2007-06

Biological terrorism preparedness evaluating the performance of the Early Aberration Reporting System (EARS) syndromic surveillance algorithms

Hegler, Benjamin L.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

BIOLOGICAL TERRORISM PREPAREDNESS: EVALUATING THE PERFORMANCE OF THE EARLY ABERRATION REPORTING SYSTEM (EARS) SYNDROMIC SURVEILLANCE ALGORITHMS

by

David A. Dunfee
Benjamin L. Hegler

June 2007

Thesis Advisor:	Ronald D. Fricker, Jr.
Second Reader:	David H. Olwell

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2007	Master's Thesis

4. TITLE AND SUBTITLE Biological Terrorism Preparedness: Evaluating the Performance of the Early Aberration Reporting System (EARS) Syndromic Surveillance Algorithms	5. FUNDING NUMBERS
6. AUTHOR(S) David Dunfee, Benjamin Hegler	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	A

13. ABSTRACT (maximum 200 words)

After the terrorist attacks of September 11, 2001, questions developed over how quickly the country could respond if a bioterrorism attack was to occur. "Syndromic surveillance" systems are a relatively new concept that is being implemented and used by public health practitioners to attempt to detect a bioterrorism attack earlier than would be possible using conventional biosurveillance methods. The idea behind using syndromic surveillance is to detect a bioterrorist attack by monitoring potential leading indicators of an outbreak such as absenteeism from work or school, over-the-counter drug sales, or emergency room counts. The Center for Disease Control and Prevention's Early Aberration Reporting System (EARS) is one syndromic surveillance system that is currently in operation around the United States.

This thesis compares the performance of three syndromic surveillance detection algorithms, entitled C1, C2, and C3, that are implemented in EARS, versus the CUSUM applied to model-based prediction errors. The CUSUM performed significantly better than the EARS' methods across all of the scenarios evaluated. These scenarios consisted of various combinations of large and small background disease incidence rates, seasonal cycles from large to small (as well as no cycle), daily effects, and various levels of random daily variation. This results in the recommendation to replace the C1, C2, and C3 methods in existing syndromic surveillance systems with an appropriately implemented CUSUM method.

14. SUBJECT TERMS Syndromic Surveillance, Biosurveillance, Bioterrorism, Public Health, Early Event Detection, C1, C2, C3, Cumulative Sum (CUSUM)	15. NUMBER OF PAGES 145
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

BIOLOGICAL TERRORISM PREPAREDNESS: EVALUATING THE PERFORMANCE OF THE EARLY ABERRATION REPORTING SYSTEM (EARS) SYNDROMIC SURVEILLANCE ALGORITHMS

David A. Dunfee
Ensign, United States Navy
B.S., United States Naval Academy, 2006

Benjamin L. Hegler
Ensign, United States Navy
B.S., Auburn University, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED SCIENCE (OPERATIONS RESEARCH)

from the

NAVAL POSTGRADUATE SCHOOL
June 2007

Authors: David A. Dunfee

Benjamin L. Hegler

Approved by: Dr. Ronald D. Fricker, Jr.
Thesis Advisor

Dr. David H. Olwell
Second Reader

Dr. James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

After the terrorist attacks of September 11, 2001, questions developed over how quickly the country could respond if a bioterrorism attack was to occur. "Syndromic surveillance" systems are a relatively new concept that is being implemented and used by public health practitioners to attempt to detect a bioterrorism attack earlier than would be possible using conventional biosurveillance methods. The idea behind using syndromic surveillance is to detect a bioterrorist attack by monitoring potential leading indicators of an outbreak such as absenteeism from work or school, over-the-counter drug sales, or emergency room counts. The Center for Disease Control and Prevention's Early Aberration Reporting System (EARS) is one syndromic surveillance system that is currently in operation around the United States.

This thesis compares the performance of three syndromic surveillance detection algorithms, entitled C1, C2, and C3, that are implemented in EARS, versus the CUSUM applied to model-based prediction errors. The CUSUM performed significantly better than the EARS' methods across all of the scenarios evaluated. These scenarios consisted of various combinations of large and small background disease incidence rates, seasonal cycles from large to small (as well as no cycle), daily effects, and various levels of random daily variation. This results in the recommendation to replace the C1, C2, and C3 methods in existing syndromic surveillance systems with an appropriately implemented CUSUM method.

THIS PAGE INTENTIONALLY LEFT BLANK

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Andy Dunfee:

First, I would like to thank Professor Fricker for his guidance and mentorship during the entire thesis process. You were a joy to work with and made completing a thesis an overall enjoyable experience. Thanks for all the time and work you put into assisting Ben and I. I would also like to thank my partner, Ben. You were a pleasure to work and a good friend during the course of this past year. Finally, I would like to thank my family and friends for being so supportive of me during my time here at Naval Postgraduate School.

Benjamin Hegler:

First, a great big "Thank you!" to Andy Dunfee for his hard work, consistency, and friendship throughout the year. It was a pleasure to work with you, Andy. Also, of course, many thanks to Dr. Ronald Fricker for the devotion and foresight he displayed toward Andy and I throughout the year with regard to our thesis research. Thanks for keeping us on track, Dr. Fricker. It was a pleasure working with you.

Thanks also to my family for being there for me and supporting me in whatever I do. You guys rock.

Lastly, I would like to thank the Operations Research department faculty at the Naval Postgraduate School for their dedication to their students' education and well-being. I have learned much more than I realize this year. Thank you all.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

After the terrorist attacks of September 11, 2001, many questions arose over how quickly the country could respond if a bioterrorism attack were to occur. Using syndromic surveillance for early event detection of a bioterrorism attack and situational awareness during the course of the attack is a relatively new concept that is becoming used in the public health world. The idea behind using syndromic surveillance is to detect a bioterrorist attack by monitoring potential leading indicators of an outbreak such as absenteeism from work or school, over-the-counter drug sales, or the number of people presenting at an emergency room that exhibit a specific chief complaint. Three such syndromic surveillance algorithms which have been used by the Center for Disease Control and Prevention's (CDC) program entitled Early Aberration Reporting System (EARS) are the C1, C2, and C3 methods.

Currently almost no published performance analysis comparisons between the EARS methods and any other methods exist. This research evaluates the performance of the EARS methods versus a cumulative sum (CUSUM) method applied to forecast errors of an "adaptive regression with sliding baseline" model. The adaptive regression with sliding baseline was used to predict the current day's observation, and this prediction was compared to the actual count. Counts that exceed the predictions are evidence of a possible bioterrorist attack or a natural disease outbreak. Three adaptive regression models were fit, each using a sliding baseline based on a different amount of historical data: the previous eight weeks; the past seven days (corresponding to what is used in the C1, C2, and C3 methods); and the "optimal" amount of historical data that minimizes the forecast errors under each of the background disease incidence scenarios.

The methods were compared using synthetic syndromic surveillance data, simulated in such a way as to exhibit the major characteristics of syndromic surveillance data. Examples of these characteristics are: large and small baseline disease incidence; small, large, and no seasonal cycles; day-of-the-week effects; and small and large daily random variations. The large baseline disease incidence daily variation was generated by a random normal distribution, and the small baseline disease incidence daily random

variation was generated by a random lognormal distribution. Each daily observation generated was rounded up to the nearest whole number in order to make it a realistic count. The reason for using simulated data was to be able to compare the methods' relative performance under known and controlled conditions.

The analysis was conducted on 10 cases that mimicked different types of background disease incidence behavior. Six cases were examined using a large baseline mean disease incidence of 90, which involved three seasonal cycles (none, small, large), and two daily variations (small and large). Two cases were examined using a small baseline disease incidence of 0, which involved two daily variations (small and large). Finally, two cases were examined with day-of-the-week effects included: one large baseline disease incidence and one small baseline disease incidence. Each algorithm (C1, C2, C3, and the CUSUMs with the three sliding baselines) was evaluated for each case for various sizes of disease outbreaks. An outbreak was defined as a linear increase up to some day "X", followed by an equal linear decrease back to the normal incidence rate. Outbreak durations from three to fifteen days were evaluated. Three outbreak magnitudes were evaluated for each large count case: small, medium, and large. Four different outbreak sizes were evaluated for each small count case: small, medium, large, and extra-large, with the outbreak magnitudes for each size differing between the cases when daily variation was small and when it was large.

Two comparison metrics were used to analyze each algorithm's performance under each set of conditions. The first metric was the average time to first signal (ATFS) given a true signal; the goal for this measurement was to measure how quickly each method signaled an alarm, given that the method detected the outbreak within its duration. In the conduct of the simulations, the signal threshold was first set such that the procedures had equal ATFS under some background (non-outbreak) disease incidence scenario. The second metric was the fraction of the outbreaks that were not detected within their duration for each duration length (fraction missed). Both comparison metrics were taken into account when evaluating the relative performance of the algorithms for certain background disease incident patterns and various outbreak magnitudes and durations.

A clear conclusion from this work is that the CUSUM with an eight week and "optimal" sliding baseline performed significantly better in all the scenarios evaluated. While the C1 and CUSUM with a seven day sliding baseline tended to have slightly shorter ATFS given a true signal, this came at the expense of missing a much greater number of outbreaks than the CUSUMs with either the "optimal" or 56 day sliding baseline. This difference in performance is evident in all outbreak magnitudes but is most evident with the larger magnitude outbreaks. The three EARS methods performed similarly across all simulations, generally with only relatively small differences in performance. Of the EARS methods, the C2 method had the lowest fraction missed on the majority of the simulations, but the C1 was typically faster than the C2 and C3 for the ATFS given a true signal. The C3 method's performance varied, but it was typically outperformed by the C1 in the ATFS given a true signal and the C2 in the fraction missed.

Overall, the CUSUM methods, particularly with the eight week and "optimal" sliding baselines, outperformed the EARS methods. Therefore, standard syndromic surveillance systems using the EARS methods would benefit from replacing the EARS methods with a CUSUM method based on adaptive regression forecast errors, setting the CUSUM thresholds in a similar fashion as done in this research in order to minimize the false alarm burden as much as possible.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

> *Syndromic surveillance* has been defined as the ongoing, systematic collection, analysis, interpretation, and application of real-time (or near-real-time) indicators of diseases and outbreaks that allow for their detection before public health authorities would otherwise note them. It has also been defined as surveillance using health-related data that precede diagnosis and signal a sufficient probability of a case or an outbreak to warrant further public health response.
>
> Fricker (2007a)

1. Bioterrorism and Syndromic Surveillance

In our world today, the threat of bioterrorism is very real, and the potential to be surprised by an attack is certainly a concern. As with any contagious disease outbreak, earlier detection allows for easier containment and fewer infections as well as, potentially, lives saved. Simply realizing that an attack or outbreak has occurred is a great leap forward in knowing what steps need to be taken in the interest of public health and safety. While a large-scale, fast-acting bioterrorism attack that sends hundreds of extra patients with similar symptoms to an emergency room would not go unnoticed for long, what if the attack were more subtle? What if the attack was slow to act and caused a very gradual increase in the number of patients? Such a scenario might potentially go unnoticed for weeks or longer. This is exactly the sort of scenario that syndromic surveillance is intended to help with.

After the terrorist attacks on September 11, 2001, many questions arose concerning how quickly the country could respond if a bioterrorism attack were to occur. Using syndromic surveillance systems to detect a bioterrorist attack is a relatively new concept that emphasizes timeliness. Essentially, syndromic surveillance uses emperical methods to attempt to sniff out sudden but relatively small changes in some normal rate of illness occurrence, hopefully leading to proper and timely diagnosis of a potentially lethal situation that might otherwise go unnoticed for quite some time. Syndromic

surveillance can monitor many indicators of an outbreak, such as absenteeism from work or school, over-the-counter drug sales, or emergency room entries that exhibit a respiratory complaint. Thus, one of the fundeamantal goals of syndromic surveillance is to gain time during which the public health system can better respond:

> The advantage of syndromic surveillance is the lead-time it provides public health authorities to take more effective public health actions. What syndromic surveillance allows is not necessarily earlier diagnosis per se but the ability to mobilize public health investigation and response capabilities before disease an out-break confirmation.
>
> Sosin (2003)

Another goal is to help manage the response to an outbreak or an attack:

> More recently, the purpose of syndromic surveillance has been expanded to include using existing health data in real time to provide immediate analysis and feedback to those charged with investigation and follow-up of potential outbreaks. This broader focus on *electronic biosurveillance* includes both early event detection and *situational awareness*. Situational awareness is the real-time analysis and display of health data to monitor the location, magnitude, and spread of an outbreak, as well as the availability and application of public health and medical resources to the outbreak.
>
> Fricker (2007a).

Although the capabilities of syndromic surveillance are becoming better understood as the result of on-going research, the current state of knowledge is limited on its effectiveness when used to detect disease outbreaks and bioterrorism attacks. Sosin (2003) analyzes the case for skillful investment using syndromic surveillance, and Shmueli (2006) presents the statistical challenges and questions that still needed to be answered in biosurveillance. Although some questions remain unanswered, improvements and studies are being conducted, and many public health officials believe that syndromic surveillance is a promising tool for detecting a disease outbreak or bioterrorism attack in a timely and efficient manner. See Fricker (2007a), Fricker and Rolka (2006), Stoto et al. (2006) and Fricker (2007b) for more detailed discussions.

Currently, there are a number of syndromic surveillance systems that use the EARS detection algorithms. These algorithms were initially developed for the EARS syndromic surveillance system (www.bt.cdc.gov/surveillance/ears/) which was "designed for monitoring for bioterrorism during large-scale events that often have little or no baseline data" (Fricker, 2007a). These methods were then incorporated into the BioSense system (www.cdc.gov/biosense). BioSense is a federally directed effort by the CDC that currently uses the EARS' C1 and C3 algorithms and the W2 algorithm (a variant of the C2 algorithm):

> Begun in 2003, BioSense is intended to be a United States-wide electronic biosurveillance system that initially used Department of Defense and Department of Veterans Affairs outpatient data along with medical laboratory test results from a nationwide commercial laboratory. In 2006, BioSense began incorporating data from civilian hospitals as well. The primary objective of BioSense is to expedite event recognition and response coordination among federal, state, and local public health and healthcare organizations.
>
> Fricker (2007a)

2. Relevant Previous Syndromic Surveillance Research

The syndromic surveillance literature documents quite a number of efforts to develop and measure the performance of various individual detection algorithms. However, it contains very few comparisons between algorithms in order to assess the relative strengths and weaknesses of the algorithms. It is as if everyone is trying to develop a new hammer, but few are comparing among the hammers to determine which are to be preferred. See, for example, Brillman (2005), Farrington et al. (1996), and Reis et al. (2003).

Examples of past research that do compare between detection methods include Fricker (2007b) and Stoto et al. (2006) who evaluated different methods' performances in the context of syndromic surveillance. In particular, they compared simultaneous univariate CUSUMs against a multivariate CUSUM. The idea of their comparison was to evaluate whether it would be more effective to use simultaneous individual CUSUMs with each applied to a different data stream – say, each type of chief complaint at each hospital – or a multivariate method that evaluates a series of data streams – say, all chief

complaints at each hospital or one type of chief complaint across multiple hospitals. It was found that under certain conditions the univariate CUSUMs outperformed the multivariate CUSUM, but under other conditions the multivariate CUSUM outperformed the univariate CUSUMs. Their findings provided researchers and practitioners with some useful information about which method should be used in which situation.

Other relevant past research includes Zhu et al. (2005) who conducted an initial evaluation of the EARS methods versus Shewhart "p-chart" and exponentially weighted moving average (EWMA) methods. They concluded that the C2 method was best for autocorrelated data. However, this is not surprising since they did not modify the application of the Shewhart and EWMA methods to account for the autocorrelation.

Currently, the EARS methods used for biosurveillance do not directly take into account trends, day-of-the-week effects, or certain other systematic behavior. Burkom, Murphy, and Shmueli (2006) discuss options to account for these trends in their paper "Automated Time Series Forecasting for Biosurveillance." They use regression and time series methods to remove this behavior by subtracting forecasts from observations to form residuals for algorithmic input. They describe and compare three forecast methods: a nonadaptive loglinear regression model using a long historical baseline, an adaptive regression model with a shorter, sliding baseline, and the Holt-Winters method for generalized exponential smoothing.

Additional prior syndromic surveillance detection algorithm research is described in Woodall (2006), Shmueli (2006), Fricker (2007a), and Fricker and Rolka (2006). Also see the abstracts and papers posted online for the journal *Advances in Disease Surveillance* (www.isdsjournal.org) and in the *Morbidity and Mortality Weekly Report* (MMWR) published by the CDC (www.cdc.gov/mmwr/).

B. COURSE OF RESEARCH

1. Research Objectives

The objective of this research was to compare the performance of the C1, C2, and C3 EARS methods to a CUSUM-based method specifically designed for the syndromic

surveillance problem. As was previously mentioned, all three of the EARS methods do not explicitly account for trends, day-of-the-week effects, or other systematic behavior. The EARS methods are compared to a CUSUM applied to forecast errors of an "adaptive regression" model as defined in Burkom et al. (2006). Three adaptive regression models were fit, each with a different amount of historical data. They are: (1) the previous eight weeks, as was done in Burkom; (2) the seven days of data corresponding to what is used in the C1, C2, and C3 methods; and, (3) the optimal length of data that minimizes the forecast errors under each of the background disease incidence scenarios. Comparisons were made under a series of different scenarios, where each scenario was designed to mimic certain behaviors in syndromic surveillance data.

2. Assumptions

The comparisons were based on simulated syndromic surveillance data, both background disease incidence and outbreaks. This simulated data was, by design and of necessity, somewhat idealized. Specifically, the simulated data was designed to exhibit the major characteristics of syndromic surveillance data in order to compare and contrast the *relative* performance of the various methods under these conditions.

The objective was to gain insight into how the major features of syndromic surveillance data (e.g., large vs. small seasonal cycles, day-of-the-week effect vs. no effect, etc.) affect the relative performance of the various methods. That is, for example, the goal is to understand which method or methods work best in data with large seasonal fluctuations with significant day-to-day variation compared to, say, data that has large seasonal fluctuations but little day-to-day variation or small seasonal fluctuations but large day-to-day variation.

In the simulated data, the background disease incidence was characterized in terms of a mean disease incidence rate that varies according to a seasonal cycle and a day-of-the-week effect. Individual daily counts were then generated as the sum of the mean disease incidence, an outbreak when appropriate, and a random deviation from the mean. This is discussed further in the next chapter.

C. THESIS OUTLINE

The thesis is organized as follows. Chapter II describes how the synthetic syndromic surveillance data used in this research was simulated. Chapter III describes the various detection algorithms which were used in this research. Chapter IV describes the methodology used to evaluate the relative performance of the various methods studied, as well as a description of how various input and threshold values were chosen. Chapter V summarizes the results of the simulations, focusing on the large count, small count, and day-of-the-week effect results.

II. SIMULATING SYNDROMIC SURVEILLANCE DATA AND SCENARIOS

A. DATA GENERATION ASSUMPTIONS AND SIMULATION

1. Syndromic Surveillance Data

Syndromic surveillance data generally contain various trends and cycles. For example, syndromes related to the flu frequently exhibit a cycle in which disease incidence increases sometime in the fall or winter corresponding to the annual flu cycle. Other types of syndromic surveillance data may exhibit other types of seasonal cycles. Syndromic surveillance data also often exhibit day-of-the-week effects corresponding to the fact that, for example, people tend to go to hospital emergency rooms differentially depending on the day of the week. Similarly, over-the-counter medication sales vary in a systematic way with day of the week (as well as holiday and seasonal cycles). For a more detailed discussion and examples and plots of actual data, see Shmueli (2006), Lotze, et al. (2006), and Burkom et al. (2006).

2. Generating Synthetic Syndromic Surveillance Data

In order to capture the main features of syndromic surveillance data, the background disease incidence was characterized in terms of a mean disease incidence rate with a systematic seasonal (sinusoidal) cycle and day-of-the-week variation. Individual daily counts were then generated as the sum of these systematic effects and a random deviation from the mean. Specifically, a daily observation X_i, was simulated as

$$X_i = \max\left(0, \lceil \beta + \alpha_i + \delta_i + Y_i(\beta) \rceil\right),\ i = 1, 2, ..., \quad (1)$$

where:

- β is the baseline disease incidence;

- α is the seasonal deviation from the baseline mean, calculated as $\alpha_i = A\left[\sin(2\pi i / 365)\right]$, where A is the amplitude (which is the maximum deviation from β) with i corresponding to October 1st;
- δ is the systematic deviation from the mean (day-of-the-week effect), where $\delta_i = \delta_{i+7}$ for all i ;
- $Y_i(\beta)$ is the random noise around the systematic component $\left(\beta + \alpha_i + \delta_i\right)$ with
    - $Y_i(\text{large } \beta) \sim N(0, \sigma_L)$ and
    - $Y_i(\text{small } \beta) \sim LN(\mu, \sigma_s)$;

and where $\lceil \bullet \rceil$ is the ceiling function, which rounds the value up to the next largest integer.

The simulated years were always 365 days long, but this is only relevant when calculating α. Extending this data generation method to account for leap years is an unnecessary complication that was not considered in this work since it would not affect the results or conclusions.

3. Discussion of Assumptions

a. Seasonal Cycles

In general, there is some semblance of an annual periodic cycle in syndromic data over the course of a year or two. For example, the mean number of hospital respiratory chief complaints is likely to be higher in the month of February compared to the month of July. This is in part due to what is commonly known as the flu season, beginning sometime in the late fall or winter and ending sometime in the late winter or spring.

However, the rises and falls of this pattern occur at different times each year and, while there is a general pattern, each flu season has a different duration,

amplitude and start time. Hence, going back years in the data in an attempt to model the seasonal cycle, if such data exists, is generally not useful for syndromic surveillance.

In spite of the fact that the annual pattern is typically unpredictable and variable, for this work an artificial sinusoid was used in order to simulate the general rises and falls in real data. Although the sinusoid is an idealized characterization of the natural periodicity of disease incidence data, as will be demonstrated in more detail in the next chapter, none of the methods evaluated are designed to, nor are they capable of, exploiting this feature of the synthetic data. Simply put, the EARS methods only use seven days of past data and so are incapable of modeling the sinusoid. Similarly, the adaptive regression uses a linear model fit to 8-weeks or less of past data and thus is also incapable of modeling the sinusoid. The result is that the perfect sinusoids of the synthetic data are no more predictable for the methods being evaluated than the real, less predictable seasonal variations.

b. Day-of-the-Week

Typically, there is a day-of-the-week effect in syndromic surveillance data. For example, when looking at hospital chief complaint data, there is a systematic, daily trend that appears in most data (c.f. Brillman, 2005, and Shmueli, 2006). This day-of-the-week effect may differ depending on the location, time of year, or other factors. Similarly, when looking at over-the-counter medication sales, the counts generally exhibit a regular day-of-the-week effect. This work included a day-of-the-week effect by including the previously mentioned parameter δ in the data generation model. The main idea in this research was not that the day-of-the-week effect be accurately simulated, due to the variable nature of this effect in real data. What was most important, however, was to demonstrate the implications of including this effect in some of the simulations – namely, that the effect can be included in the simulations, and the subsequently "removed" by the regression.

For the simulations, the day-of-the-week effect δ is the systematic deviation from the baseline (with annual periodic cycle – i.e., the sinusoid), where each day of the week had a certain δ_i value assigned to it. The δ_i values were rather

arbitrarily chosen and were defined in terms of σ, the standard deviation parameter of *Y*, for both large and small count simulations as follows: $\delta = -0.5\sigma$ on Sunday; $\delta = 0.1\sigma$ on Monday; $\delta = 0.2\sigma$ on Tuesday; $\delta = 0.3\sigma$ on Wednesday; $\delta = 0.4\sigma$ on Thursday; $\delta = 0$ on Friday; and $\delta = -0.3\sigma$ on Saturday. It should be noted that these values result in a positive bias of 0.2σ, which essentially just increases the baseline for the scenarios which included the day-of-the-week effect. This change did not affect the relative performance of the methods.

