported libraries are Sun Java(TM) 1.4.2 libraries. Jama. is the numerics and matrices Java Class Library that helps store and manipulate M-Stat matrices, and is publicly available from http://math.nist.gov/javanumerics/jama/ , created by The MathWorks and the NIST.*

M-Stat API-Level Methods

public static void executeCompositeMstatForRegion (int region_id, int int_num_prev_days)

This method computes the composite m-statistic for a region (which is comprised of a group of hospitals with the same region_id field value.) This uses all patients from all of those hospitals that meet the basic inclusion requirements for the computation (same syndrome, same date span.)

This method does not allow for customization of the m-stat query and only takes the region_id and the number of days to compute each syndrome's mstat.
Outbreaks are entered into the outbreaks table of the database as appropriate.

public static void executeAllMstatsForRegion(int region_id, int num_prev_days)
This method independently executes all of the relevant mstat calculations for a region, but ***does not combine the patients from different hospitals into one single mstat calculation*** *for each syndrome. Instead, patients with each syndrome, from each hospital are run independently of one another.*
Outbreaks are entered into the outbreaks table of the database as appropriate.

public static void executeAllMstatsForSpecHosp (int num_prev_days ,int hosp_id)
This method calculates the mstat values for each syndrome from a single hospital for the past n days.
Outbreaks are entered into the outbreaks table of the database as appropriate.

public static void executeAllMstatsGeneralNDays (int num_prev_days)
This method independently executes all of the relevant mstat calculations for a region, but ***does not combine the patients in the calculation****. Patients with each syndrome from each hospital are run independently of one another.*
Outbreaks are entered into the outbreaks table of the database as appropriate.

public static Vector getSyndromeIdVector() throws ClassNotFoundException, SQLException
This method returns a vector of Integer objects that contains the active syndromes in the database relation syndrome_info. Active syndromes are determined by inspection of the active field, which has a value of '1' for active syndromes.

public static Vector getHospitalIdVector() throws ClassNotFoundException, SQLException

This method returns a hospital id vector which is a vector of Integer objects that contains only hospitals with an active field = 1 in the hospitals table.

public static Vector getHospitalIdVectorInRegion(int region_id) throws ClassNotFoundException, SQLException

This method returns a hospital id vector which is a vector of Integer objects; this only returns the active hospitals in a specific region. Active is described in the hospitals relation of the database with active = 1.

public static void executeMstatLastNDaysSingle (int num_prev_days, String season, int syndrome_id , int hospital_id)

This is a helper method that simply handles getting the Timestamps that are relevant for the date span that will be part of the actual Mstat calculation.

public static void executeMstatSpecific (String season, int syndrome_id, int hospital_id, Timestamp start_date, Timestamp end_date)

This method is the most specific execution method for the mstatistic. It allows for a specific start and end date, specific hospital_id, syndrome_id, and season, and automatically retrieves the appropriate values, calculates the specific m-statistic value, compares that calculated value against the baseline value in the database, and then writes an outbreak into the outbreaks table of the database as appropriate.

public static void writeOutbreaksToDatabase(Vector outbreakVector) throws ClassNotFoundException, SQLException

This method takes a Vector of Outbreak objects and writes it to the outbreaks table. Outbreak objects follow the descriptions in the Outbreak class section of the documentation, and specify whether the calculated mstat value was over the expected threshold by some specified amount.

public static Timestamp getCurrentTimestamp()

This method returns the current time as a java.sql.Timestamp. It is a helper used to calculate specific date spans for methods that require timestamps.

public static Timestamp getTimestampNDaysBefore(Timestamp currentTimestamp, int daysBefore)

This method takes a java.sql.Timestamp and returns a java.sql.Timestamp that corresponds to the date/time of the current time less n days before. This is a helper method used to calculate specific date spans for methods that require them, as well.

public static String findCurrentSeason()

This helper method returns a String that is either "winter", "spring", "summer", or "fall", depending on the appropriate season. This is a potential change in v.2.0 to ints with findCurrentSeasonInt().