The day-of-the-week effect was not included in the majority of the simulations because it did not affect the results. Simply put, all the methods were effective at accounting for and eliminating day-of-the-week effects. When reviewing simulation results, the day-of-the-week effect can be assumed to be zero, except where explicitly stated.

c. Holiday Effects

In addition to the day-of-the-week effect, there is also a holiday effect that often appears in the data. For example, stores may be closed on certain holidays and people go to hospitals in much fewer numbers. This is an issue in real data but will be left to be addressed in later work; in the current work, the methods utilized can be naturally extended to account for holiday effects.

d. Long Term Trends

The issue of long term trends also exists in hospital admittance data where, over time, the number of people presenting gradually rises or falls, perhaps in conjunction with changes in the surrounding popuation. Since the focus of this work is on a sudden, relatively small shift in the mean, long term trends that span back many months were not included in the work. Furthermore, methods that are effective at modeling the seasonal sinusoid of the synthetic data will also be able to effectively model a long term linear trend. Hence, including such a long term trend in the data generation model would have been an unnecessary complication.

e. Daily Random Variability

As with most processes, daily random variability occurs in syndromic surveillance data. For example, if the mean number of chief complaints in a hospital during the month of August over the last few years is fifty and, having chief complaints between, say, forty and sixty would not be considered particularly strange.

This work included daily random variability by way of generating data with random variability around the systematic component which represents the mean disease incidence rate. In the large count scenarios the daily random variability was normally distributed around the systematic mean component with a distribution mean of zero and various standard deviations. The daily variation in the small counts was modeled as a lognormal with a distributional mean and standard deviation.

The choice of normally distributed daily variation for the large counts was based on the idea that large sums of individuals randomly arriving at a hospital emergency room ought to be approximately normally distributed via the Central Limit Theorem. For the small counts, the daily variability should be skewed to the right and bounded by zero. The lognormal provided a convenient way to model this behavior.

f. Whole Number Counts

Although the method of data generation inherently produces non-integer values, whole number counts were used in the data simulation. This was achieved by using the "ceiling" function in MatLab, which rounds a non-integer value generated by the data generation model up to the next largest integer. In addition, for some combinations of parameters, it is possible for the term $\beta + \alpha_i + \delta_i + Y_i(\beta)$ in Equation (1) to generate negative numbers. Hence, the "max" function was used in the data generation function to ensure all the synthetic observations were non-negative.

B. SIMULATION SCENARIOS

1. Case and Scenario Definition

A "case" for the purposes of this work is defined as a certain baseline (β), amplitude (A), mean (μ), and standard deviation (σ) combination. For each of these cases, three and four different outbreaks were simulated for the large and small count cases, respectively. A "scenario" is defined as a certain baseline, amplitude, mean, standard deviation, and outbreak size combination.

2. Large and Small Count Parameters

Once again, the overall goal of this work was to compare the relative performance of several methods for different disease incidence cases. Specifically, the two cases of small and large baseline values, parameterized as $\beta = 90$ and $\beta = 0$, were studied. For each of these, all possible combinations of those baselines with varying standard deviation and mean values were created, as shown in Tables 1 and 2.

The values in Table 1 and Table 2 result in $1\beta \times 3A \times 2\sigma = 6$ parameter combinations (cases) for the large count and $1\beta \times 1A \times 1\mu \times 2\sigma = 2$ parameter combinations for the small count. It should be noted that there are three amplitude sizes for the large counts, but only one for the small counts. Originally, there were three small count amplitude sizes, but in running the simulations it became clear that the adaptive regression methodology was very effective at removing the amplitude effect, so it was decided that there was no need to vary amplitude for the small counts. This reduced the total number of simulation scenarios to be run. This is discussed further in Chapter IV.

Day-of-the-week effects were also added to one large count case and one small count case for a total of 10 parameter conditions. This, combined with three different size outbreaks for the large counts and four different size outbreaks for the small counts, resulted in 33 simulation scenarios, which were used to assess the relative performance of the six methods (C1, C2, and C3 plus the CUSUM with three different sliding baselines).

Large Count Parameters			
	none	small	large
A	0	20	80
σ	n/a	10	30

Table 1. Parameter values for $\beta = 90$ (large count).

Small Count Parameters			
	none	small	large
A	n/a	n/a	6
σ	n/a	0.5	0.7
μ	n/a	1.0	1.0

Table 2. Parameter values for $\beta = 0$ (small count).

The "large" values in Table 1 result in disease incidence patterns similar to the CDC's S08 simulated datasets at www.bt.cdc.gov/surveillance/ears/datasets.asp. The "small" and "none" values result in disease incidence patterns similar to the S01 dataset, as well as other patterns that are intermediate between S01 and S08. Combinations of the values in Table 2 result in disease incidence patterns similar to S03, S04, S15, and S34.

3. Imposing Outbreaks

An outbreak in this work was defined to be a linear increase up to some day "X", followed by an equal linear decrease back to the normal incidence rate. Outbreaks durations from three to fifteen days were evaluated. For example, a seven day outbreak starting on day one would include a linear increase in the mean up to day four where it would peak and then decrease back to its original incidence rate on day eight. Small, medium, and large outbreaks for the large baseline means were defined as 10%, 25%, and 50% of the baseline mean, respectively. The four outbreak magnitudes for the small count cases were small, medium, large, and extra large; these were calculated by taking 10%, 25%, 50%, and 100% of the sum of the expected value and three standard deviations of *Y*, respectively.

In order to ensure a proper "warm-up" period, outbreaks were imposed on the synthetic syndromic surveillance data only after 100 days of no-outbreak data had been generated. During this period, each method was run on the data, just as would be done in an actual syndromic surveillance application. Also, in order to ensure a random beginning day for the outbreak, the first day of this 100 day warm-up period was a random day in the year, with the outbreak immediately following this 100 day period.

III. DESCRIPTION OF THE DETECTION ALGORITHMS

A. CURRENT UNIVARIATE METHODS

1. C1, C2, and C3 Methods

As described in Fricker (2007a), the current EARS methods called "C1," "C2," and "C3" are defined as follows. Let $X(t)$ be an observation for period *t*, such as the number of individuals arriving to a particular hospital with a specific syndrome on day *t*. The C1 method calculates the statistic $C_1(t)$ for day *t* as

$$C_1(t) = \frac{X(t) - \bar{X}_1(t)}{s_1(t)}$$

where $\bar{X}_1(t) = \sum_{i=t-1}^{t-7} X(i)/7$ and $s_1(t) = \sqrt{\sum_{i=t-1}^{t-7} \left(X(i) - \bar{X}_1(i) \right)^2 / 6}$, where $\bar{X}_1(t)$ and $s_1(t)$ are the moving sample mean and standard deviation, respectively. If $X(t)$ equals $\bar{X}_1(t)$ for seven continuous days (which can sometimes occur, particularly in the small count scenarios), $s_1(t)$ is automatically set to the previous day's $s_1(t)$. (Setting $s_1(t)$ to $s_1(t-1)$ avoids dividing by zero when calculating the C1 statistic.) The C1 method signals an alarm at time *t* when the C_1 statistic exceeds a fixed threshold, which in EARS is fixed at three sample standard deviations above the moving sample mean: $C_1(t) > 3$.

The C2 method is similar to the C1 method, but incorporates a two-day lag in the mean and standard deviation calculations. It calculates

$$C_2(t) = \frac{X(t) - \bar{X}_3(t)}{s_3(t)},$$

where $\bar{X}_3(t) = \sum_{i=t-3}^{t-9} X(i)/7$ and $s_3(t) = \sqrt{\sum_{i=t-3}^{t-9} \left(X(i) - \bar{X}_1(i) \right)^2 / 6}$, and signals an alarm when $C_2(t) > 3$.

The C3 method uses the C2 statistics from the past three days to calculate the C3 statistic, signaling an alarm when $C_3(t) > 2$. The C3 statistic for day *t* is calculated as

$$C_3(t) = \sum_{i=t}^{t-2} \max\left[0, C_2(i) - 1\right].$$

For the implementation of the three methods in this work, fixed thresholds as described above were not used. Instead, the thresholds were adjusted to achieve comparable average time to first signal (ATFS) for each scenario without outbreaks. In industrial statistical process control (SPC) terms, this is equivalent to choosing a threshold to achieve a desired in-control average run length.

2. The CUSUM Method

The cumulative sum (CUSUM) is a well known statistical process control methodology. Montgomery (2001) provides an excellent introduction to the CUSUM method in an industrial statistical process control setting and Hawkins and Olwell (1998) provide a comprehensive treatment of the CUSUM. Here, the focus is on the standardized CUSUM as described in Montgomery. Let

$$Y_i = \frac{X_i - \mu}{\sigma}$$

where X_i is the *i*th observation, μ is the expected mean, and σ is the standard deviation. If it is assumed that the X_is are normally distributed so that $X_i \sim N(\mu, \sigma^2)$, then $Y_i \sim N(0,1)$. The CUSUM for Y_i at time *i*, calculates

$$C_i^+ = \max\left[0, y_i - k + C_{i-1}^+\right]. \qquad (2)$$

The value *k* is called the reference value and is generally set at one-half of the shift in the mean that is desired to be detected quickly. In this research, Y_i will be the difference between the prediction an adaptive regression and the observed count and thus *k* was set in terms of a fraction of the standard deviation of the prediction error.

For the CUSUM, the threshold h was set such that when there has not been a disease or bioterrorism outbreak, the ATFS is equal to the ATFS for the EARS methods. If at some point $C^+ > h$, it is flagged as a possible disease outbreak or bioterrorism attack.

The CUSUM defined in Equation (2) is a one-sided CUSUM, meaning that, in this case, it will only detect increases in the mean. If it is important to detect both increases and decreases in the mean, a second CUSUM can be used to flag decreases. However, in syndromic surveillance, decreases are not relevant since it is only important to quickly flag increases in disease incidence.

B. ADAPTIVE REGRESSION

As stated before, syndromic surveillance data often has systematic trends, such as seasonal cycles, day of the week effects, and other patterns. One interpretation of this fact is that, given the last few weeks of counts are "high" on average, one would generally expect the next day's count to be "high" as well – that is, the counts exhibit autocorrelation. However, traditional statistical process control methods such as the CUSUM assume that observations are independent and identically distributed (*i.i.d.*), which is to say that these methods assume that the data *do not* contain such trends (Fricker, 2007a). This is clearly not the case with disease incidence data.

One approach is to model the systematic component of the data, use the model to forecast the next day's observation, and then apply the standard SPC methods to the forecast errors. For a model that results in *i.i.d.* forecast errors, such an approach is appropriate. Examples in the literature include the CUSUM applied to prediction errors in Brillman et al. (2005), the CDC's cyclical regression models discussed in Hutwagner et al. (2003), log-linear regression models in Farrington et al. (1996), and time series models in Reis and Mandl (2003). See Shmueli (2006) for additional discussion of the use of regression, and see time series methods for syndromic surveillance and Burkom et al. (2006) for a comparison of two regression-based methods and an exponential smoothing method applied to biosurveillance forecasting. Also see Lotze et al. (2006) for a detailed discussion of preconditioning applied to syndromic surveillance data.

The challenge for the purposes of this work was to construct a model capable of handling the systematic trends in the data. As stated earlier, for the purposes of these simulations, an assumption was made in order to model those trends: the mean follows an annual sinusoidal cycle. However, it was not assumed that it would be possible to go back far enough in time to model the sinusoidal cycle and make accurate predictions (i.e., to go back years in the data to predict the time in the current year one would expect to see a certain part of the sinusoid). There are two reasons for this:

(1) Multiple years of data would be needed, which is undesirable both because the data may not be available and because, even if it is, changes in population and other factors are likely to make older data unreliable.

(2) The annual cycle in the data generated in this work was fixed, which is artificial. In real data, disease incidence is variable, with the beginning, end, and amplitude of the sinusoidal pattern all being essentially random from year to year.

In order to still to capture the systematic component (i.e., the current sinusoid) of the data, yet not to attempt to predict it from year to year, the "adaptive regression model with sliding baseline" of Burkom et al. (2006) was employed. It can be described as follows. Let X_t be the observation (say chief complaint count on day t); the observations are regressed over time for some fixed number of time periods n (Burkom et al. used an 8-week period). Then the regression would look like

$$X_t = \beta_0 + \beta_1 t + \varepsilon$$

where β_0 is an intercept term, β_1 is the slope, and ε is the error term, meaning that, due to random variability, the model cannot fit perfectly. The model is fit using the least squares approach. The estimates of the counts for each time period $t,...,t-(n-1)$, where time is always relative to the current observation, are

$$\hat{X}_t = \hat{\beta}_0 + \hat{\beta}_1 t \ ,$$

and the forecast error for time t+1 is

$$\Delta_{t+1} = X_{t+1} - \left[\hat{\beta}_0 + \hat{\beta}_1 (t+1) \right].$$

Within the framework of syndromic surveillance, the model is refit at each time t (say, each day) and, if the model fits well, then one hopes that the Δ_i s are "small" according to some measure, such as mean square or mean absolute deviation. In the above two equations, subscripts for the time period t were left off of the coefficient terms for the sake of clarity, but it should be understood that, really, the coeffecients are reestimated at each time t.

One can easily generalize this approach to allow for nonlinearities. This may or may not be important, depending on how long of a time period is considered and how big the amplitude of the sinusoid is. To allow for a quadratic trend, one would include a quadratic term in the model,

$$X_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \varepsilon \, ,$$

and estimate the forecast error as

$$\Delta_{t+1} = X_{t+1} - \left[\hat{\beta}_0 + \hat{\beta}_1 (t+1) + \hat{\beta}_2 (t+1)^2 \right].$$

Finally, when introducing day-of-the-week effects in the data (and keeping the quadratic term), the model becomes

$$X_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 I_{Mon} + \beta_4 I_{Tues} + \beta_5 I_{Wed} + \beta_6 I_{Thurs} + \beta_7 I_{Fri} + \beta_8 I_{Sat} + \varepsilon \, ,$$

where the Is are indicators that take on the value 1 if t is that day of the week and 0 otherwise. The forecast error when day t+1 is a Sunday is

$$\Delta_{t+1} = X_{t+1} - \left[\hat{\beta}_0 + \hat{\beta}_1 (t+1) + \hat{\beta}_2 (t+1)^2 \right]$$

and for any other day of the week it is

$$\Delta_{t+1} = X_{t+1} - \left[\hat{\beta}_0 + \hat{\beta}_1 (t+1) + \hat{\beta}_2 (t+1)^2 + \hat{\beta}_j \right]$$

where j=3 for Monday, j=4 for Tuesday, etc.

Due to the nature of the regression model used, it attempts to predict the *mean* (i.e., the expected value) of the process which is generating the data, and, therefore, it will predict non-integer count values. Although the actual count values were forced to be integers, these predicted counts produced by the regression were not. Forcing these mean values to be whole numbers would only add excess noise to the simulations.

IV. COMPARISON METHODOLOGY

This chapter describes the methodology used to evaluate the relative performance of the various methods studied, as well as a description of how various input and threshold values were chosen. A master table of all parameters, input values, and threshold values is included at the end of the chapter.

A. METRICS

1. Average Time to First Signal Given a True Signal

The average run length (ARL) metric that is common in the SPC literature was not used in this work. Instead, a "time to first alarm" metric was employed. The problem with using the ARL is that the autocorrelation in syndromic surveillance data causes sequences of alarms to occur. Therefore, a metric similar to what is sometimes used in the SPC literature called "average time to signal" (ATS) was employed, but only the first signal was counted as an alarm; thus, the metric for this work was the "average time to first signal" (ATFS).

In the conduct of the simulations, the thresholds were set such that all methods had the same ATFS under some background (non-outbreak) disease incidence scenario; then, under outbreak conditions, the ATFSs given a true signal was calculated for the methods for each outbreak scenario. The goal for this measurement was to measure how quickly each method signals an alarm, given that the method detects the outbreak. In instances when an outbreak goes undetected by some method during its duration but is later detected, that detection is not counted in the ATFS given a true signal.

2. Fraction Missed

Another useful metric used to evaluate relative performance of the methods was the fraction missed. Unlike in the typical SPC comparison where the mean is assumed to jump and remain in an "out-of-control" condition until the detection algorithm signals, in syndromic surveillance an outbreak period is transitory. As such, it is possible for a

detection algorithm to miss the outbreak and subsequently signal after the outbreak has passed. Such signals are useless and it is important to understand how well a detection algorithm signals during the outbreak period. The fraction missed shows the fraction of the time that a method misses detecting an outbreak. Of course, the higher the fraction missed for a particular method, the worse the performance of that method.

B. CHOOSING INPUT AND THRESHOLD VALUES

1. Optimizing *n* for the Adaptive Regression Model

When using regression to predict future observations, the question that naturally arises is how much historical data should be used in order to fit the regression. In this work, two alternatives were drawn from existing practice and the literature: (1) seven days of historical data, which matches what is used in the C1, C2, and C3 EARS methods, and (2) eight weeks of historical data as recommended by Burkom et al. (2006).

Of course, with too few days of data, the regression model would have a tendency to follow the daily variability too much, missing the actual underlying data trend. While, generally speaking, more data should allow for a more detailed regression model and presumably a better prediction, often in syndromic surveillance the amount of available data is limited, or the older data is of questionable relevance due to changing trends or phenomena. Hence, there is a trade-off to be made between the amount of historical data used in a particular model and the predictive accuracy of that model.

This led to attempting to come up with an "optimal" *n* (number of days to regress over) for a given type of model for each case (baseline, amplitude, and standard deviation combination) under consideration since, just due to the daily variability, even a regression that perfectly captured the underlying data trend would have both positive and negative residuals (the difference between the actual observation and the predicted observation). The best *n* would be the *n* which minimizes the average squared residual over the entire simulated data series.

Two separate regression models were evaluated in order to decide on the best model, and an optimal *n* was found for the best model. The first model considered was a simple linear regression model with an intercept term and a slope term; this model found

the least-squares best fit line for the counts of the past *n* days and predicted the next day's count. The second was a multiple regression model with a quadratic term – i.e., the model had an intercept, slope, and quadratic term. This model was like the first with the added ability to capture curves in the data with the squared term, where the squared term was simply the time squared.

Simulations were run in order to determine the optimal *n* for each model. The results of the simulations were plotted, and an example of one such plot is shown in Figure 1 for the case of baseline 90, amplitude 80, and standard deviation 10. The optimal *n* was chosen simply by visual inspection with the criteria that the *n* be as small as possible but also as close to achieving the minimum average squared residual as possible. This meant that the chosen "optimal" *n* was the smallest *n* that achieved *most* of the reduction in the average squared residuals, as opposed to the *n* that occurred precisely at the minimum point on the curve. The result of this metric was that, given a chosen "optimal" *n*, there might be a larger *n* (such as the eight weeks used by one of the CUSUM methods) that performed better in terms of minimizing the average squared residual, simply because it used more days of data. For the curves in Figure 1, the "optimal" *n* was chosen to be 15 days for the linear model and 50 days for the quadratic model. The increase in average squared residual as the *n* gets large is due to the model over-regressing the data. That is, the regression model is using too many days of data to regress over, and it starts missing the more localized in-control trends.

Figure 1 also shows that the linear model achieved the same minimum average squared residual as the quadratic model but with a smaller *n*. This occurred consistently for all of the scenarios (see Appendix A), which led to the linear model being chosen as the superior model for all scenarios.

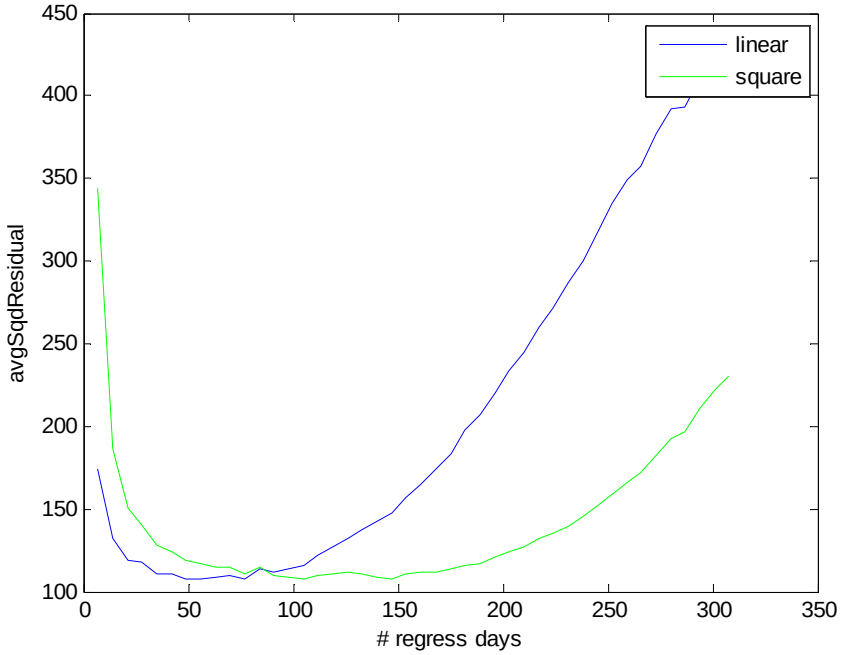


Figure 1. Predictive performance of the linear and quadratic models for baseline 90, amplitude 80, and standard deviation 10.

Tables 3 and 4 show the "optimal" *n* for all the scenarios without and with the day-of-the-week effects, respectively. As just described, it shows that the linear model achieved smaller *n*s than the quadratic model. It is important to note that when the amplitude is set to zero, there will never be an *n* value that over-regresses, causing the average squared residual to increase as *n* increases. The average squared residual will always decrease with an increase in *n* even if it only decreases marginally, theoretically making the optimal *n* value infinity. As before, visual inspection was used as before to pick an *n* value where the average squared residual value began to flatten out.

When the day-of-the-week effect is included, additional terms are needed in the predictive model, which increases the required amount of historical data needed to estimate the coefficients for those terms. For example, the presence of day effects in Burkom et al. (2006) is one of the reasons they used eight weeks of historical data. Table 4 summarizes the results.

"Optimal" *n* Values for Adaptive Regression				
no day-of-the-week effects				
β	**A**	**σ**	"optimal" *n* (linear)	"optimal" *n* (quadratic)
90	80	30	30	50
90	80	10	15	28
90	20	30	40	55
90	20	10	35	50
90	0	30	45	55
90	0	10	35	50
0	6	0.7	25	45
0	6	0.5	25	45

Table 3. "Optimal" *n* values for linear and quadratic adaptive regression models, with no day-of-the-week effect.

"Optimal" *n* Values for Adaptive Regression				
with day-of-the-week effects				
β	**A**	**σ**	"optimal" *n* (linear)	"optimal" *n* (quadratic)
0	6	2.8	56	75
90	80	10	40	55

Table 4. "Optimal" *n* values for linear and quadratic adaptive regression models, with the day-of-the-week effect included.

2. Choosing *k*

In Montgomery and Peck (1992) the variance of the predication error for a new observation *y** in a linear regression is

$$Var(y^* - \hat{y}^*) = \sigma_Y^2 \left[1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}} \right]$$

where:

- $\hat{y}^*$ is the predicted value and y^* the observed value for a specific x^* value;
- n is the number of (x,y) pairs of data in the original regression;
- S_{xx} is the sum of the squared differences between the xs in the regression and the mean of the xs:

- $S_{xx} = \sum_{i=1}^{n} \left(x_i - \bar{x} \right)^2$;
- $\left(x^* - \bar{x} \right)^2$ is the squared difference between the x^* value for the prediction and the mean of the *n* *x*s in the regression model; and,
- σ_Y^2 is the variance of Y which can be estimated as $\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$.

Thus, the standard deviation of the prediction error can be calculated as

$$s.d.(\text{prediction error}) = \sigma_Y \sqrt{1 + \frac{1}{n} + \frac{\left(x^* - \bar{x} \right)^2}{S_{xx}}}, \tag{3}$$

which then can be used as some multiple or fraction of the value for *k* in the univariate CUSUM.

For this problem, Equation (3) can be further simplified, since the *x* values are sequential integers representing time relative to the current day. That is, *n* is the number of days to regress over, with yesterday being "day *n*" and going back in time to "day 1" (which is the *n*th day in the past relative to today). By always trying to predict today's value and using the regression fit on the past *n* days, one can set $x^* = n+1$ and the mean is always $\bar{x} = (n+1)/2$. Therefore,

$$\left(x^* - \bar{x} \right)^2 = \left(n + 1 - \frac{(n+1)}{2} \right)^2 = \left(\frac{(n+1)}{2} \right)^2 = \frac{(n+1)^2}{2}. \tag{4}$$

Similarly, one can solve for S_{xx} as follows:

$$
\begin{aligned}
S_{xx} &= \sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 \\
&= \sum_{i=1}^{n}\left(i - \frac{n+1}{2}\right)^2 \\
&= 2\sum_{i=1}^{(n-1)/2} i^2 \\
&= \frac{1}{3}\left(2\left[\frac{n-1}{2}\right]^3 + 3\left[\frac{n-1}{2}\right]^2 + \left[\frac{n-1}{2}\right]\right)
\end{aligned}
$$

where the third step follows assuming n is odd and the last step follows since $\sum_{x=1}^{k} x^2 = \frac{1}{6}\left(2k^3 + 3k^2 + k\right)$. After some algebra, this becomes

$$S_{xx} = \frac{n\left(n^2 - 1\right)}{12}. \tag{5}$$

Substituting (4) and (5) into Equation (3) and doing some additional algebraic simplification gives

$$\sigma_{p.e.} = s.d.\left(\text{prediction error}\right) = \sigma_Y \sqrt{\frac{n^2 + 3n + 2}{n^2 - n}}.$$

Now, this is just a function of σ_Y ,and n where σ_Y is known or estimated. In the above equation, the square root part is called the "sigma multiple", and it has been plotted against various values of n in Figure 2.

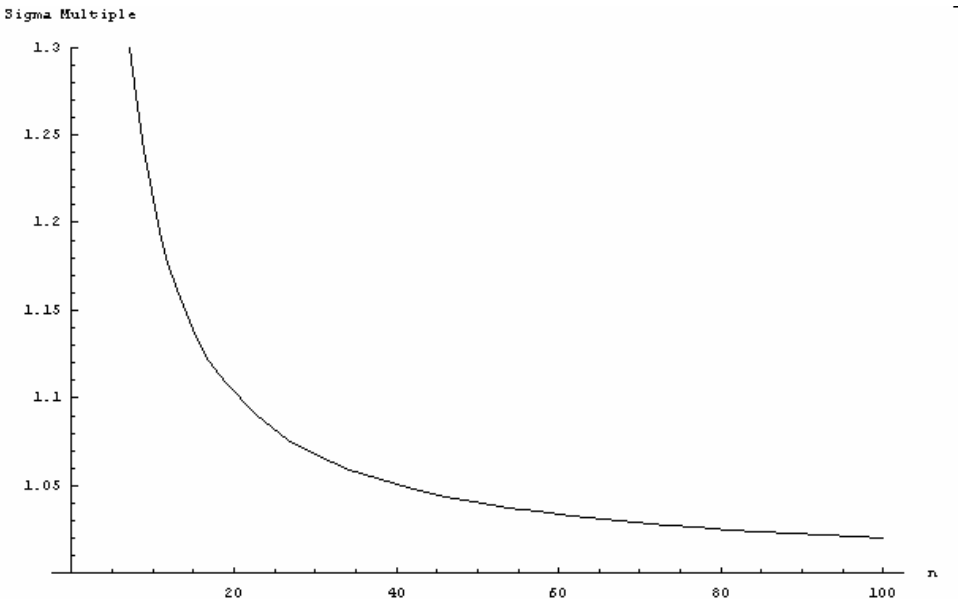


Figure 2. "Sigma multiple" for various *n* values

What this plot shows is that if optimal *n*s in the regression fall between about 30 and 60, then the standard deviation of the prediction error should be roughly about 1.07 to 1.03 times the size of the standard deviation used in the data simulation.

Assuming it is important to detect an increase in mean disease incidence of one standard deviation of the prediction error set $k = \sigma_{p.e.}/2$ in the CUSUM. Thus, for adaptive regression models with sliding baselines between 30 and 60, the reference value was set

$$k = \sigma_{p.e.}/2 \approx 1.05\sigma_Y/2 \approx \sigma_Y/2.$$

For large counts, the reference value was thus set $k = \sigma/2.$ For small counts, because of the lognormal random daily variation, $k = \sigma_Y/2$ where

$$\sigma_Y = \left[\left(\exp(\sigma^2 - 1)\right)\exp\left(2\mu + \sigma^2\right)\right]^{1/2}.$$

When *n* = 7 – i.e., when the length of the adaptive regression sliding baseline was matched to the amount of data as used in the C1, C2, and C3 methods – Figure 2 shows that $k = 0.65\sigma$ for the large counts and $k = 0.65\sigma_Y$ for the small counts. In particular,

when the random noise is generated by $LN(1.0,0.7)$, k = 2.8/2 = 1.4, and when the random noise is generated by $LN(1.0,0.5)$, k = 1.6/2 = 0.8.

3. Choosing *h*

A threshold for each method needed to be chosen in some way. While a low threshold would allow for faster detection of an outbreak, it would also increase the number of false alarms occurring. As is common practice, the non-outbreak ATFS was set to 100 days for this work, and the *h* (threshold) was empirically determined for each method. This was done in order to ensure equal performance among all the procedures in the absence of an outbreak. An ATFS of 100 means that for an in-control mean, a method using the predetermined *h* will signal an initial false alarm one out of every hundred days on average. It should be noted that the thresholds for the C1, C2, and C3 methods were chosen empirically in order to generate an ATFS of 100 days; this was different from the literature, where the C1, C2, and C3 had thresholds of 3, 3, and 2, respectively.

It is important to note that with the use of a 100 day ATFS, one result could be an excessive number of false alarms if one or more of these methods were being simultaneously run on the data from, say, 1000 hospitals. Specifically, with an ATFS of 100 days, one would expect roughly one false signal every 100 days for each hospital. With 1000 hospitals this would result in, on average, about 10 false alarms each day or 3650 false alarms over the course of a year. Then, if there was one real attack in the year, how would this one attack be distinguished from all the false signals? This is essentially the situation the EARS or BioSense systems are in when they use the fixed thresholds, since for the C1 and C2 our empirical thresholds were close to the fixed values used in practice by EARS.

Since longer ATFS periods would have taken significantly longer to simulate, the 100 day ATFS was chosen for computational convenience, where the goal was to assess the relative performance of the methods. It is assumed that this relative ranking would not change with changes in the ATFS. However, if these methods were implemented in simultaneous monitoring schemes, the thresholds could and should be adjusted

(increased) in order to appropriately balance between expected number of false alarms and the probability of detecting an actual attack.

Choosing the value for *h* was a simple procedure via simulation. For this work, it was done by first choosing an *h* value for a certain method, and then running the method on simulated data with no outbreak, using the *h* as the alarm threshold. The number of days the method ran without an alarm for each run was recorded and the average of those was calculated. If the ATFS was not very close to 100 days, the *h* was adjusted and the method rerun multiple times once again. This was repeated until the ATFS approached 100, and the standard error of the ATFS was less than 1.0. The *h* values for each of the methods for all of the scenarios are shown in Table 5.

				Parameters				Outbreak		*h* values						
Case	Scen.	Count Size	DoW effect?	β	A	μ	σ	Size	Value	7 day CUSUM	"optimal" CUSUM	56 day CUSUM	C1	C2	C3	"optimal" CUSUM *n* (days)
1	1	large	no	90	80	0	30	small	9.0	86.9	88.8	88.3	2.98	2.89	3.55	30
1	2	large	no	90	80	0	30	med	22.5	86.9	88.8	88.3	2.98	2.89	3.55	30
1	3	large	no	90	80	0	30	large	45.0	86.9	88.8	88.3	2.98	2.89	3.55	30
2	4	large	no	90	80	0	10	small	9.0	31	33	39	2.71	2.63	3.4	15
2	5	large	no	90	80	0	10	med	22.5	31	33	39	2.71	2.63	3.4	15
2	6	large	no	90	80	0	10	large	45.0	31	33	39	2.71	2.63	3.4	15
3	7	large	no	90	20	0	30	small	9.0	94.4	94.2	92.9	2.74	2.7	2.97	40
3	8	large	no	90	20	0	30	med	22.5	94.4	94.2	92.9	2.74	2.7	2.97	40
3	9	large	no	90	20	0	30	large	45.0	94.4	94.2	92.9	2.74	2.7	2.97	40
4	10	large	no	90	20	0	10	small	9.0	31.6	32	31.7	2.735	2.685	3.01	35
4	11	large	no	90	20	0	10	med	22.5	31.6	32	31.7	2.735	2.685	3.01	35
4	12	large	no	90	20	0	10	large	45.0	31.6	32	31.7	2.735	2.685	3.01	35
5	13	large	no	90	0	0	30	small	9.0	94.9	94	93	2.75	2.7	2.97	45
5	14	large	no	90	0	0	30	med	22.5	94.9	94	93	2.75	2.7	2.97	45
5	15	large	no	90	0	0	30	large	45.0	94.9	94	93	2.75	2.7	2.97	45
6	16	large	no	90	0	0	10	small	9.0	31.7	31.8	31	2.745	2.7	2.967	35
6	17	large	no	90	0	0	10	med	22.5	31.7	31.8	31	2.745	2.7	2.967	35
6	18	large	no	90	0	0	10	large	45.0	31.7	31.8	31	2.745	2.7	2.967	35
7	19	small	no	0	6	1.0	0.7	small	2.0	9.2	10.2	10.4	8.2	7.42	18.15	30
7	20	small	no	0	6	1.0	0.7	med	4.0	9.2	10.2	10.4	8.2	7.42	18.15	30
7	21	small	no	0	6	1.0	0.7	large	8.0	9.2	10.2	10.4	8.2	7.42	18.15	30
7	22	small	no	0	6	1.0	0.7	X-large	16.0	9.2	10.2	10.4	8.2	7.42	18.15	30
8	23	small	no	0	6	1.0	0.5	small	1.0	5.2	5.63	6	6.52	6.1	18.2	15
8	24	small	no	0	6	1.0	0.5	med	2.0	5.2	5.63	6	6.52	6.1	18.2	15
8	25	small	no	0	6	1.0	0.5	large	4.0	5.2	5.63	6	6.52	6.1	18.2	15
8	26	small	no	0	6	1.0	0.5	X-large	8.0	5.2	5.63	6	6.52	6.1	18.2	15
7	27	small	yes	0	6	1.0	0.7	small	2.0	n/a	10.8	10.8	7.7	7.3	15.5	56
7	28	small	yes	0	6	1.0	0.7	med	4.0	n/a	10.8	10.8	7.7	7.3	15.5	56
7	29	small	yes	0	6	1.0	0.7	large	8.0	n/a	10.8	10.8	7.7	7.3	15.5	56
7	30	small	yes	0	6	1.0	0.7	X-large	16.0	n/a	10.8	10.8	7.7	7.3	15.5	56
2	31	large	yes	90	80	0	10	small	9.0	n/a	39	41.9	2.7	2.61	3.38	40
2	32	large	yes	90	80	0	10	med	22.5	n/a	39	41.9	2.7	2.61	3.38	40
2	33	large	yes	90	80	0	10	large	45.0	n/a	39	41.9	2.7	2.61	3.38	40

Table 5. Summary of input parameters, outbreak parameters, *h*, and *n* values.

V. RESULTS

This chapter summarizes the results of the simulations, focusing on the large count, small count, and day-of-the-week effect results. See Appendix B for the plots from all 33 scenarios, and see Appendix C for all simulation code used. See Table 5 for a full summary of all parameters and input values, including outbreak values.

In general terms, the simulations were conducted as follows. A loop was used in order to have the program run for a set number of outbreaks. First, a random start day was chosen, and data was generated for a warm-up period of 100 days following the start day. Then, the outbreak was imposed, and the methods analyzed the data in an attempt to find an outbreak. Once an alarm occurred, the data generation terminated and the loop restarted. This process continued until the completion of all loops. Given a detection within the outbreak duration, the data generation would start again in the above description with a random start day. Given that the outbreak was not detected within its duration, the data generation would continue until a false alarm occurred (for programming convenience). Once a false alarm occurred, the data generation terminated and the loop restarted. In the context of statistics collection, the false alarms were kept separate from the actual alarms.

A. LARGE COUNT BASELINE MEAN

Although 18 simulation scenarios were run, the results can be largely summarized by the six plots in Figure 3, which show the results of using the case 2 parameters for scenarios 4, 5, and 6 for small, medium, and large outbreaks, respectively. For all large count cases (all with baseline 90), small, medium and large outbreaks were calculated to be of magnitude 9, 22.5, and 45, respectively. The upper, left-hand plot shows the average time to first signal given a true signal starting with a small outbreak, with the middle and lower plots being for medium and large outbreaks, respectively. The right-hand plots show, in increasing order of outbreak, the fraction of the time a procedure

missed detecting the outbreaks. Each plot evaluates the performance for the C1, C2, C3, and CUSUM methods, using the three sliding baseline lengths (*n*): 7, 15 (the "optimal" for case 2), and 56 days.

The graphs demonstrate that the CUSUM methods – particularly the ones with larger *n* – consistently outperform the three EARS methods studied. The three EARS methods performed similarly across all simulations. Of the EARS methods, the C2 method had the lowest fraction missed on the majority of the simulations, but the C1 was typically faster than the C2 and C3 for the ATFS given a true signal. The C3 method's performance varied, but it was typically outperformed by the C1 in the ATFS given a true signal and the C2 in the fraction missed. In the few scenarios where the C3 did outperform the C1 or C2, the difference in performance between the C3 and C1 or C3 and C2 was small.

While the EARS methods and the CUSUM with a 7 day sliding baseline tended to have slightly shorter ATFS given a true signal than the other two CUSUMs, this came at the expense of missing significantly more outbreaks than the CUSUMs with either a 15 ("optimal" *n* for this case) or 56 day sliding baseline. Recall that the metric used to choose the "optimal" *n* was visual inspection of the smallest *n* that achieved *most* of the reduction in the average squared residuals (see Chapter IV). There was a tradeoff in determining which method was better because, for the simulated data, the CUSUM with 56 day sliding baseline consistently had the lowest fraction missed but did not necessarily have the shortest ATFS given a true signal. It also had the largest *n* value, which might be considered undesirable in some situations. This difference in performance is evident in all outbreak magnitudes but is most evident with the larger magnitude outbreaks. For the case 2 scenarios considered here, the CUSUM with a 56 day sliding baseline is preferred, because it consistently had the lowest fraction missed, with an ATFS given a true signal in the 2-6.5 day range.

However, other methods, given they detected the outbreak within its duration, often had slightly lower ATFS given a true signal, and these ATFS values usually only differed by somewhere around 0-2 days, but usually towards the lower end of that range for outbreaks of up to nine day duration. This indicates that, were the simulated data real,

other methods might slightly outperform the CUSUM with a 56 day sliding baseline in terms of speed of detection *for the times that they detected within the outbreak's duration*. While it is true that a tradeoff exists here, these simulations are simply representations of actual data in terms of actual counts, outbreaks, and different variations (annual cycle, standard deviation, etc.). Therefore, the specific value of the ATFS given a true signal should only be used to determine the *relative* performance of the methods – it should not be used to conclude that the real-life ATFS given a true signal would be what is shown in the plots.

Also, in Figure 3 the C1 and CUSUM with a 7 day sliding baseline suffer the most from being contaminated by the outbreak data in the largest magnitude outbreak scenarios. That is, in the lower, right-hand plot the fraction missed by these two procedures increases for longer duration outbreaks. This is because, if these procedures fail to detect the outbreaks early on, they begin to incorporate the outbreak data into their calculations (either the moving average for the C1 or the adaptive regression predictions for the CUSUM). As outbreak data is incorporated into the calculations, it becomes increasingly difficult to distinguish the outbreak from the baseline mean of disease incidence. In comparison, the two day lag in the C2 procedure essentially delays this problem. It can be seen in scenario 6 of Figure 3 that the C2 fraction missed increases for outbreaks of longer duration. This is also true for the C3, which is a function of the C2 test statistic. In spite of the contamination, the relative performance of the EARS methods remains unchanged.

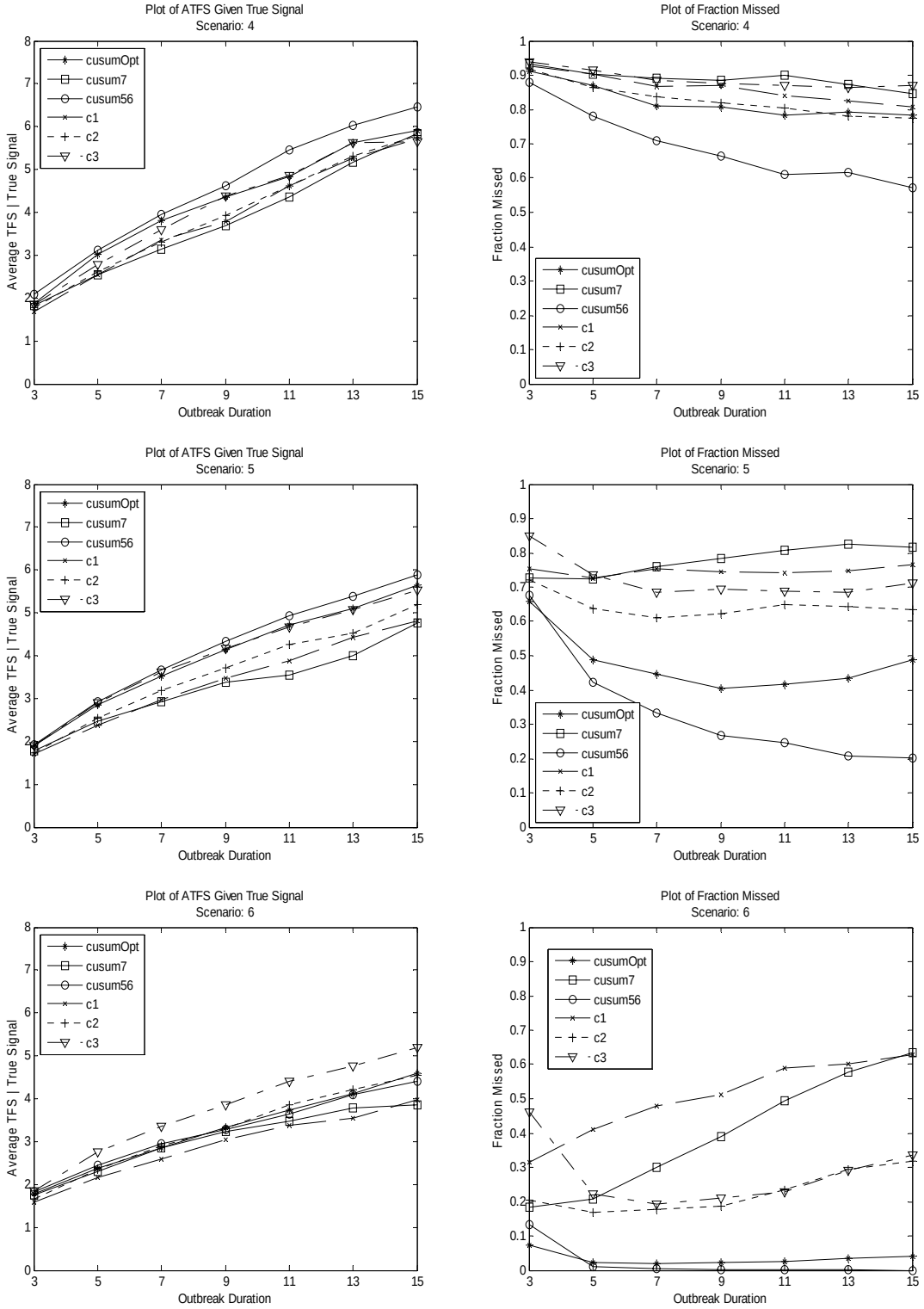


Figure 3. Performance of the procedures for case 2 (large count), scenarios 4-6.

B. SMALL COUNT BASELINE MEAN

When the first entire set of small count scenarios was run, a mistake was made that set the *k* value to a lower value than shown in the calculations used to set *k*. Although lower, the threshold values were set to have the non-outbreak ATFS equal to 100, and the relative performance of each algorithm did not change. As with the large counts it could clearly be seen in the plots that varying amplitude made no discernable difference in the results. That is, that the amplitude simply was not a factor in determining which methods were superior to the others, because the adaptive regression did a sufficient job of removing the sinusoidal effects in the data stream for the CUSUMs. Due to the error discovered, the small count scenario simulations were run again with the correct desired *k* value, but this time they were conducted by holding the amplitude constant at a value of 6 (as mentioned briefly in Chapter II), which reduced the total number of simulation scenarios. The four outbreak magnitudes for the small count cases were small, medium, large, and extra large; these were calculated by taking 10%, 25%, 50%, and 100% of the sum of the expected value and three standard deviations, respectively.

Figure 4 and Figure 5 show the results of using the case 7 parameters for scenarios 19, 20, 21, and 22; these plots are a good representation of all small, medium, large, and extra large magnitude outbreaks scenarios for the small count cases, where the outbreaks were of size 2, 4, 8, and 16, respectively. Comparing Figure 3 (large count) to Figures 4 and 5 (small count), one can see their overall similarity. The CUSUMs with the longer sliding baselines are again the best performing procedures (where the "optimal" sliding baseline in this scenario was 30 days), and the EARS methods performed relatively similarly. Again, the C1 generally has the lowest ATFS given a true signal and, of the EARS methods, the C2 generally has the lowest fraction missed. Although the C1 does have the shortest ATFS given a true signal, it completely misses about 85 to 90 percent of the outbreaks. In comparison, the CUSUMs using either a 30 or 56 day sliding baseline catch virtually all of the outbreaks. For these CUSUMs, the ATFS given a true signal was about two days for a three day outbreak, and the ATFS given a true signal was about four days for a 15 day outbreak.

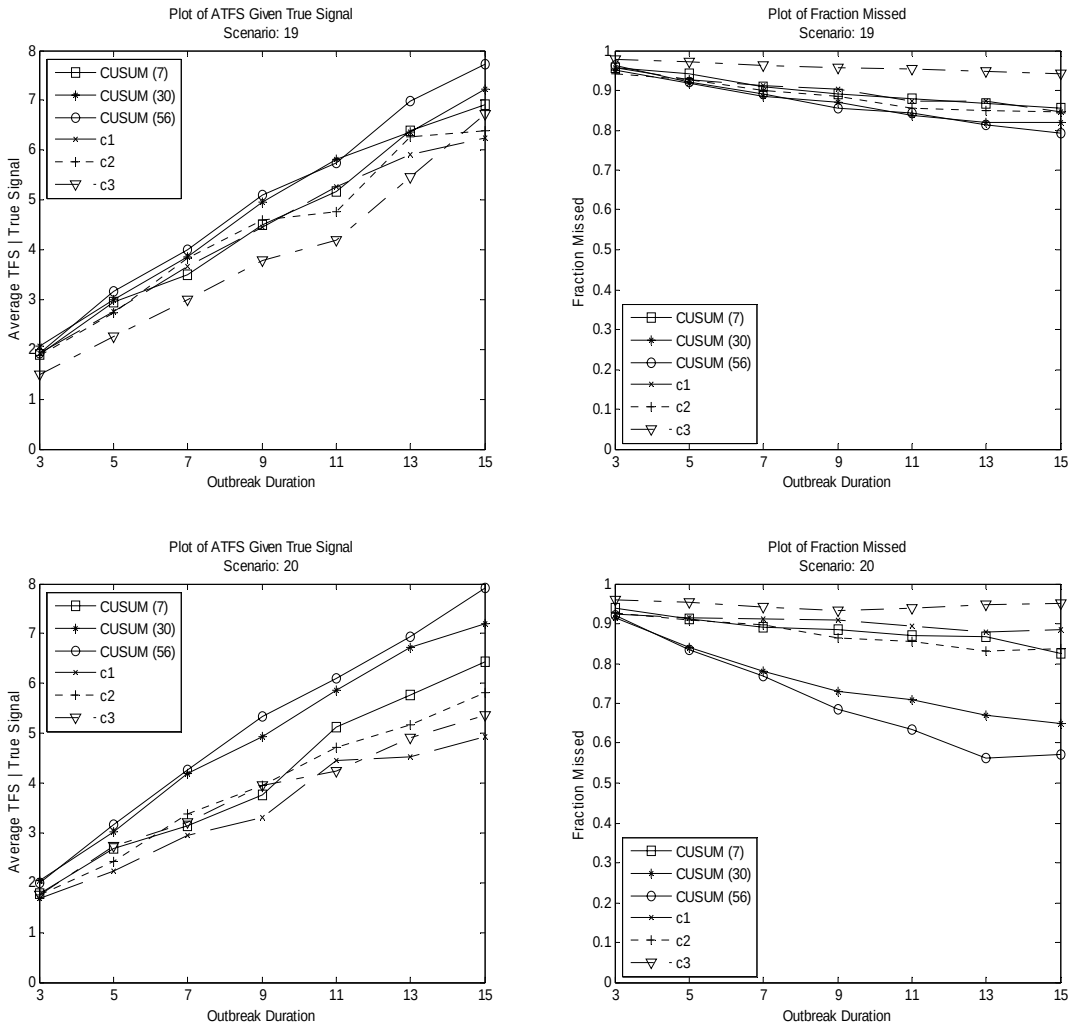


Figure 4. Performance of the procedures for case 7 (small count) for scenarios 19-20.