public static Vector requestLongLatsFromDatabaseSpecific(Timestamp start_date, Timestamp end_date, int syndrome_id, int hospital_id) throws ClassNotFoundException, SQLException

Helper method to retrieve a Vector of LongLat Objects for a specific date span, syndrome_id, and hospital_id. This method must be called from a try/catch loop.

public static Vector requestVisitsFromDatabaseForHosp(Timestamp start_date, Timestamp end_date, int hospital_id) throws ClassNotFoundException, SQLException

This method returns the Visit objects that meet the criteria from a specific hospital and date span. There is a Visit class described separately. This method retrieves Visit objects corresponding to patients with all active syndromes.

public static Vector requestVisitsFromDatabaseForRegion(Timestamp start_date, Timestamp end_date, int region_id) throws ClassNotFoundException, SQLException

This method returns the Visit objects that meet the criteria from all hospitals in a specific region and date span. There is a Visit class described separately. This method retrieves Visit objects corresponding to patients with all active syndromes.

public static Matrix calcObsMinusExpMatrix(Matrix binCountsMatrix, int vectorSize)

This method is used by the calculate Mstatistic methods to actually calculate the observed minus expected value vector described in the m-statistic literature.

public static Matrix sortDistancesIntoBins(Matrix binMatrix, Vector distanceColumnVectors)

This helper method sorts the interpoint distances into the ten value-based bins, as described in the m-statistic literature.

Appendix B: Availability and Program Requirements

- Project name: AEGIS Cluster Creation Tool
- Project home page: http://sourceforge.net/chipcluster/
- Operating system(s): Platform independent
- Programming language: Java
- Other requirements: Java 1.3.1 or higher
- License: e.g. GNU LGPL
- Any restrictions to use by non-academics: none

List of abbreviations

- ED – Emergency Department
- GIS – Geographical Information Systems
- ICD-9 – International Classification of Diseases, 9th Revision
- GUI – Graphical User Interface
- CSV – Comma-Separated Values

Acknowledgements

This work was supported by the National Institutes of Health through R01LM07677-01 and R01LM007970-01 from the National Library of Medicine and by grant 2002-12-1 from the Alfred P. Sloan Foundation.

Karin Iancu helped to implement the graphical user interface as well as the input/output and text formatting in the cluster generator program. Karen Olson assisted by providing real hospital datasets and knowledge of the pertinent issues in the outbreak detection arena. Karen Olson also provided detection rate tables in Chapter 4 related to her study of statistical M-Statistic analysis. Kenneth Mandl conceived of parts of the study, and participated in its design and coordination.

References:

1. Center for Biopreparedness at Children's Hospital Boston; Automated Epidemiologic Geotemporal Integrated Surveillance (AEGIS). Available at http://www.chip.org/biosurv/index.htm. Accessed August 30, 2004.
2. Syndromic Surveillance: A Population-Adjusted, Stable Geospatial Baseline for Outbreak Detection; Karen L. Olson PhD, Marco Bonetti PhD, Marcello Pagano PhD, Kenneth D. Mandl, MD MPH.
3. "Virtues of the Haversine," Sky and Telescope, R. W. Sinnott, vol. 68, no. 2, 1984, p. 159
4. Sosin D. Framework for evaluating public health surveillance systems for early detection of outbreaks. Available at: http://www.cdc.gov/epo/dphsi/syndromic/framework.htm. Accessed December 4, 2003.
5. Meselson M, Guillemin J, Hugh-Jones M, et al. The Sverdlovsk anthrax outbreak of 1979. *Science.* 1994;266(5188):1202-1208.
6. Jernigan JA, Stephens DS, Ashford DA, et al. Bioterrorism-related inhalational anthrax: the first 10 cases reported in the United States. *Emerging Infectious Diseases.* 2001;7(6):933-944.
7. Mandl KD, Overhage JM, Wagner MM, et al. Implementing syndromic surveillance: a practical guide informed by the early experience. *J Am Med Inform Assoc.* 2004;11(2):141-150 [PrePrint published Nov 121, 2003; as doi:2010.1197/jamia.M1356].
8. Tsui F-C, Espino JU, Dato VM, Gesteland PH, Hutman J, Wagner MM. Technical description of RODS: a real-time public health surveillance system. *J Am Med Inform Assoc.* June 4, 2003 2003;10(5):399-408.
9. DoD-GEIS. "Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE). Available at: http://www.geis.ha.osd.mil/GEIS/SurveillanceActivities/ESSENCE/ESSENCEinstructions.asp. Accessed October 28, 2003.