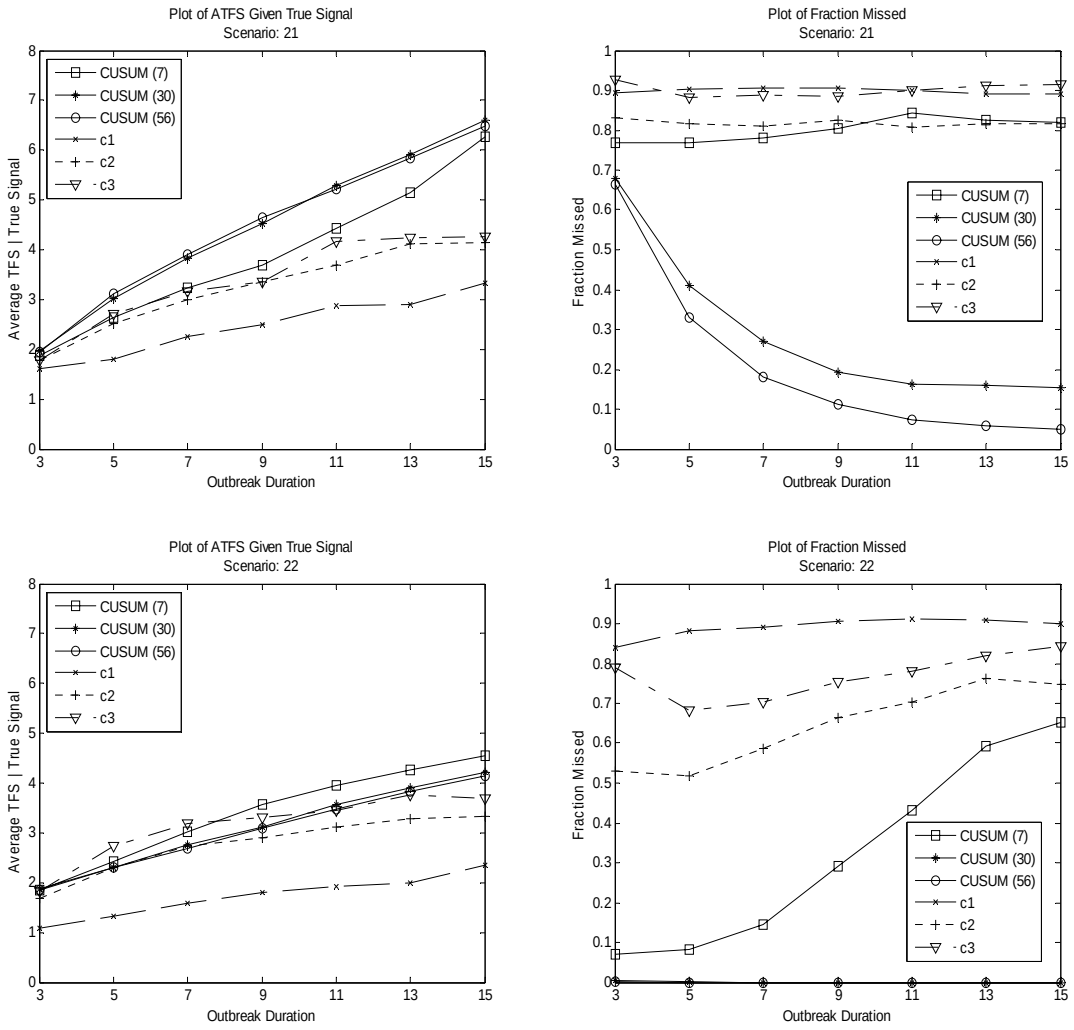


Figure 5. Performance of the procedures for case 7 (small count) for scenarios 21-22.

C. DAY OF THE WEEK EFFECTS

Lastly, seven scenarios that included day-of-the-week effects were run by revisiting large count case 2 and small count case 7, this time including the day-of-the-week effect in the baseline disease incidence. Figure 6 shows the results of these simulations for large count case 2 (scenarios 31-33) parameters, and Figure 7 and Figure 8 show the results of these simulations for small count case 7 (scenarios 27-30) parameters. Outbreaks for large count were of size 9, 22.5, and 45, and outbreaks for small count were of size 2, 4, 8, and 16, defined as stated earlier.

It should be noted that these plots do not include a CUSUM with a 7-day sliding baseline since seven data points are insufficient to estimate the eight parameters in the adaptive regression (slope, intercept, and six say-of-the-week indicators). The results with the day-of-the-week effect included were basically the same as in the large and small count cases that did not include it. Once again, the CUSUMs based on the adaptive regression residuals clearly outperform the EARS methods.

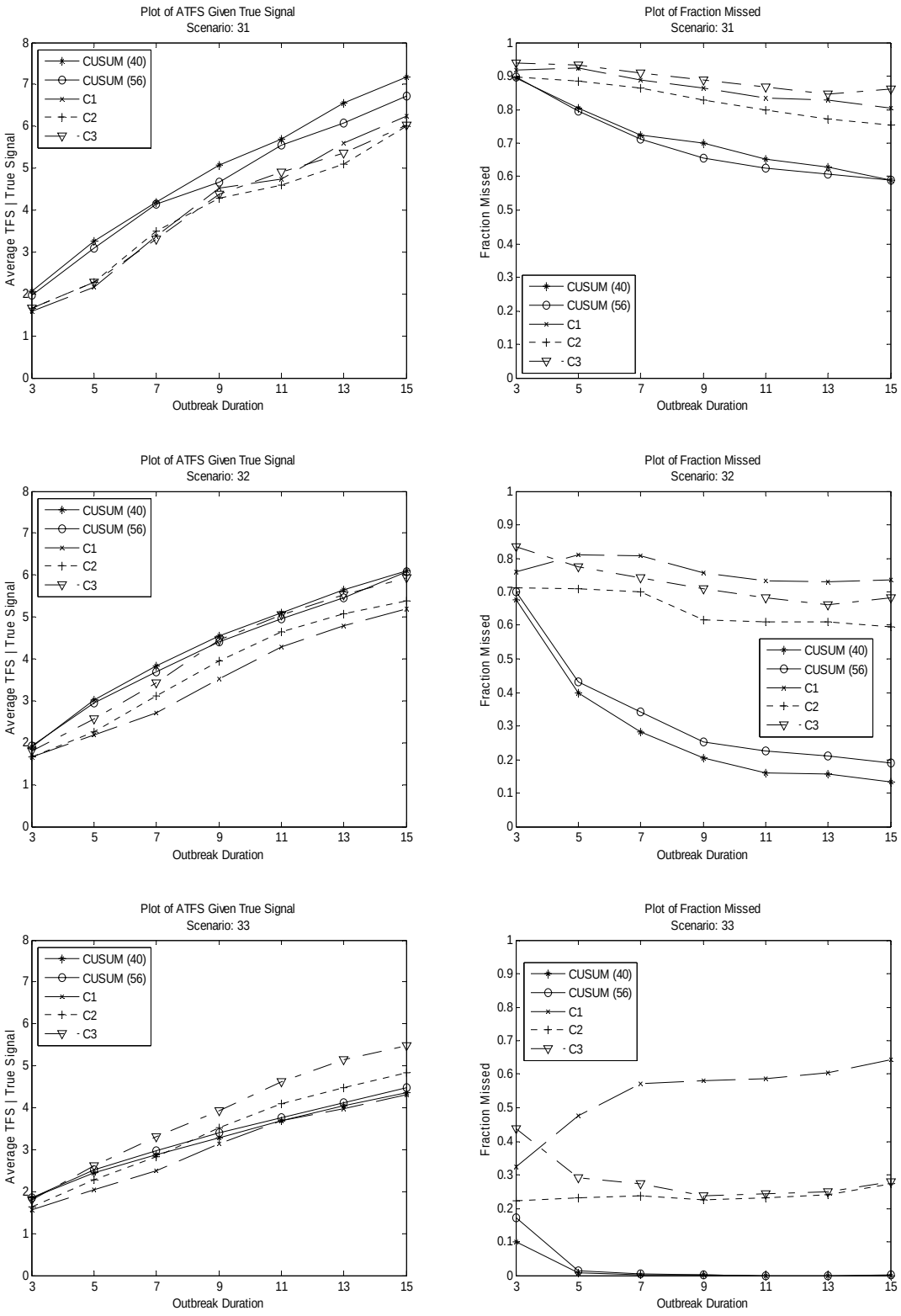


Figure 6. Performance of the procedures for case 2 (large count) for scenarios 31-33 with day-of-the-week effects included.

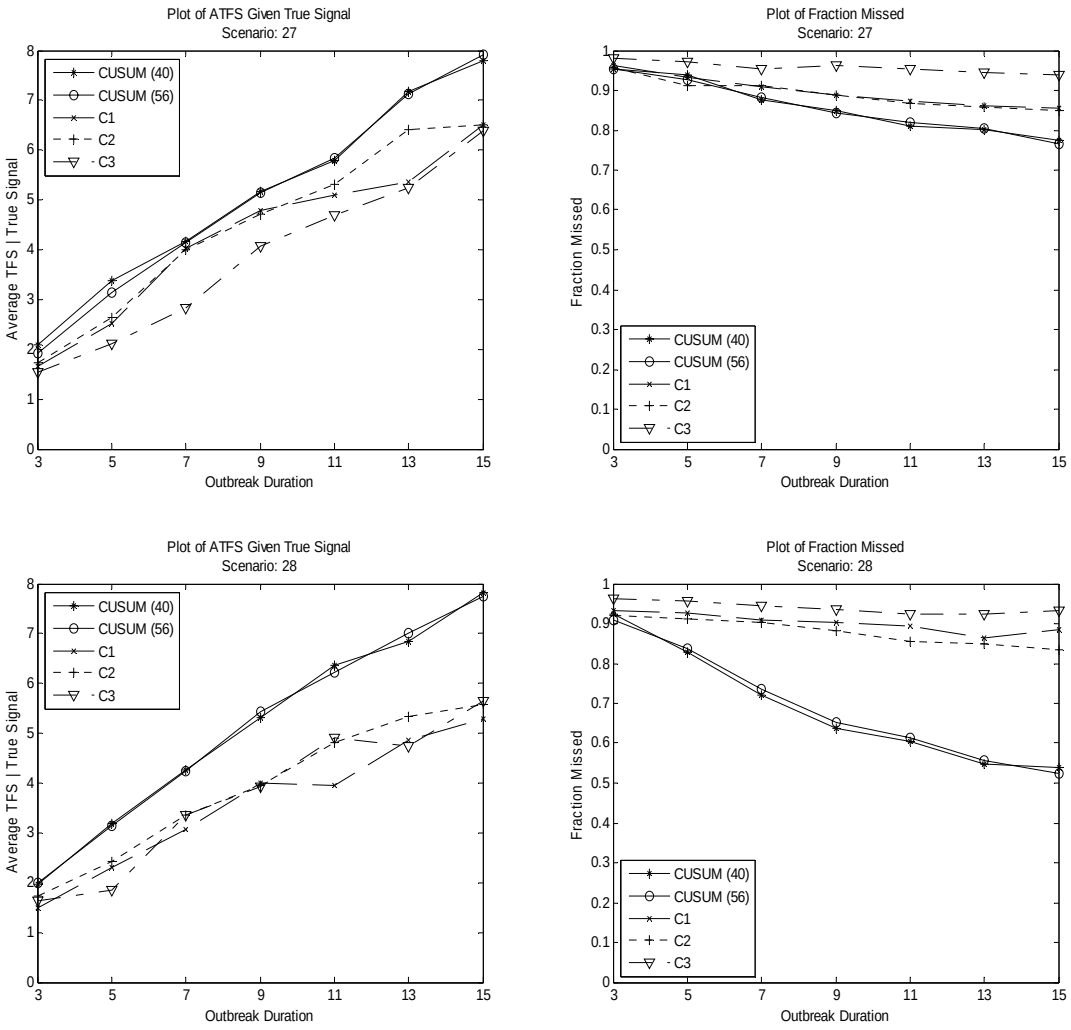


Figure 7. Performance of the procedures for case 7 (small count) for scenarios 27-28 with day-of-the-week effects included.

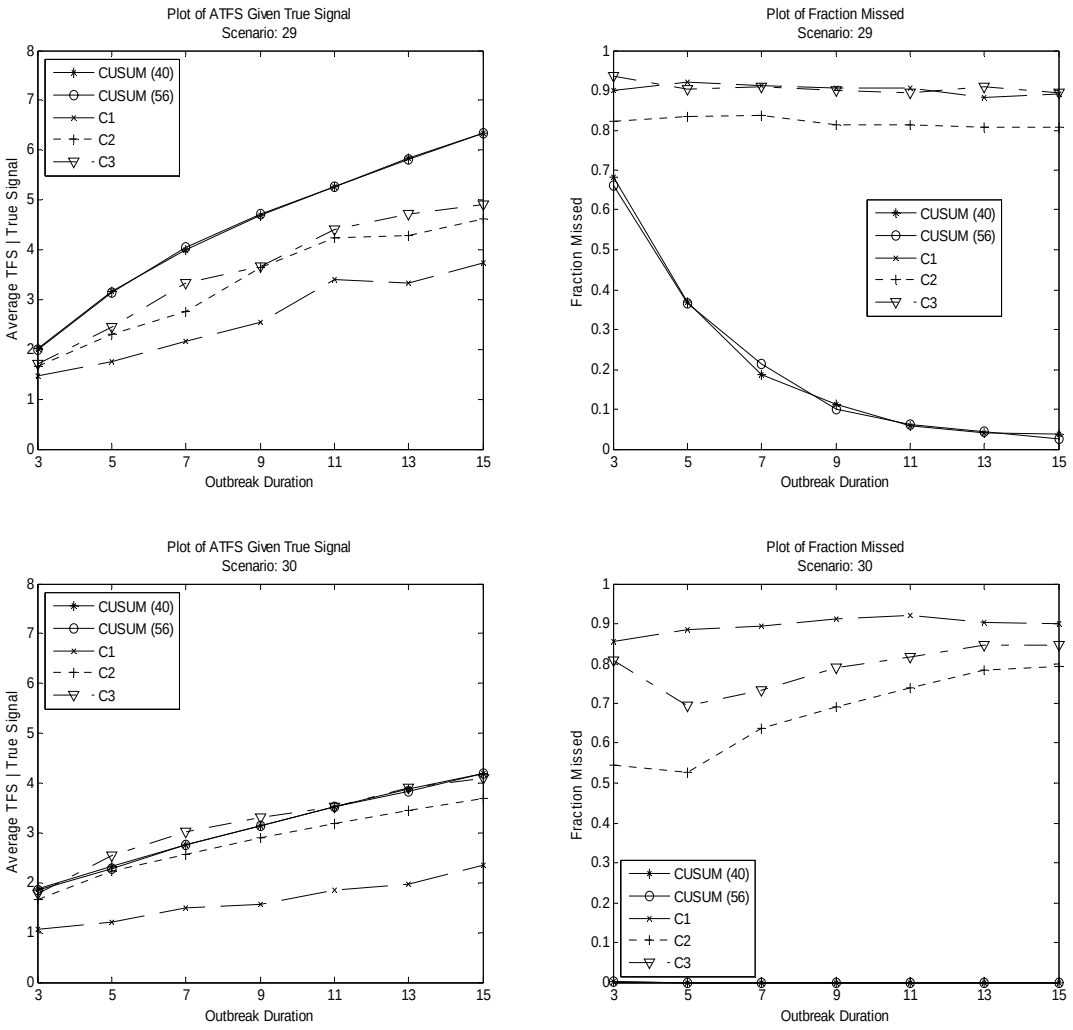


Figure 8. Performance of the procedures for case 7 (small count) for scenarios 29-30 with day-of-the-week effects included.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A clear conclusion resulting from evaluating the EARS methods versus CUSUM methods applied to the residuals of adaptive regression is that the CUSUM methods with longer sliding baselines perform significantly better in all the scenarios evaluated. These scenarios were chosen to imitate the major features of syndromic surveillance data over a wide variety of conditions. In particular, the EARS methods failed to catch a significant fraction of outbreaks across a wide variety of background disease incident patterns (large and small daily counts; large, small, and no seasonal cycles; large and small random daily fluctuations; and with and without day-of-the-week effects) and a variety of outbreak magnitudes and durations. Overall, the CUSUM methods, particularly the one that uses the 8-week (56 day) sliding baseline, outperformed the EARS methods. Therefore, standard syndromic surveillance systems using the EARS methods would benefit from replacing the EARS methods with a CUSUM method, setting the CUSUM thresholds in a similar fashion as done in this research in order to minimize the false alarm burden as much as possible.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Brillman, J.C., Burr, T., Forslund, D., Joyce, E., Picard, R., and E. Umland (2005). "Modeling Emergency Department Visit Patterns for Infectious Disease Complaints: Results and Application to Disease Surveillance," BMC Medical Informatics and Decision Making, 5.

Burkom, H.S., S.P. Murphy, and G. Shmueli (2006). "Automated Time Series Forecasting for Biosurveillance," *Statistics in Medicine*, accepted (available at http://www3.interscience.wiley.com/cgi-bin/abstract/114131913/). June 2007.

Farrington, C.P., Andrews, N.J., Beale, A.D., and M.A. Catchpole (1996). A Statistical Algorithm for the Early Detection of Outbreaks of Infectious Disease, *Journal of the Royal Statistical Society, Series A (Statistics in Society)*, 159, 547-563.

Fricker, R.D., Jr. (2007a). Syndromic Surveillance, *Encyclopedia of Quantitative Risk Assessment* (to appear).

Fricker, R.D., Jr., (2007b). Directionally Sensitive Multivariate Statistical Process Control Methods with Application to Syndromic Surveillance, *Advances in Disease Surveillance*, www.isdsjournal.org, 3:1. June 2007.

Fricker, R.D., Jr., and H. Rolka (2006). Protecting Against Biological Terrorism: Statistical Issues in Electronic Biosurveillance, *Chance*, **91**, 4-13.

Hawkins, D.M. and D.H. Olwell (1998). *Cumulative Sum Charts and Charting for Quality Improvement*, Springer.

Hutwagner, L., Thompson, W., Seeman, G.M., and Treadwell, T (2003). "The Bioterrorism Preparedness and Response Early Aberration Reporting System (EARS)," Journal of Urban Health: Bulletin of the New York Academy of Medicine, **80**, 89i-96i.

Lotze, T., S.P. Murphy, and G. Shmueli (2006). Preparing Biosurveillance Data for Classic Monitoring (draft), in submission to *Advances in Disease Surveillance*.

Montgomery, D.C., and E.A. Peck (1992). *Introduction to Linear Regression Analysis*, 2nd ed., Wiley.

Montgomery, D.C. (2001). *Introduction to Statistical Quality Control*, 4th edition, John Wiley & Sons, New York.

Reis, B.Y., and K.D. Mandl (2003). "Time Series Modeling for Syndromic Surveillance," *BMC Medical Informatics for Decision Making*, **3**.

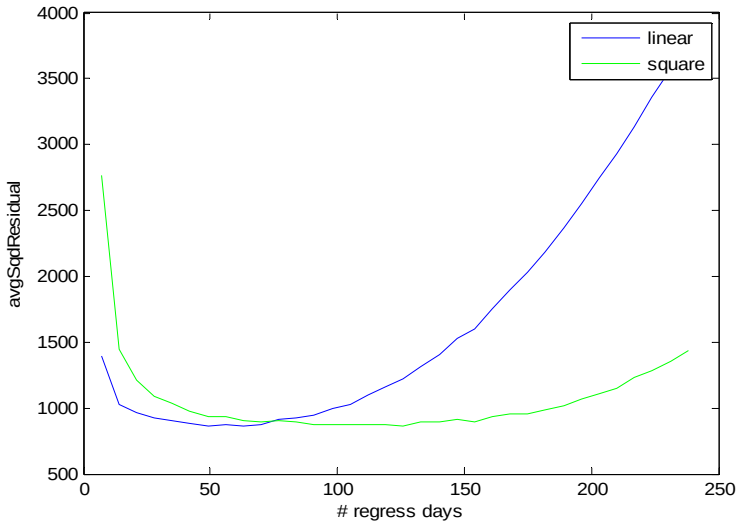
Shmueli, G. (2006). "Statistical Challenges in Modern Biosurveillance," *Technometrics* (in submission), draft dated September 18, 2006.

Sosin, Daniel M. (2003). Syndromic Surveillance: The Case for Skillful Investment View, *Biosecurity & Bioterrorism*, 1(4), Mary Ann Liebert, Inc., 247-253.

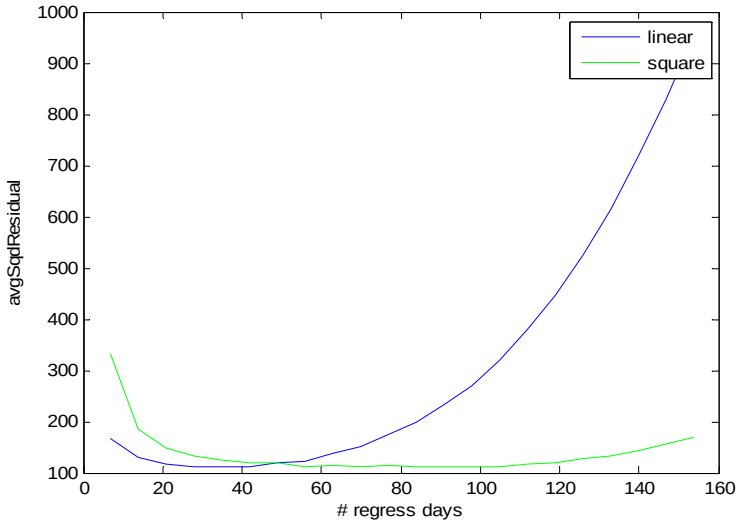
Stoto, M.A., Fricker, Jr., R.D., Jain, A., Diamond, A., Davies-Cole, J.O., Glymph, C., Kidane, G., Lum, G., Jones, L., Dehan, K., and C. Yuan (2006). Evaluating Statistical Methods for Syndromic Surveillance, in *Statistical Methods in Counterterrorism: Game Theory, Modeling, Syndromic Surveillance, and Biometric Authentication*, A. Wilson, G. Wilson, and D. Olwell, eds., New York: Springer.

Zhu, Y., Wang, W., Atrubin, D., and Y. Wu (2005). Initial Evaluation of the Early Aberration Reporting Sysytem --- Florida, *Morbidity and Mortality Weekly Report (Supplement),* Centers for Disease Control and Prevention, **54**, pp. 123-130, August 26, 2005.

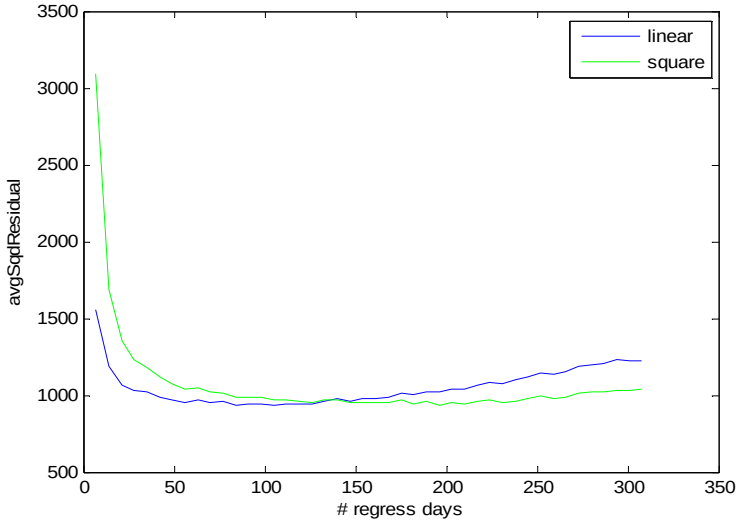
APPENDIX A: "OPTIMAL" *n* PLOTS



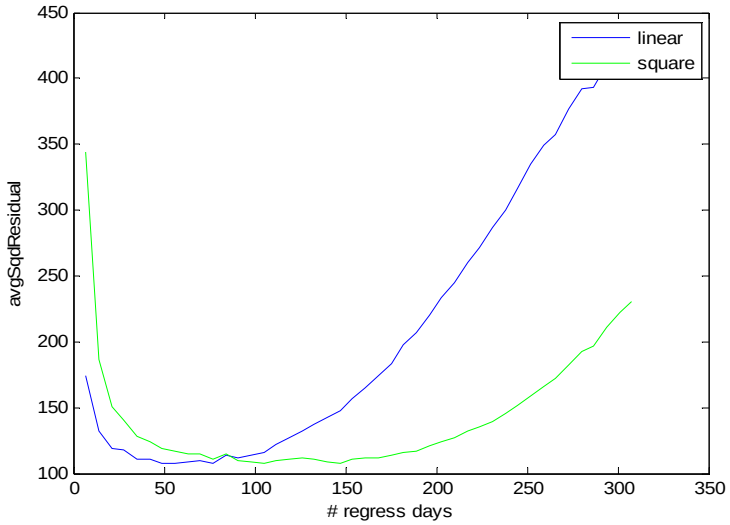
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	80	30	30	50



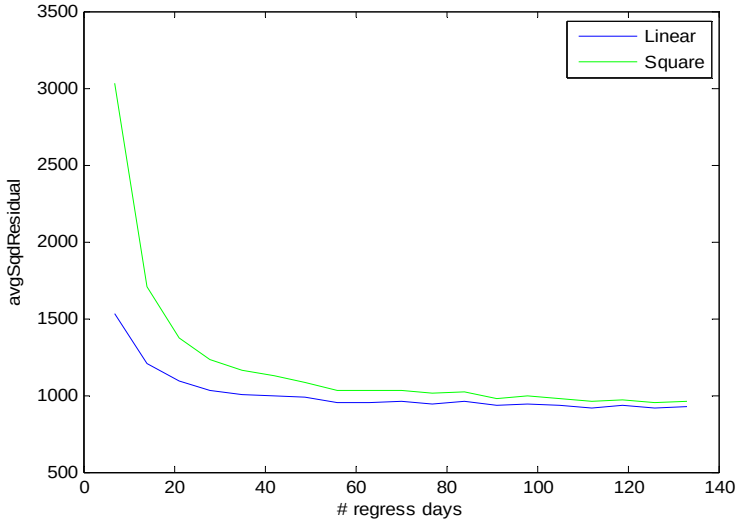
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	80	10	15	27



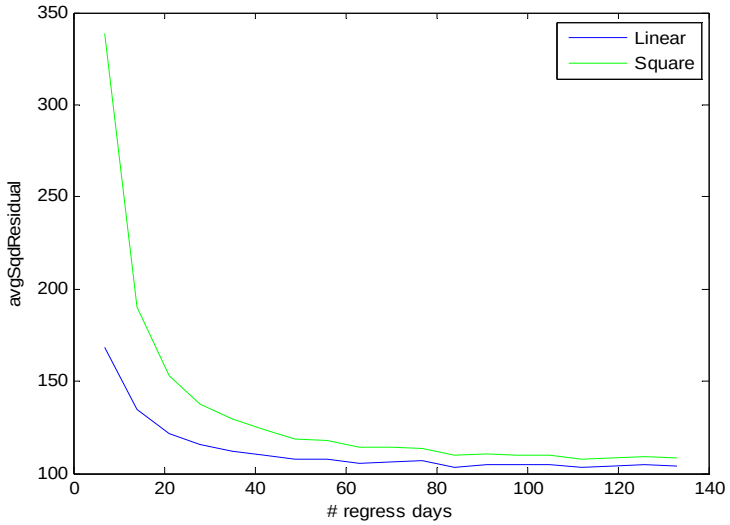
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	20	30	40	55



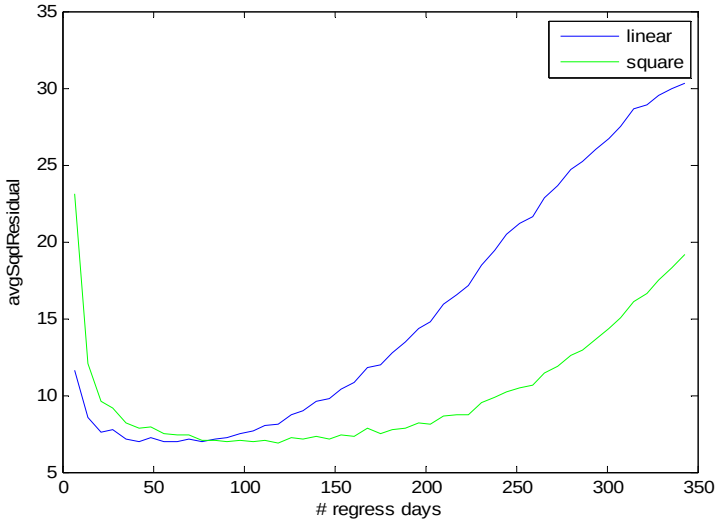
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	20	10	35	50



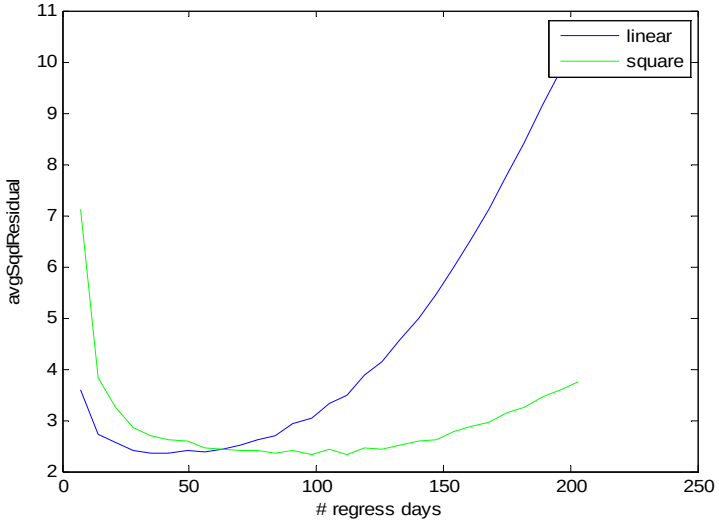
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	0	30	45	55



Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	0	10	35	50

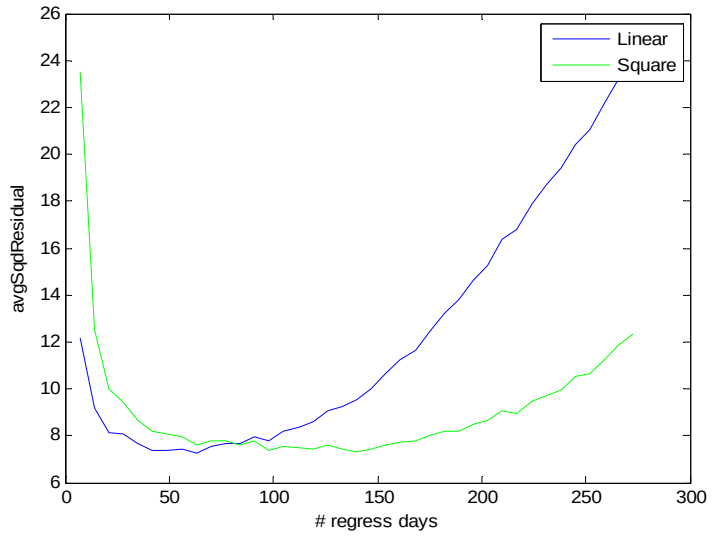


Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
0	6	2.8	30	45

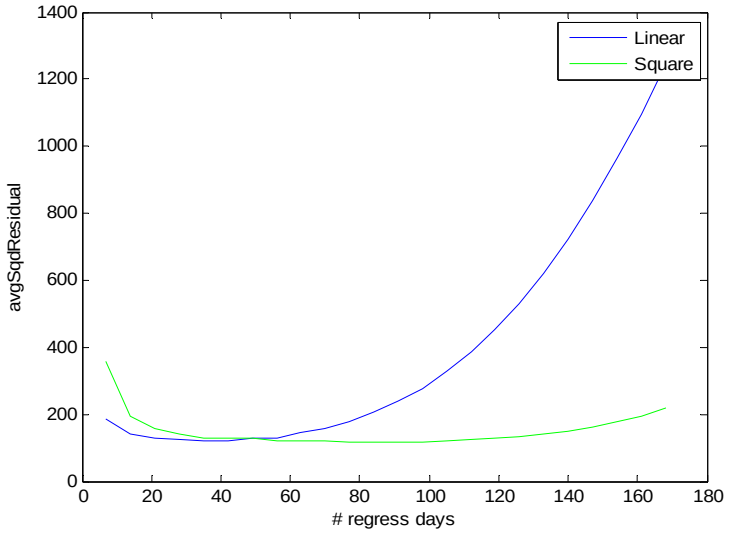


Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
0	6	1.6	15	45

Optimal *n* Plots with the Day-of-the-Week-Effect



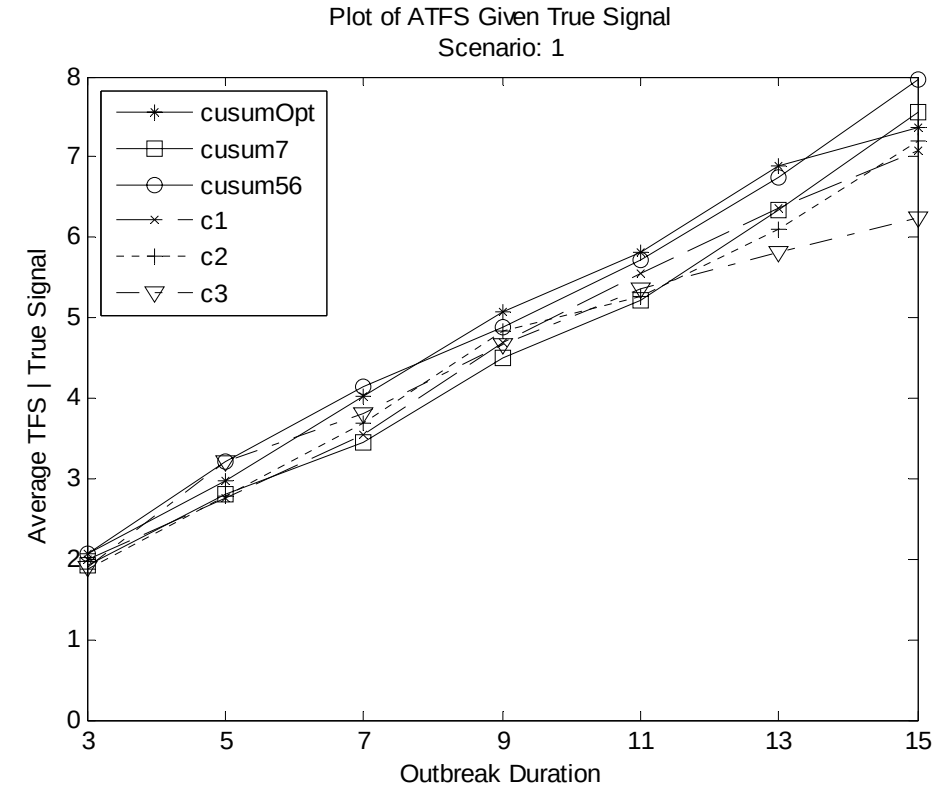
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
0	6	2.8	56	75

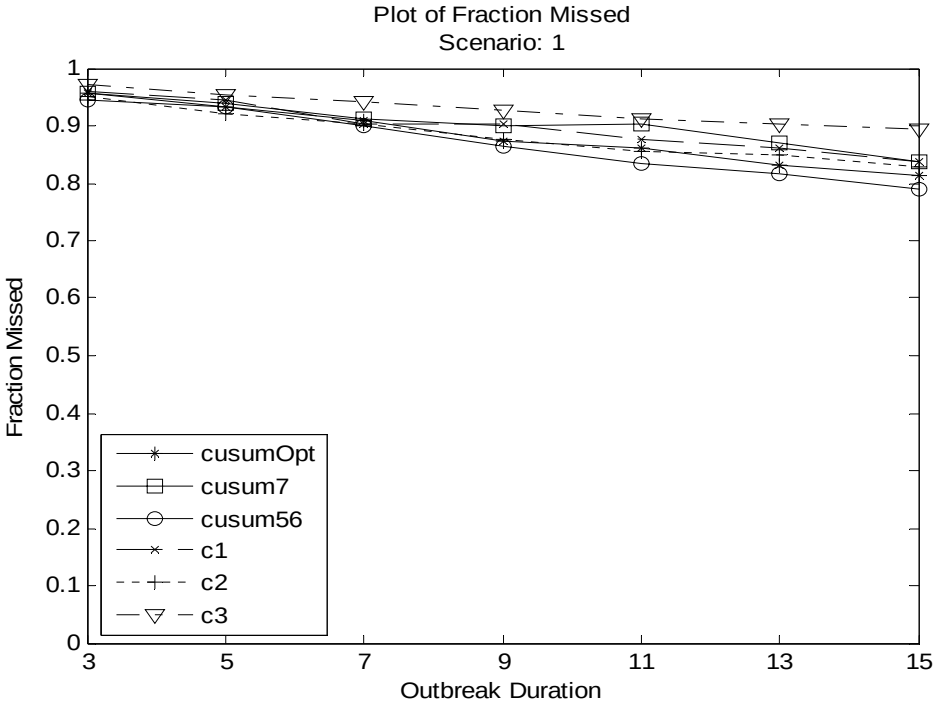


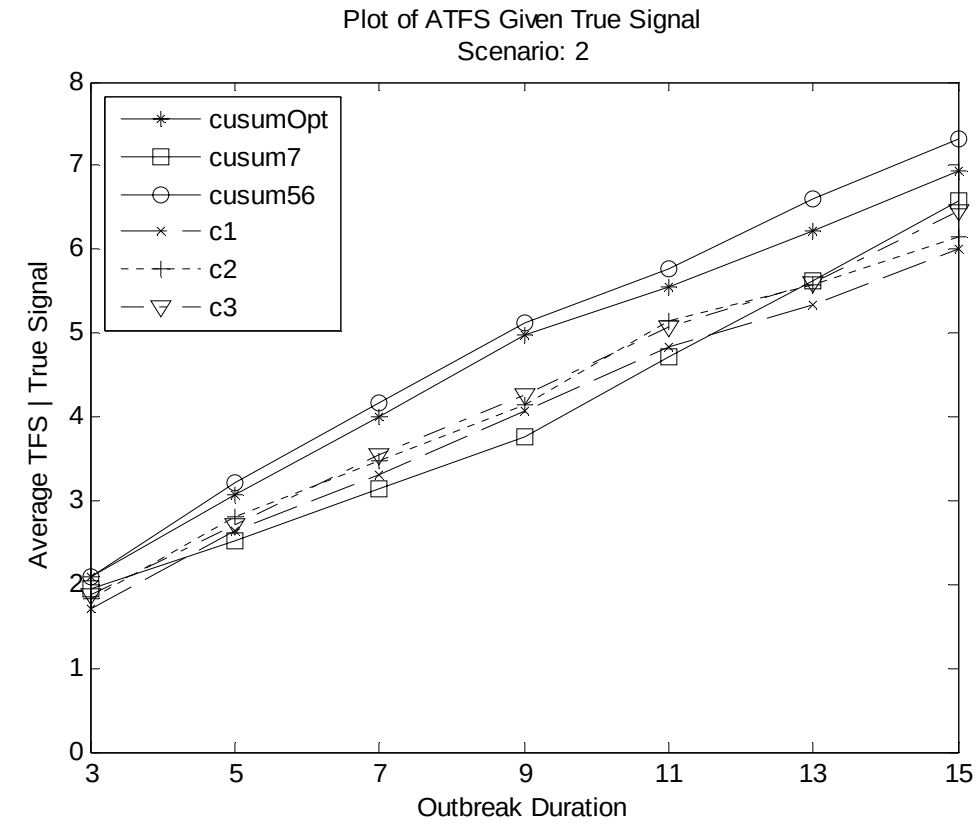
Baseline Mean	Amplitude	Standard Deviation	Optimal n (linear regression)	Optimal n (square regression)
90	80	10	40	55