10. Reis BY, Mandl KD. Time series modeling for syndromic surveillance. *BMC Med Inform Decis Mak.* 2003;3(1):http://www.biomedcentral.com/1472-6947/1473/1472.

11. Kulldorff M, Athas WF, Feurer EJ, Miller BA, Key CR. Evaluating cluster alarms: a space-time scan statistic and brain cancer in Los Alamos, New Mexico. *Am J Public Health.* Sep 1998;88(9):1377-1380.

12. Burkom HS. Biosurveillance applying scan statistics with multiple, disparate data sources. *J Urban Health.* Jun 2003;80(2 Suppl 1):i57-65.

13. Bonetti M, Olson KL, Mandl KD, Pagano M. Parametric models for interpoint distances and their use in biosurveillance. *Proceedings of the American Statistical Association, Biometrics Section [CDROM].* 2003:pp-pp.

14. Lazarus R, Kleinman K, Dashevsky I, et al. Use of automated ambulatory-care encounter records for detection of acute illness clusters, including potential bioterrorism events. *Emerging Infectious Diseases.* 2002;8(8):753-760.

15. Hogan WR, Tsui FC, Ivanov O, et al. Detection of pediatric respiratory and diarrheal outbreaks from sales of over-the-counter electrolyte products. *J Am Med Inform Assoc.* Nov-Dec 2003;10(6):555-562.

16. Goldenberg A, Shmueli G, Caruana RA, Fienberg SE. Early statistical detection of anthrax outbreaks by tracking over-the-counter medication sales. *Proc Natl Acad Sci U S A.* Apr 16 2002;99(8):5237-5240.

17. Update: influenza activity--United States, January 18-24, 2004. *MMWR Morb Mortal Wkly Rep.* Jan 30 2004;53(3):63-65.

18. Wein LM, Craft DL, Kaplan EH. Emergency response to an anthrax attack. *Proc Natl Acad Sci U S A.* Apr 1 2003;100(7):4346-4351.

19. Reis BY, Pagano M, Mandl KD. Using temporal context to improve biosurveillance. *Proc Natl Acad Sci U S A.* 2003;100(4):1961-1965.

20. Begier E, Sockwell D, Branch L, et al. The National Capitol Region's emergency department syndromic surveillance system: do chief complaint and discharge diagnosis yield different results? *Emerg Infect Dis.* 2003;9(3):393-396.

21. Beitel AJ, Olson KL, Reis BY, Mandl KD. Use of emergency department chief complaint and diagnostic codes for identifying respiratory illness in a pediatric population. *Pediatric Emergency Care.* [In Press].

22. Espino JU, Wagner MM. Accuracy of ICD-9-coded chief complaints and diagnoses for the detection of acute respiratory illness. *Proc AMIA Symp.* 2001:164-168.

23. Mocny M, Cochrane DG, Allegra JR, et al. A Comparison of Two Methods for Biosurveillance of Respiratory Disease in the Emergency Department: Chief Complaint vs ICD9 Diagnosis Code. *Acad Emerg Med.* 2003;10(5):513.

24. Reis BY, Mandl KD. Syndromic surveillance: the effects of syndrome grouping on outbreak detection performance. *Annals of Emergency Medicine.* [In Press].

25. Reis BY, Mandl KD. Integrating syndromic surveillance data across multiple locations: effects on outbreak detection performance. *Proc AMIA Symp.* 2003:549-553.