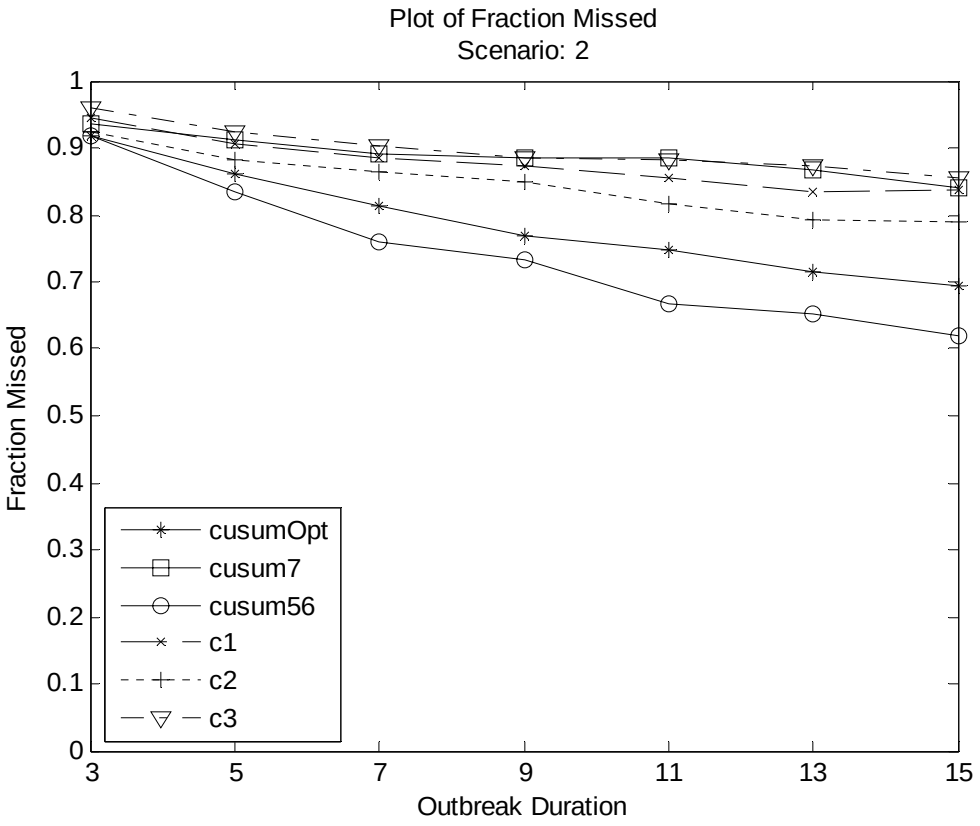
THIS PAGE INTENTIONALLY LEFT BLANK

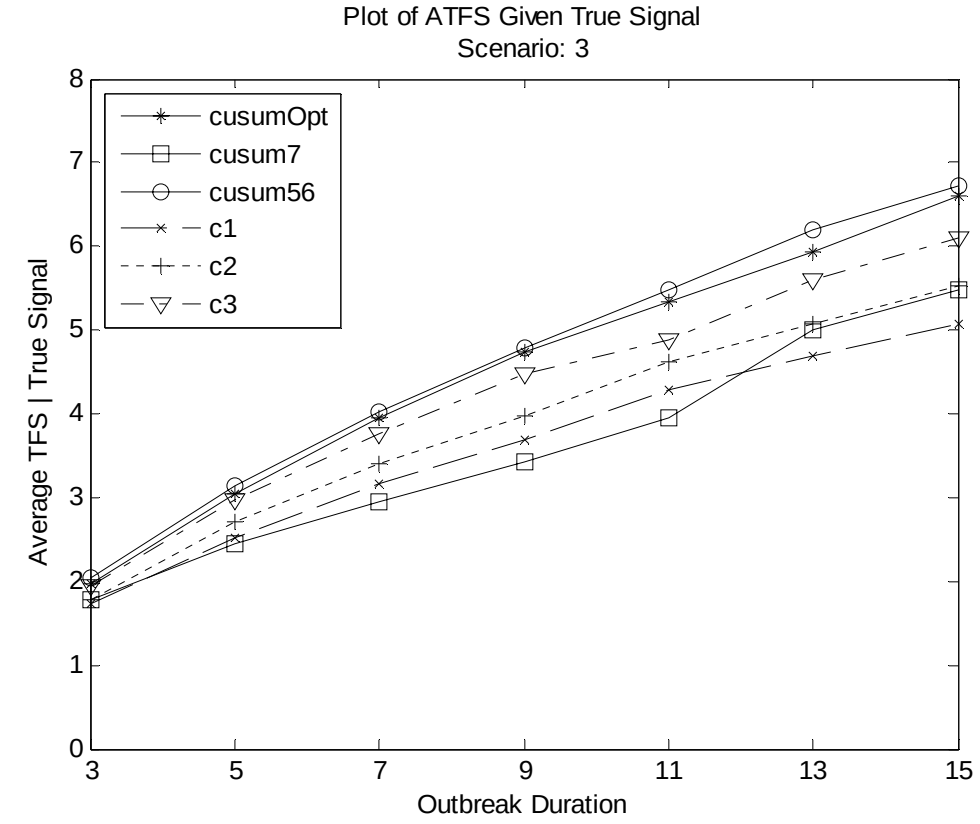
APPENDIX B: COMPARION RESULTS PLOTS

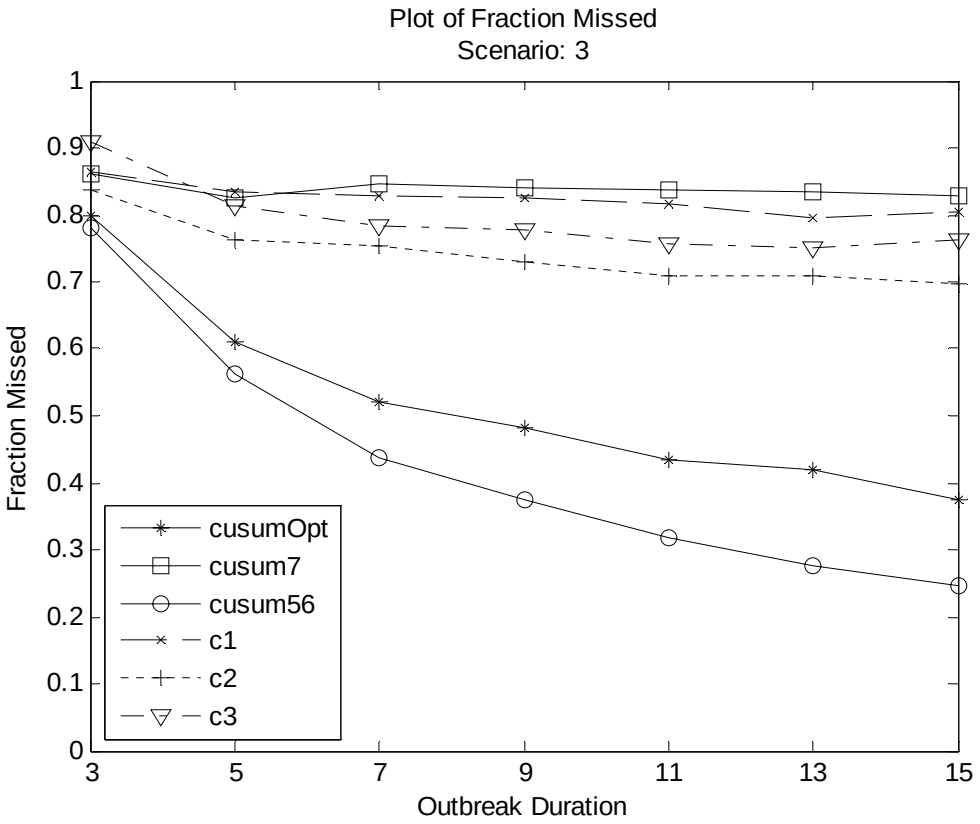


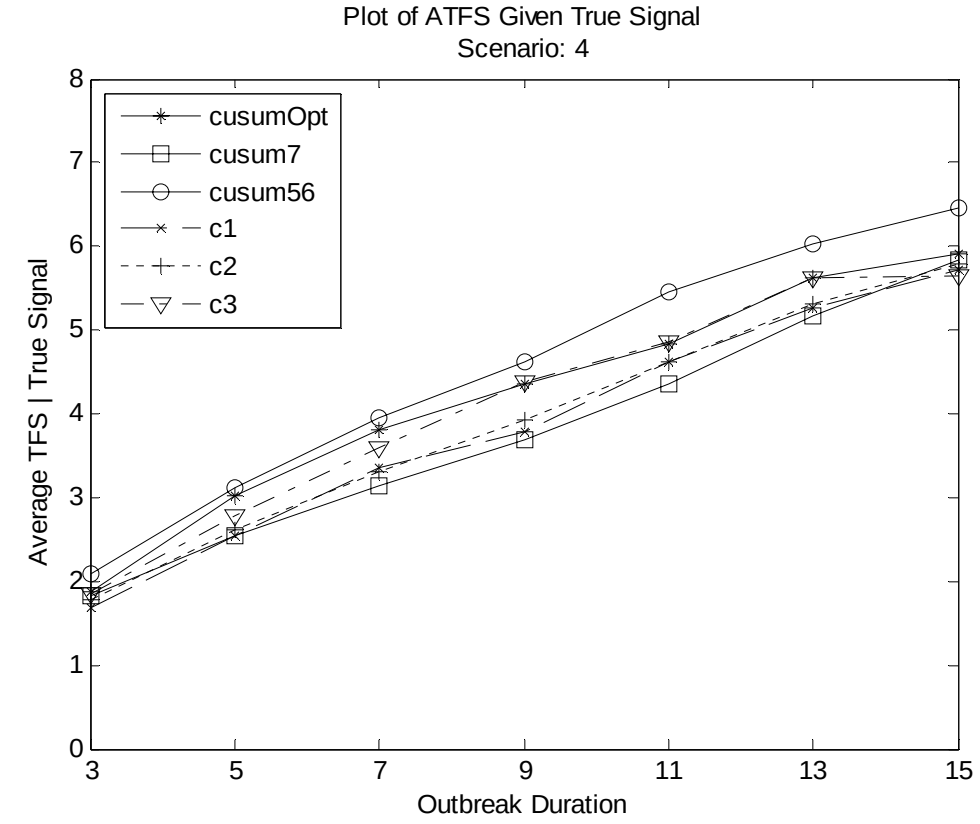


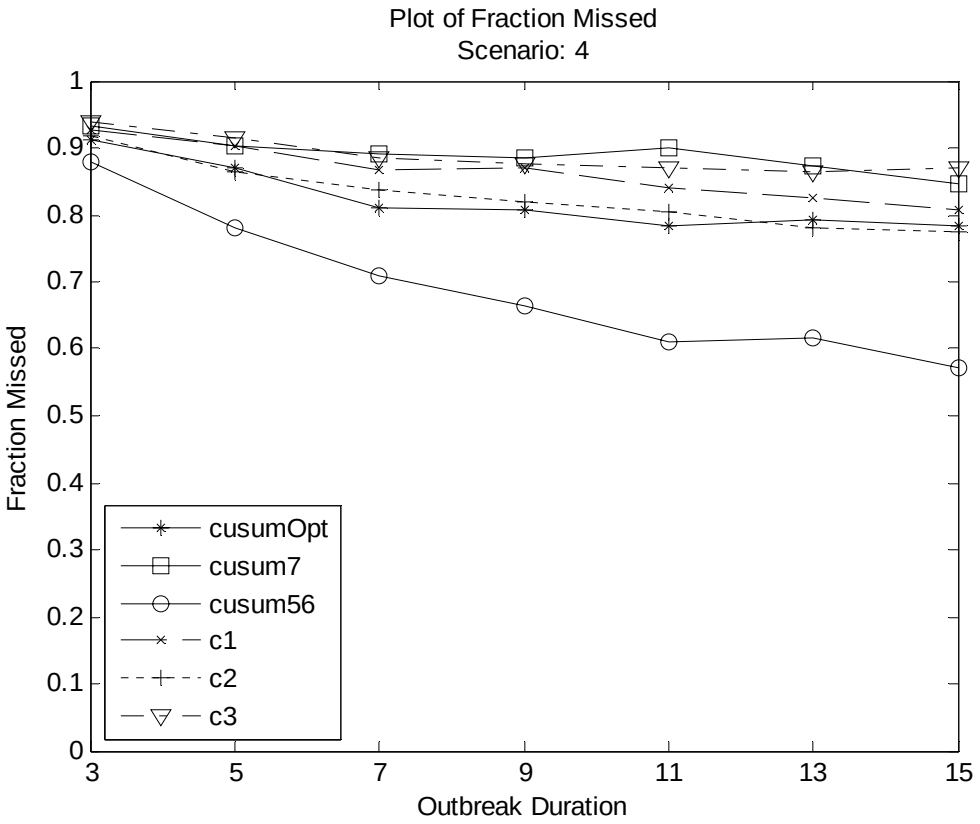

Plot of ATFS Given True Signal
Scenario: 2
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

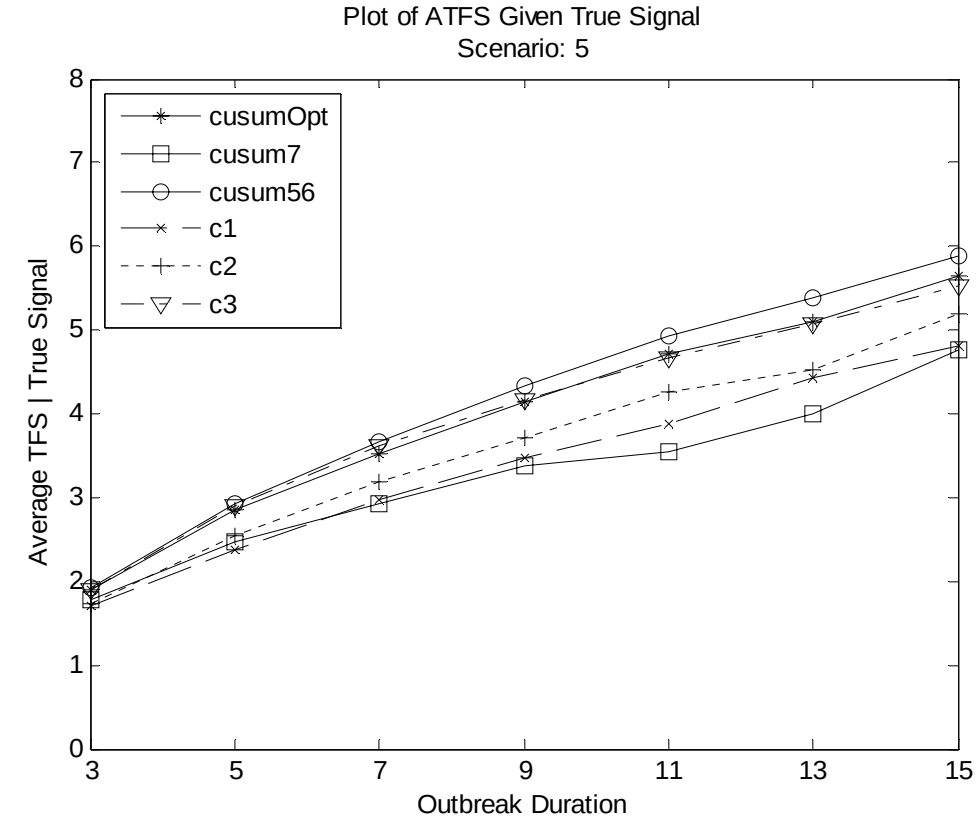

Plot of Fraction Missed
Scenario: 2
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

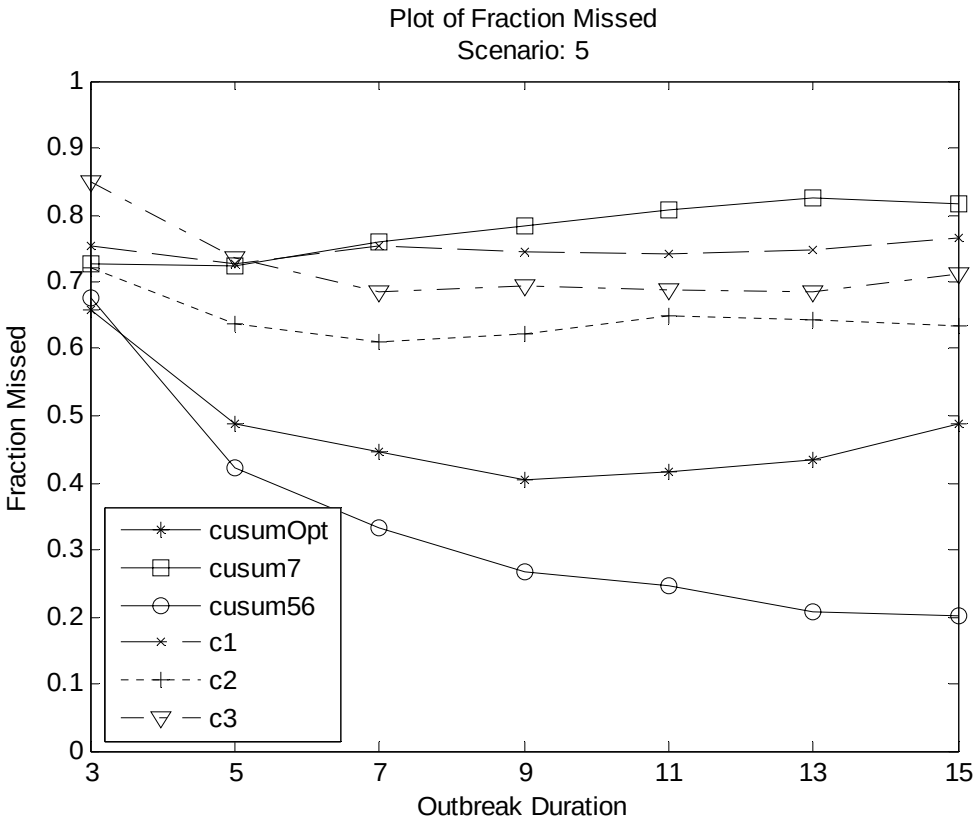

Plot of ATFS Given True Signal
Scenario: 3
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

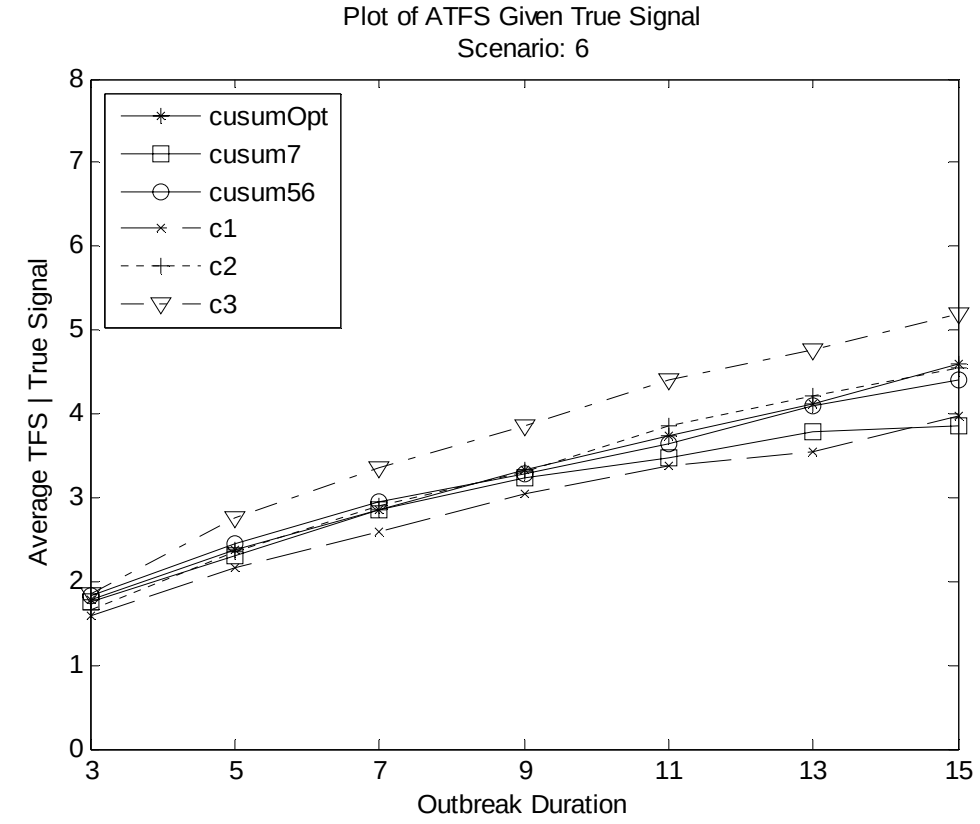

Plot of Fraction Missed
Scenario: 3
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

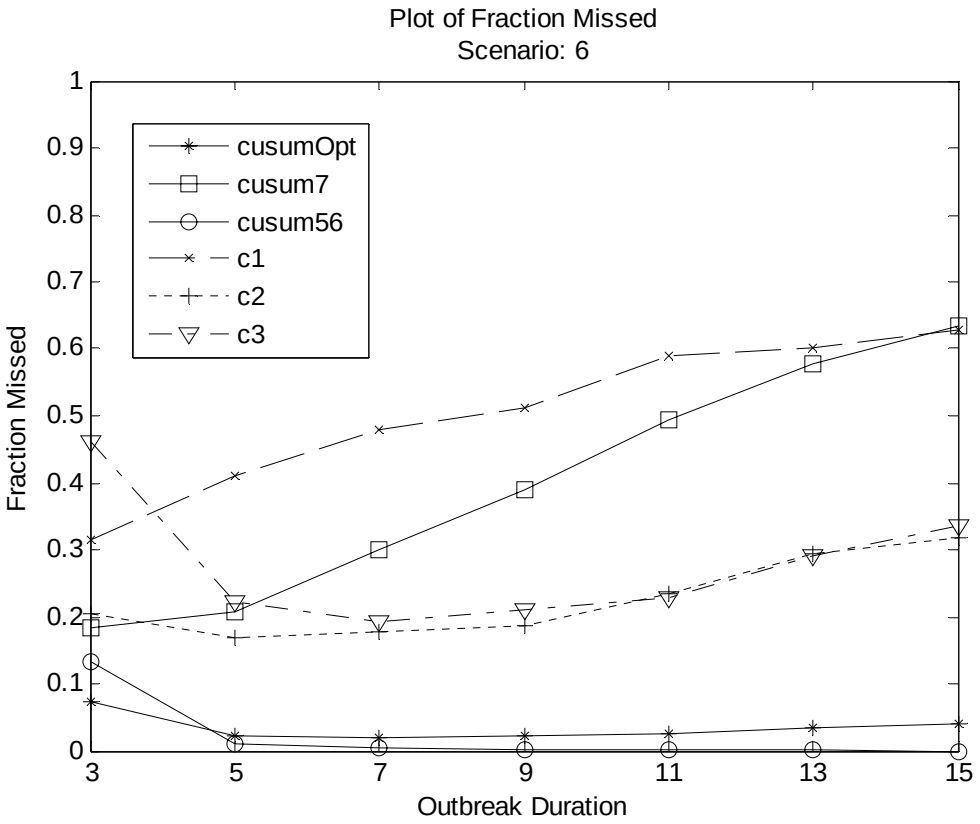

Plot of ATFS Given True Signal
Scenario: 4
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

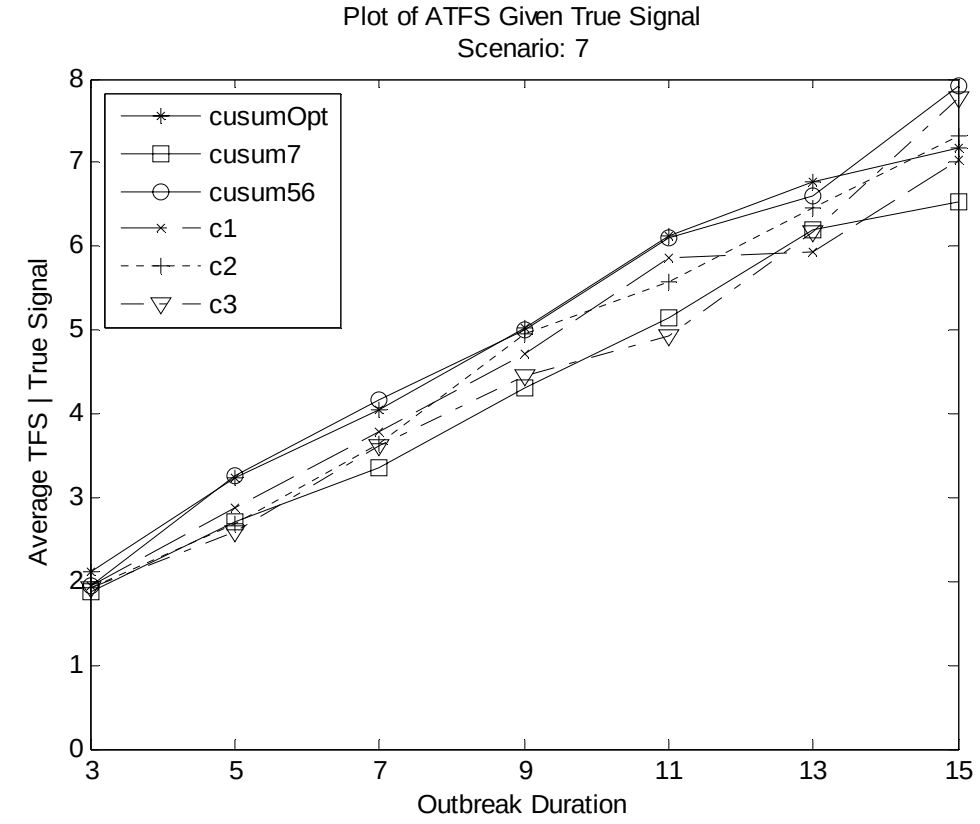

Plot of Fraction Missed
Scenario: 4
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

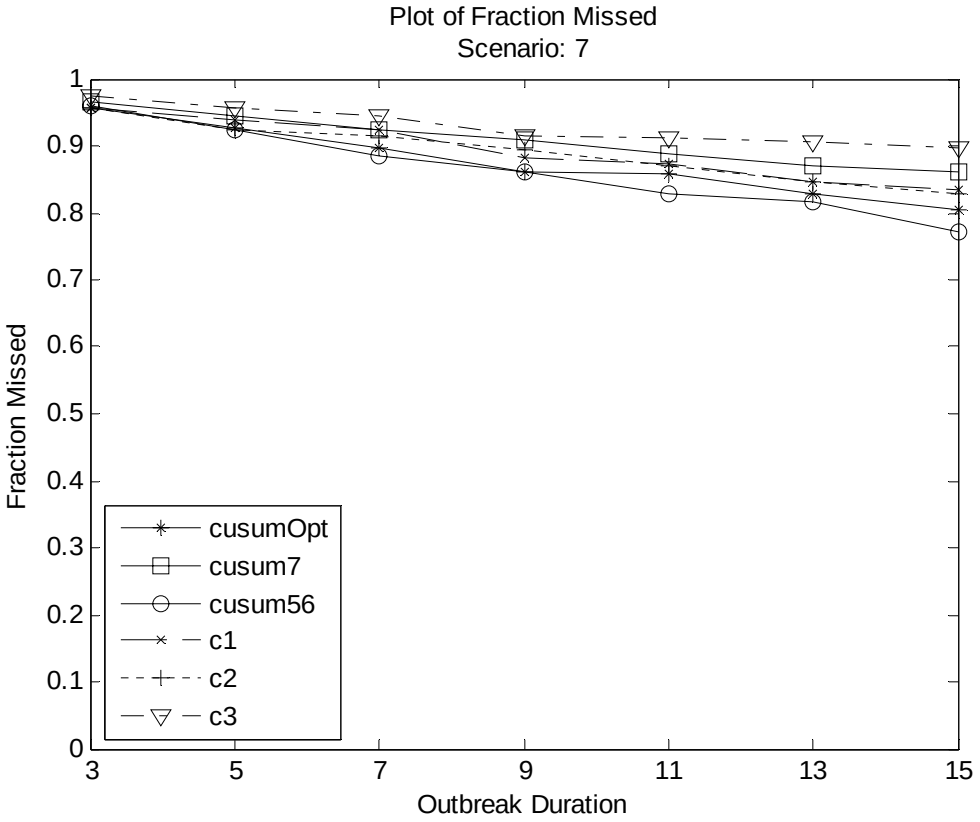

Plot of ATFS Given True Signal
Scenario: 5
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

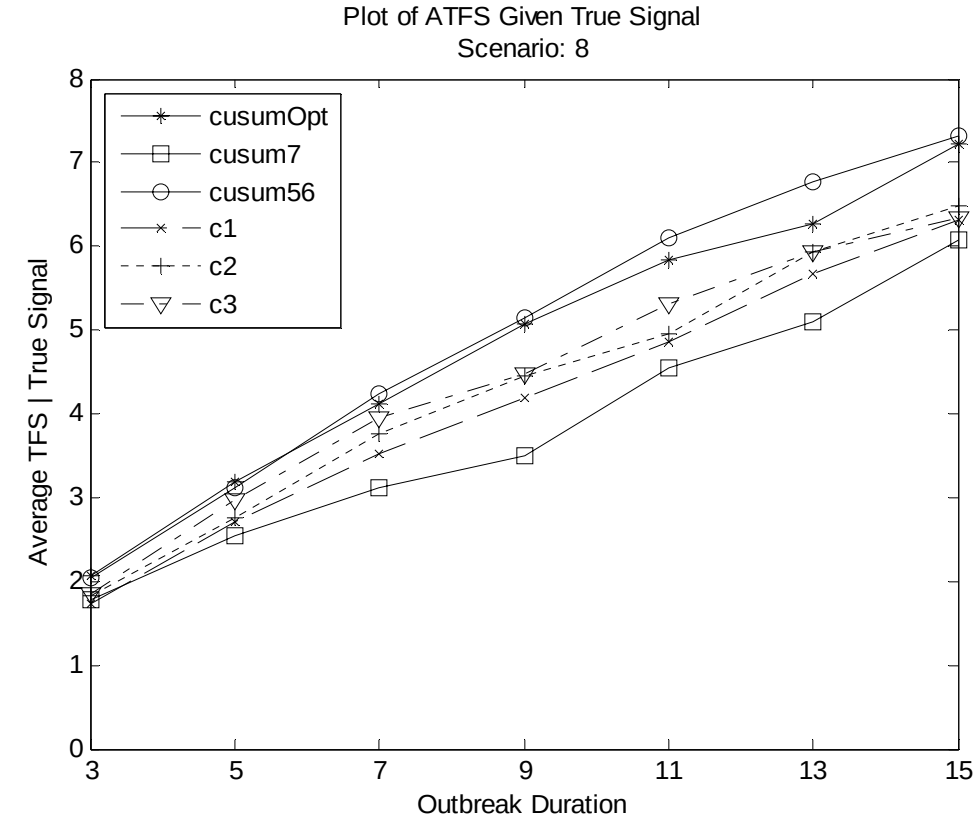

Plot of Fraction Missed
Scenario: 5
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

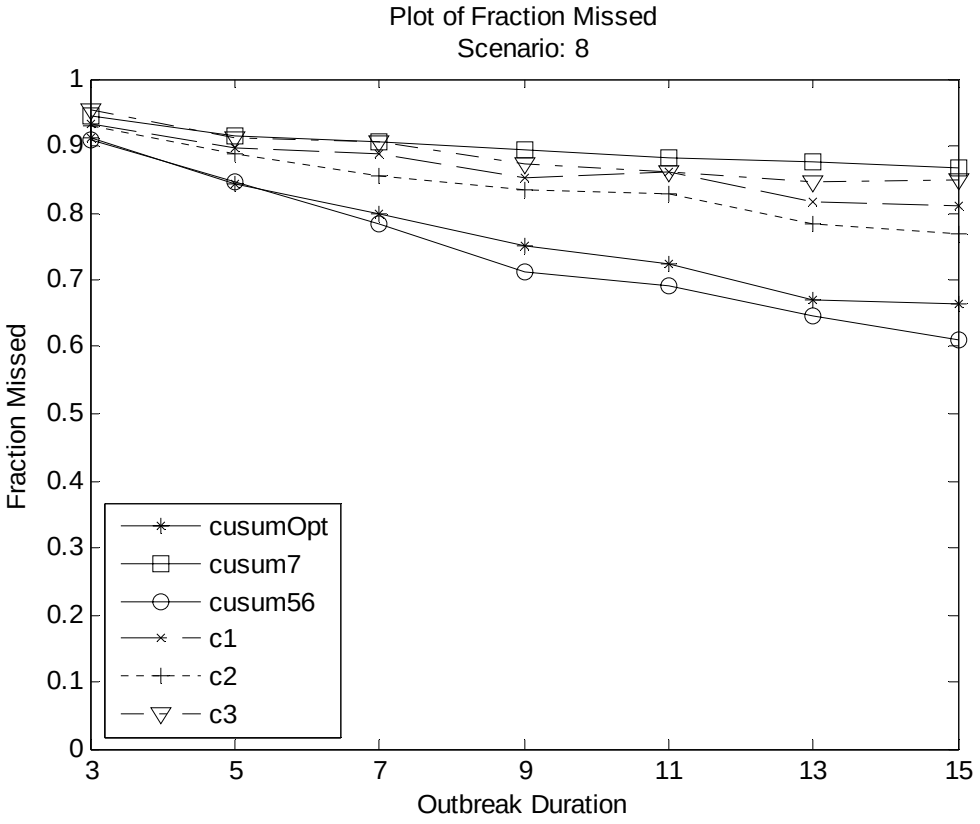

Plot of ATFS Given True Signal
Scenario: 6
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

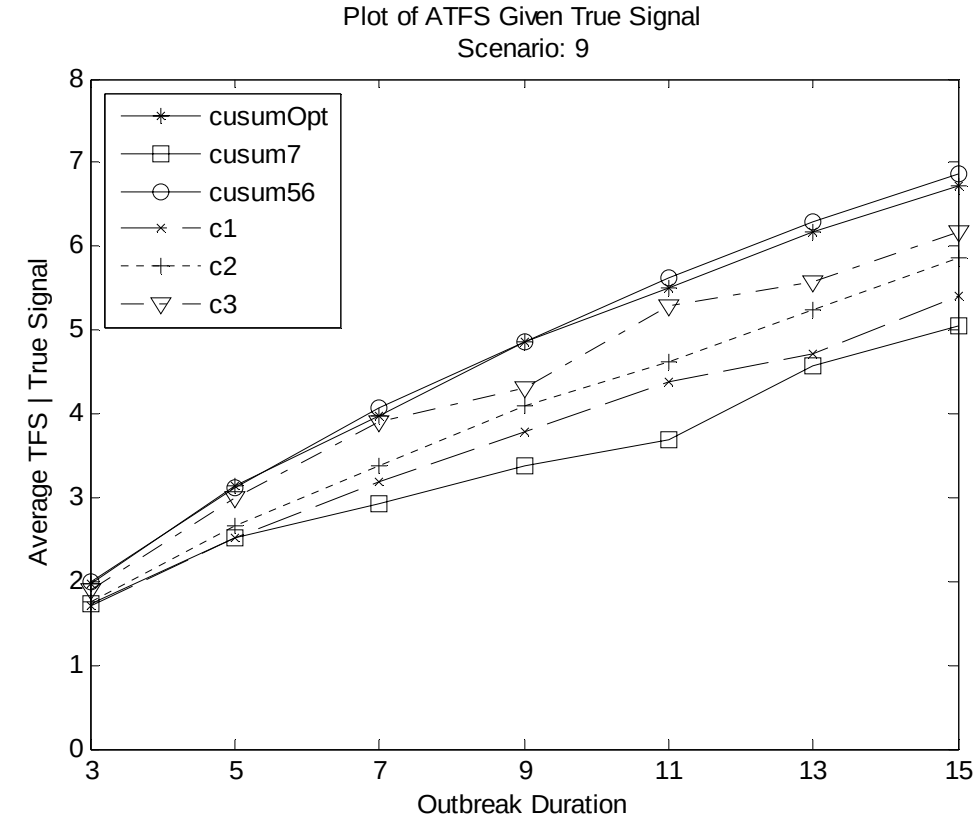

Plot of Fraction Missed
Scenario: 6
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

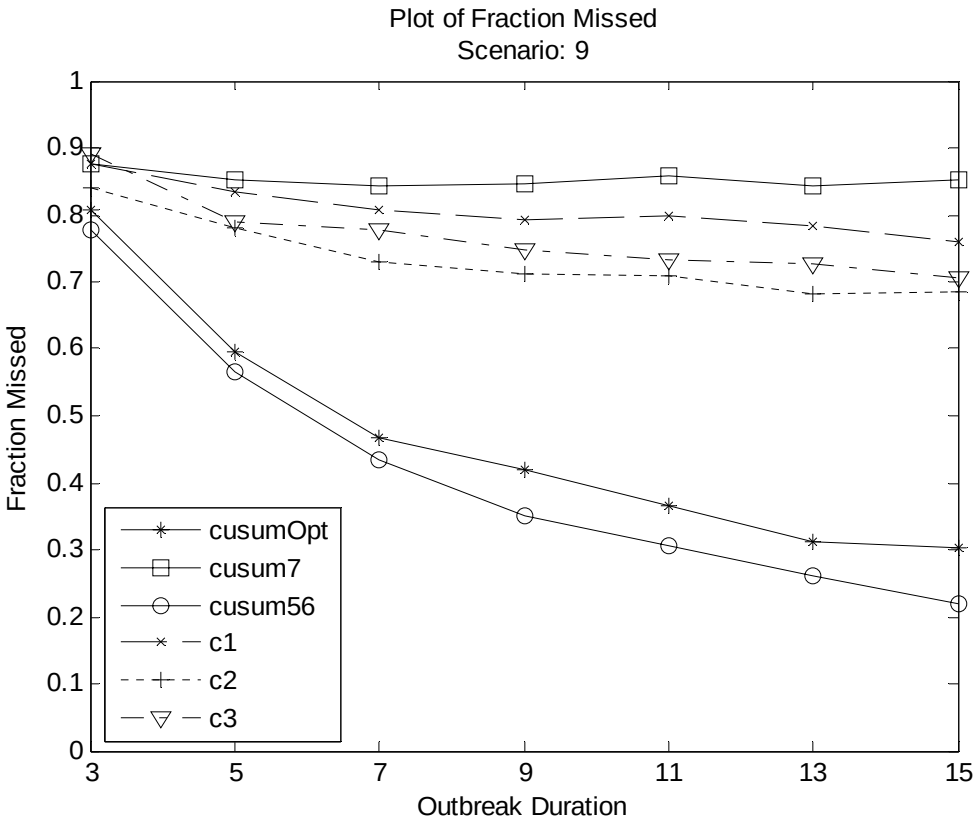

Plot of ATFS Given True Signal
Scenario: 7
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

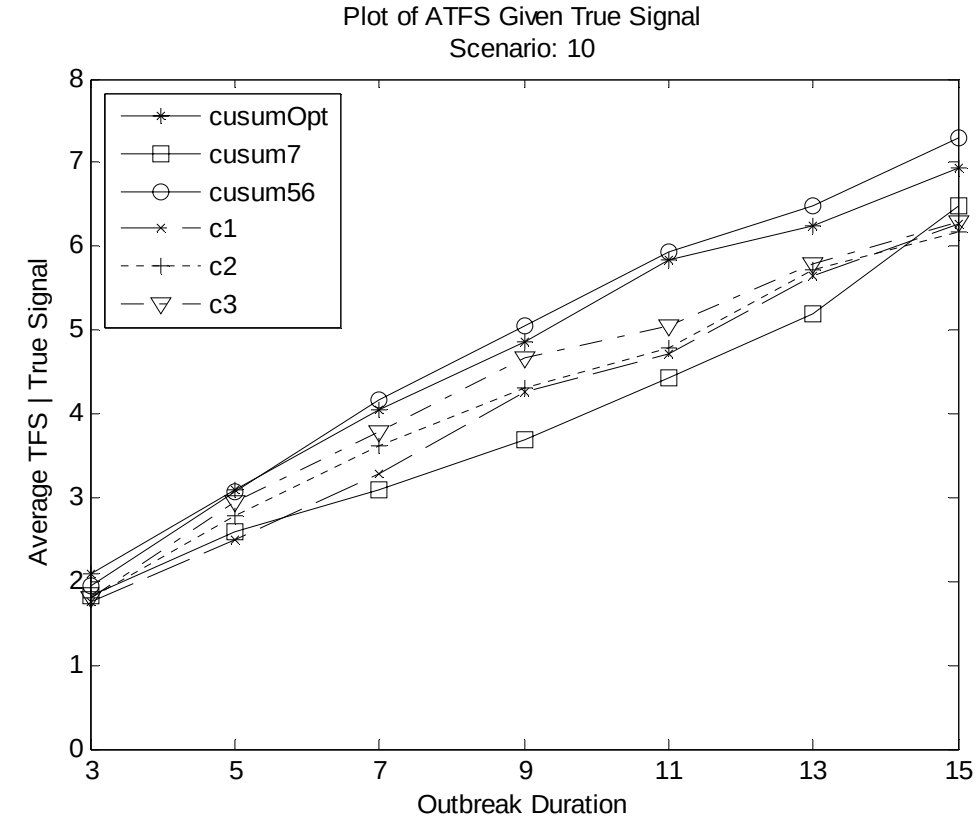

Plot of Fraction Missed
Scenario: 7
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

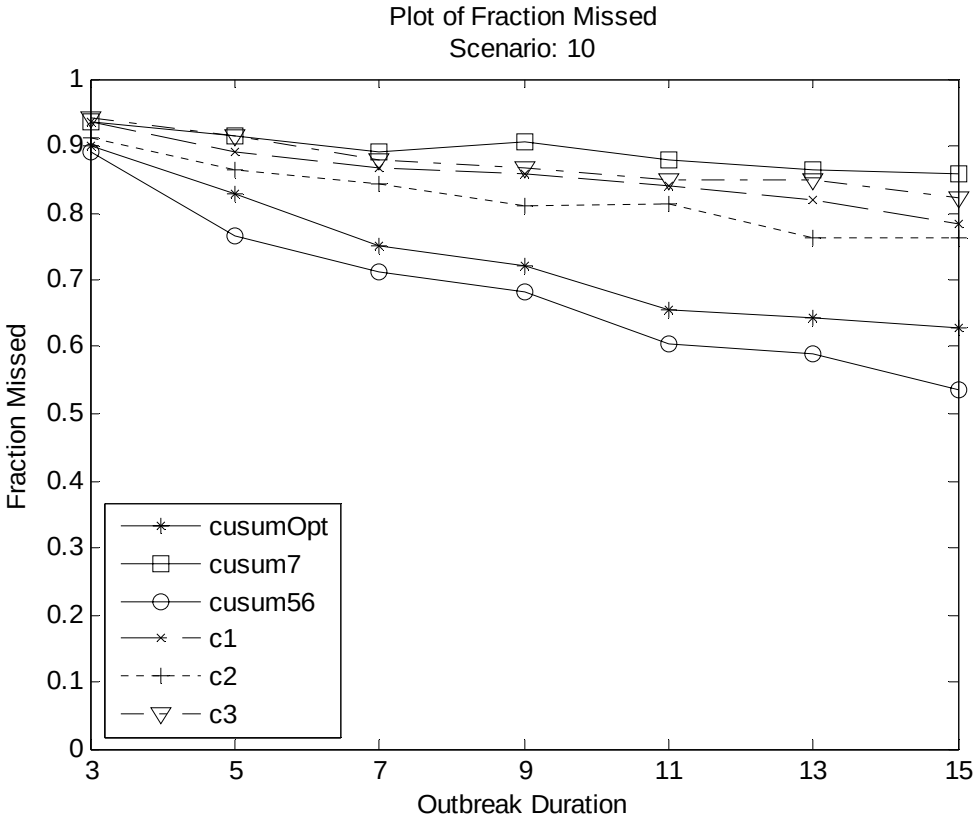

Plot of ATFS Given True Signal
Scenario: 8
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

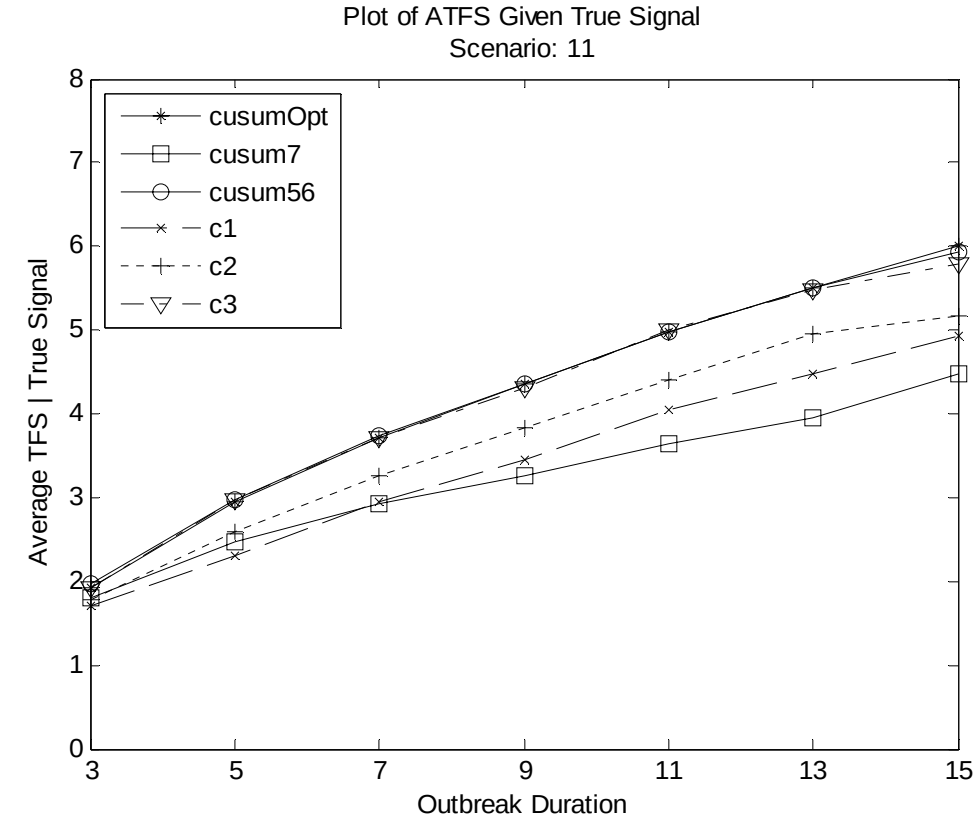

Plot of Fraction Missed
Scenario: 8
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

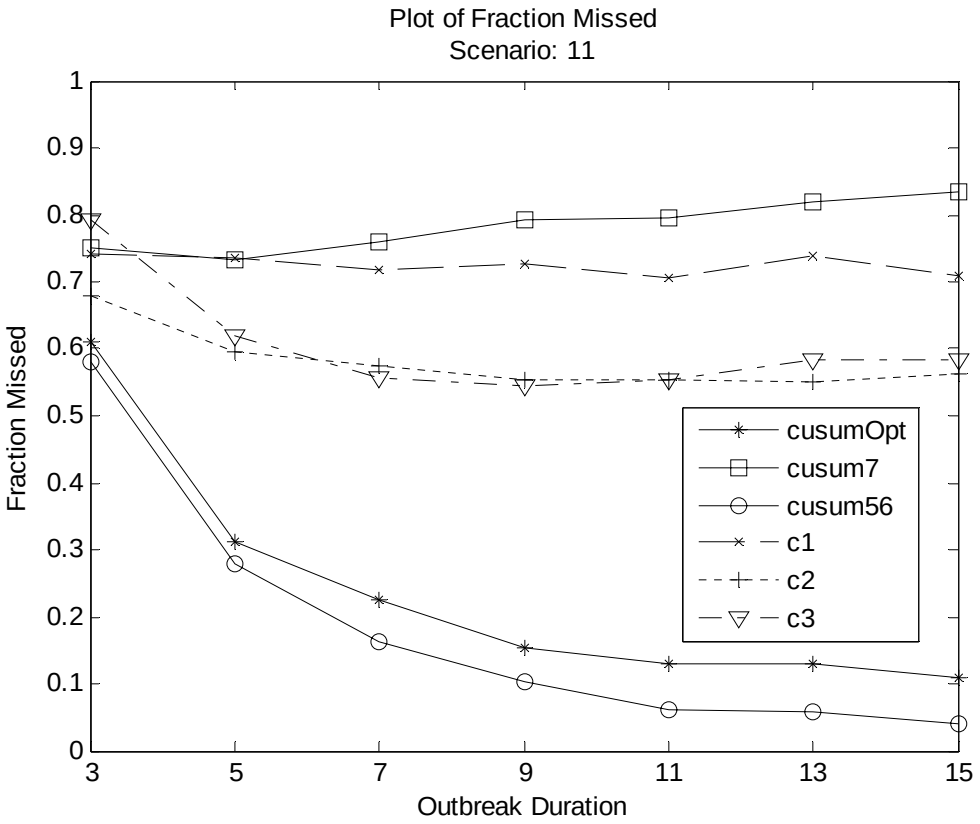

Plot of ATFS Given True Signal
Scenario: 9
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

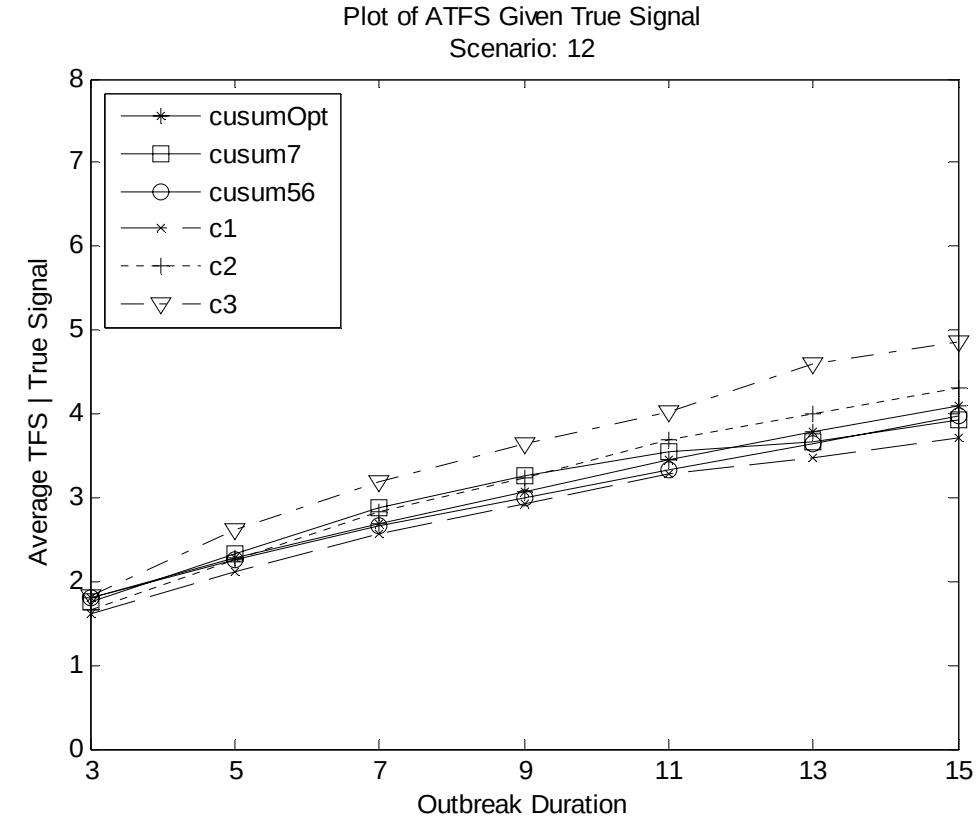

Plot of Fraction Missed
Scenario: 9
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

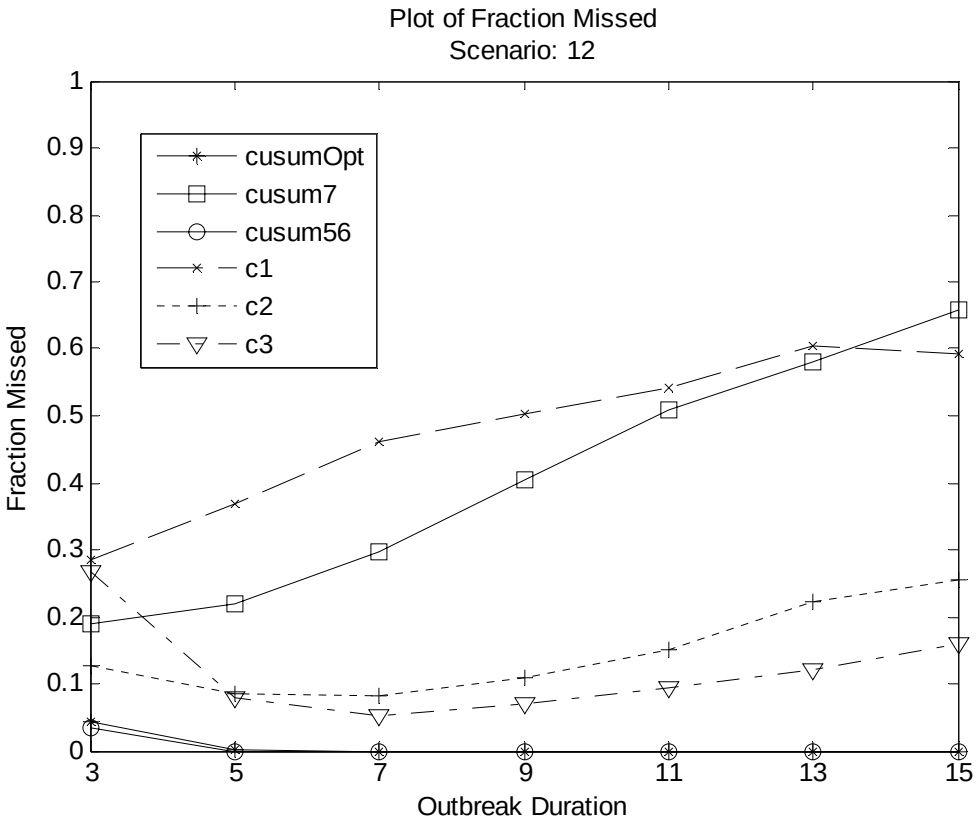

Plot of ATFS Given True Signal
Scenario: 10
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

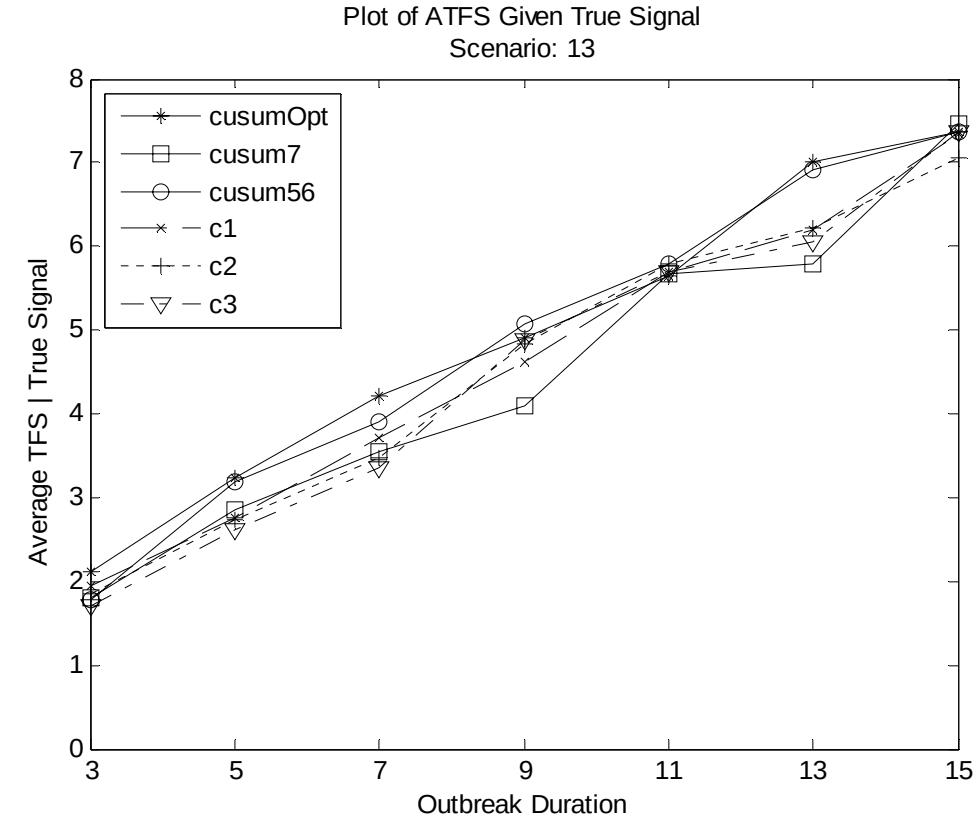

Plot of Fraction Missed
Scenario: 10
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

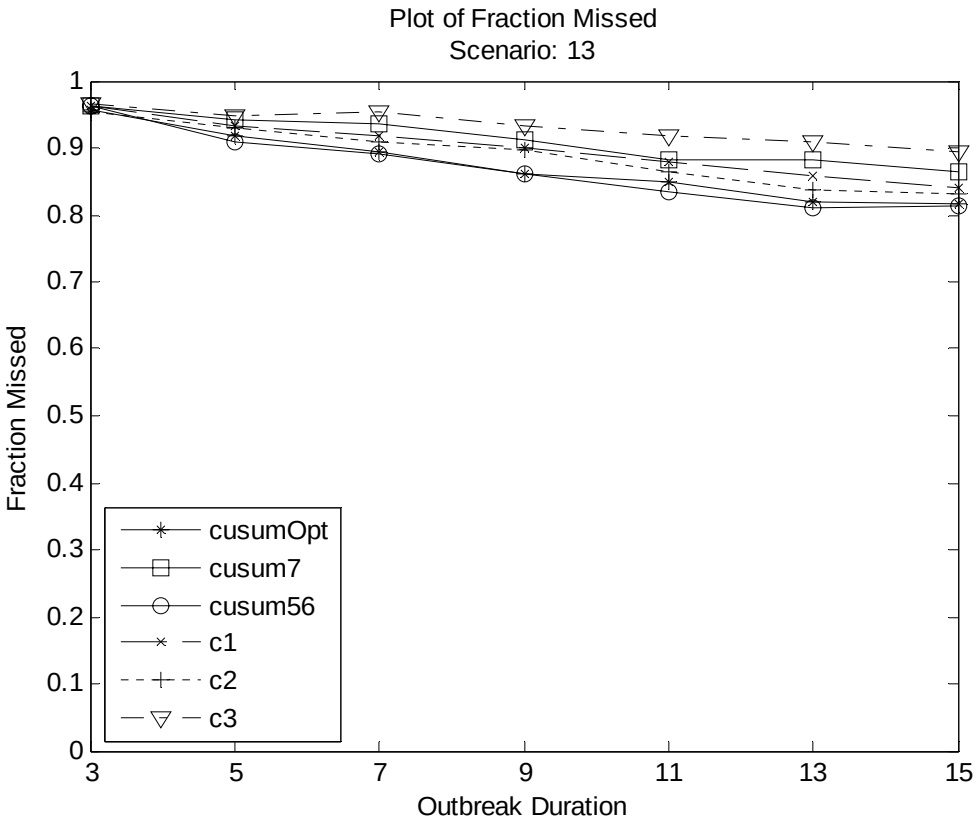

Plot of ATFS Given True Signal
Scenario: 11
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

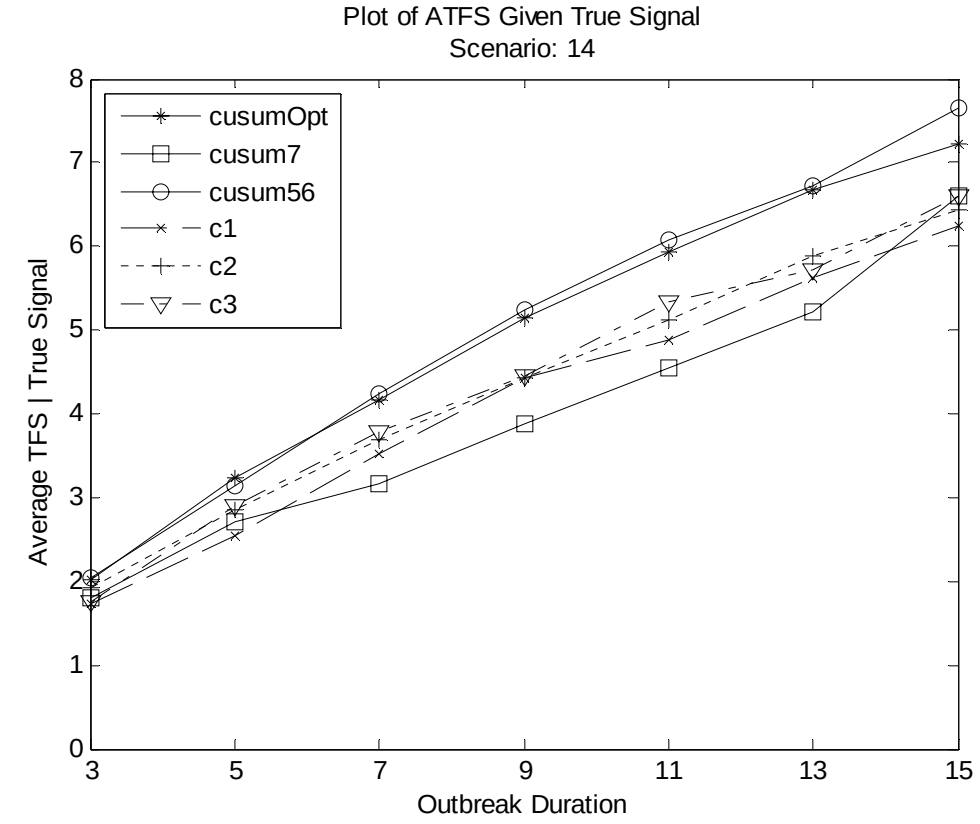

Plot of Fraction Missed
Scenario: 11
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

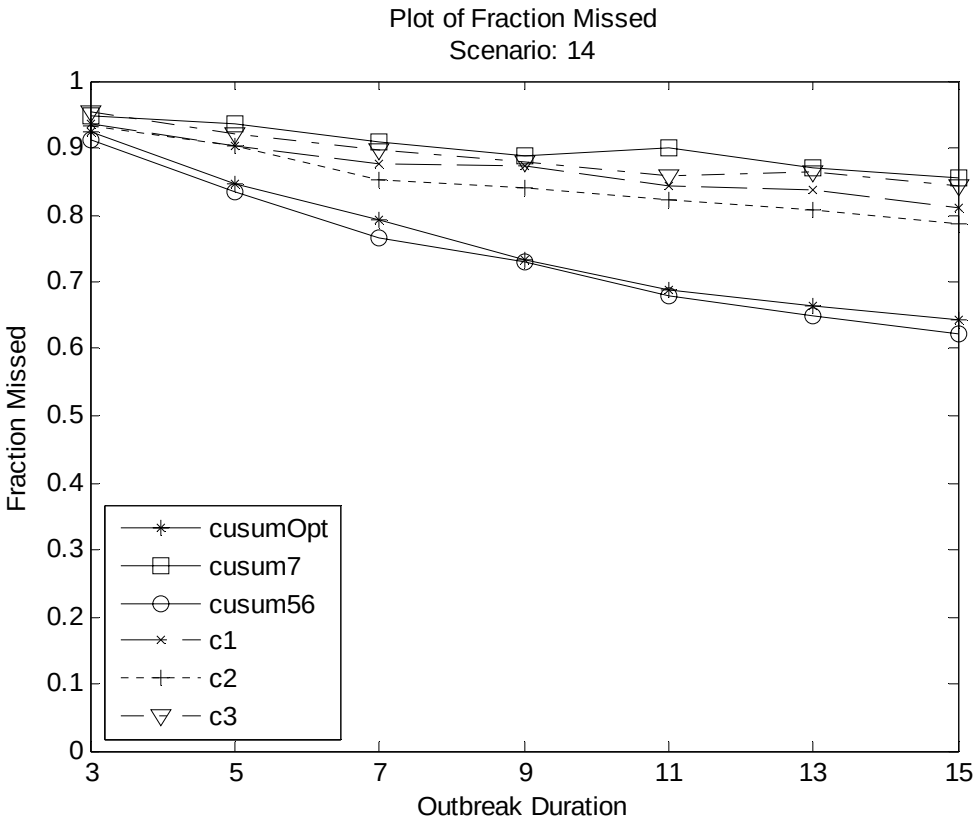

Plot of ATFS Given True Signal
Scenario: 12
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

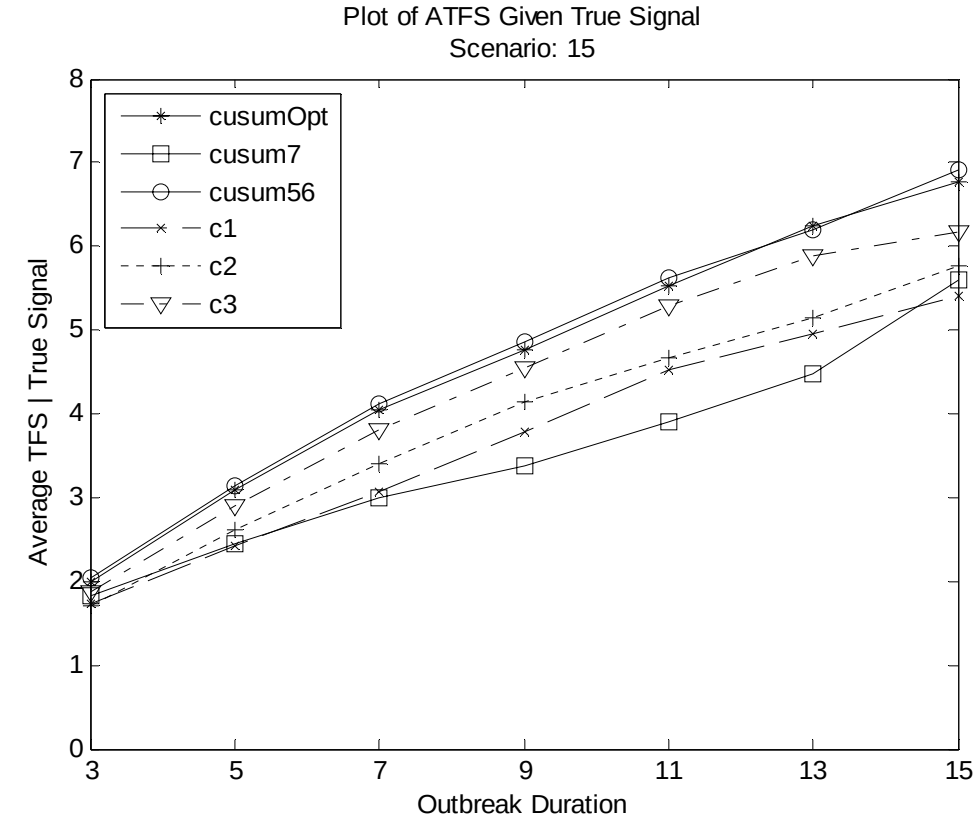

Plot of Fraction Missed
Scenario: 12
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

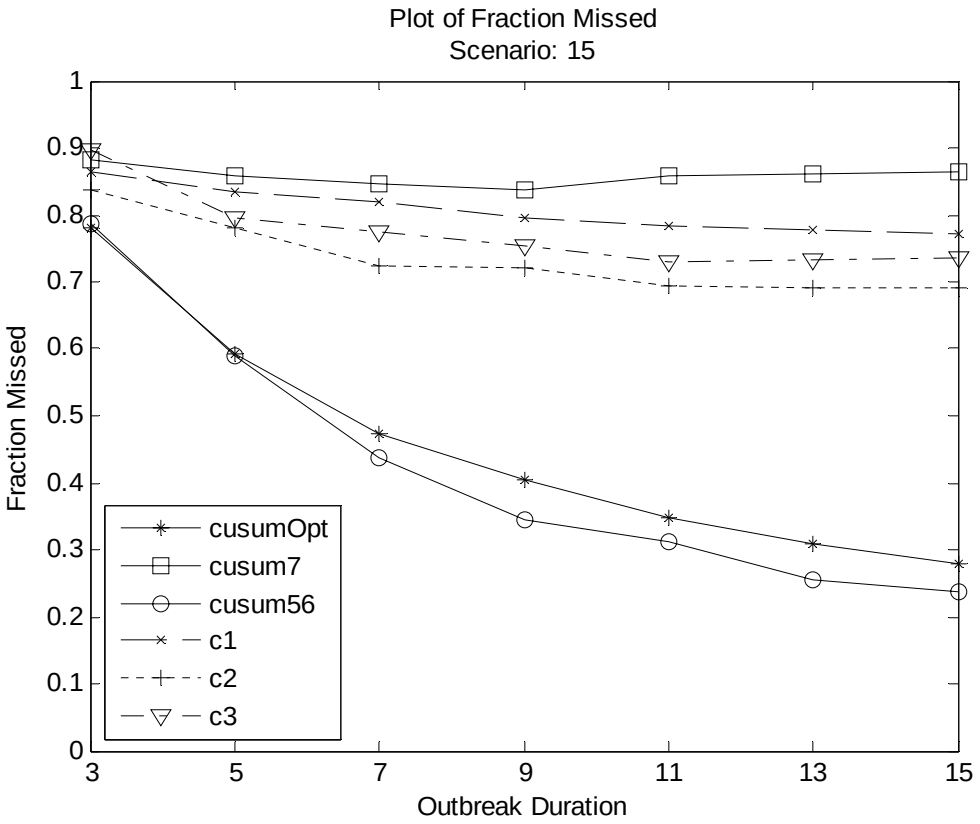

Plot of ATFS Given True Signal
Scenario: 13
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

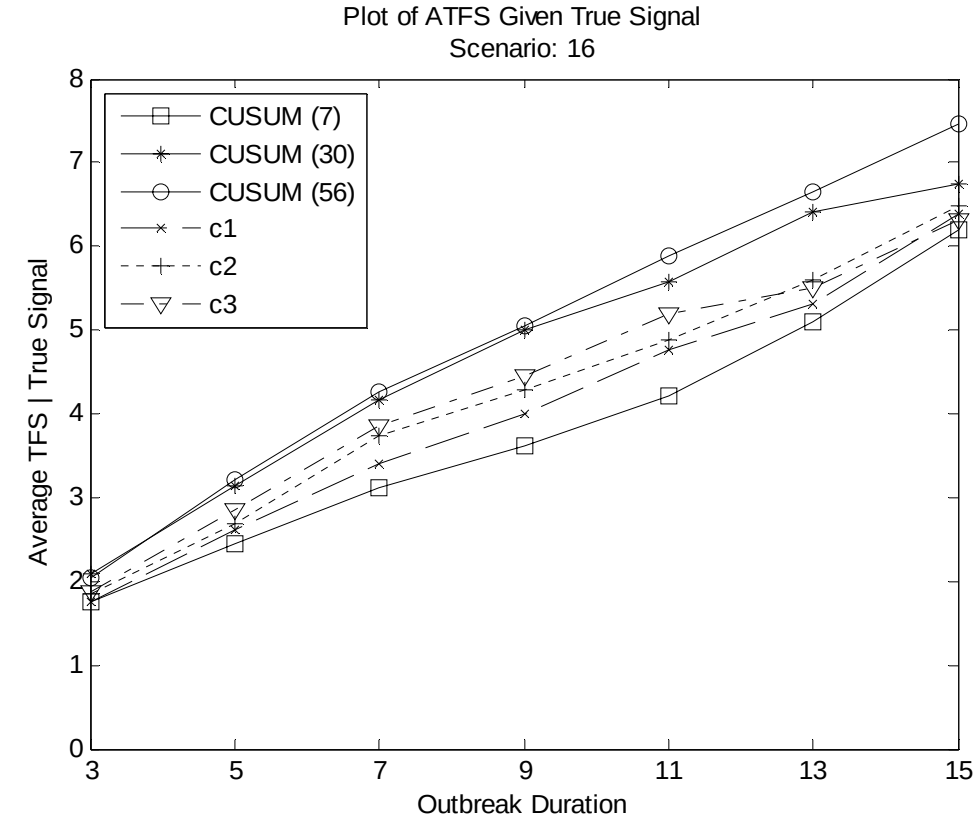

Plot of Fraction Missed
Scenario: 13
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

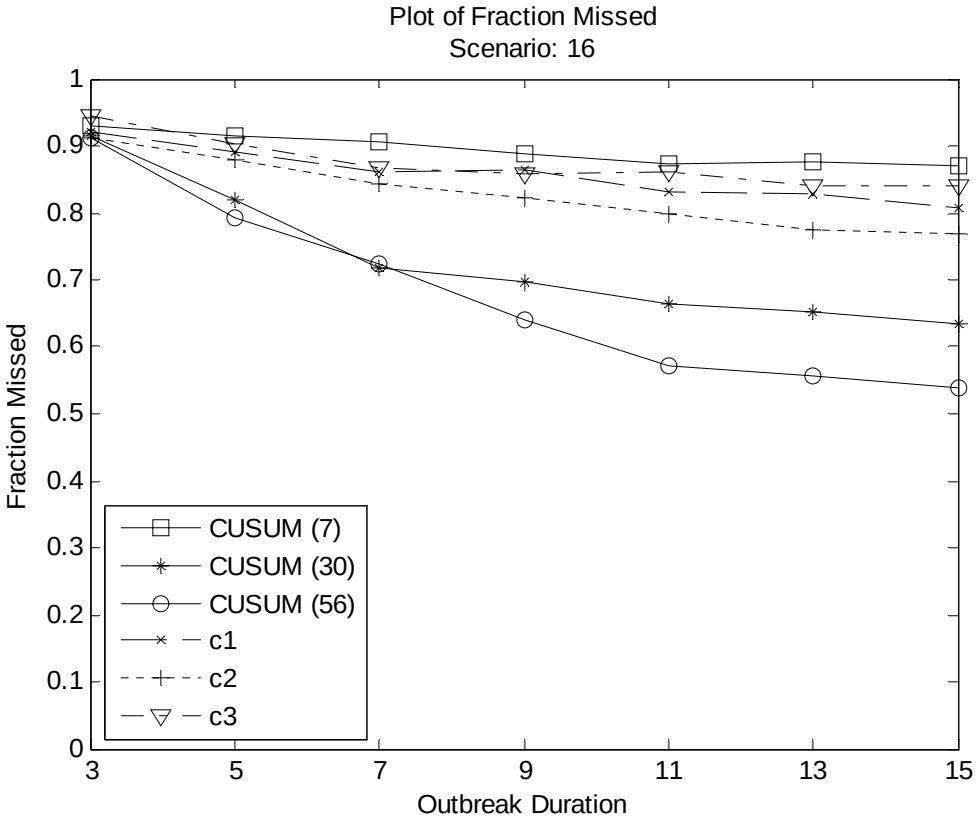

Plot of ATFS Given True Signal
Scenario: 14
Average TFS | True Signal
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

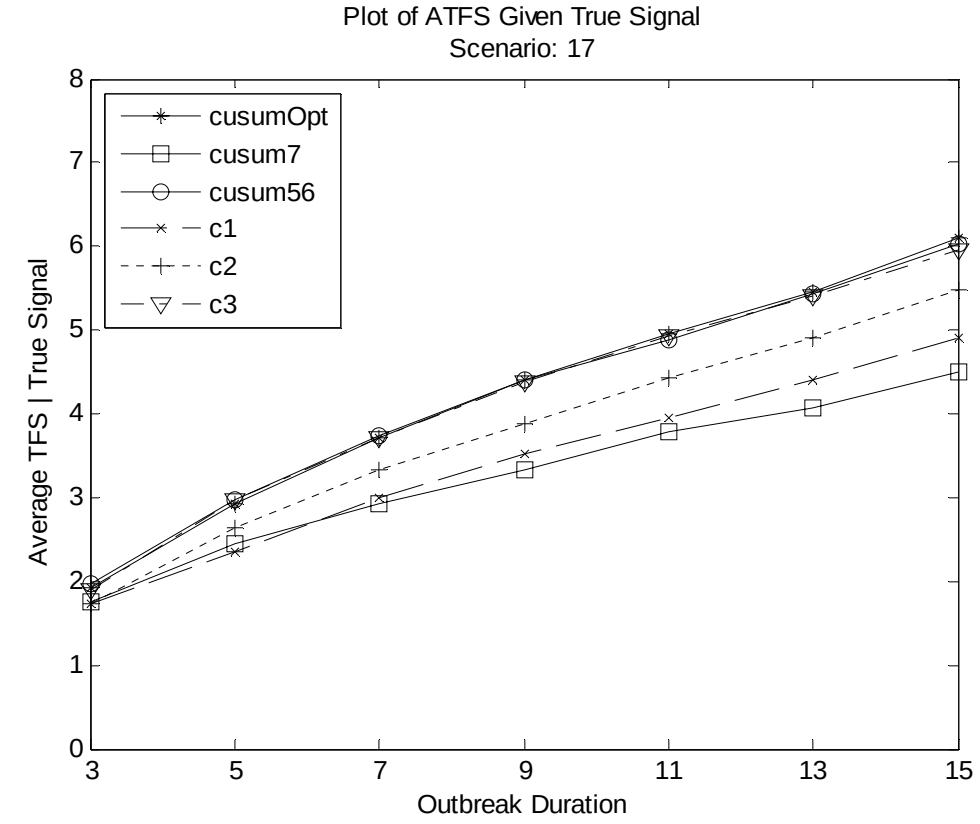

Plot of Fraction Missed
Scenario: 14
Fraction Missed
Outbreak Duration
cusumOpt
cusum7
cusum56
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

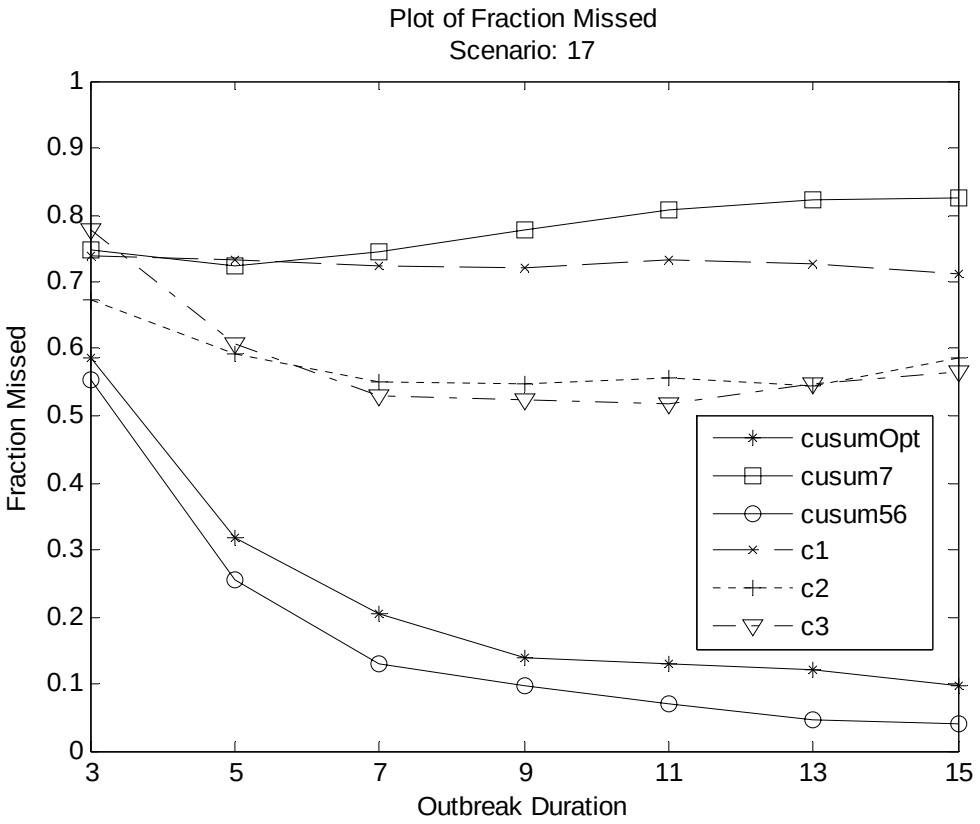

Plot of ATFS Given True Signal
Scenario: 15
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

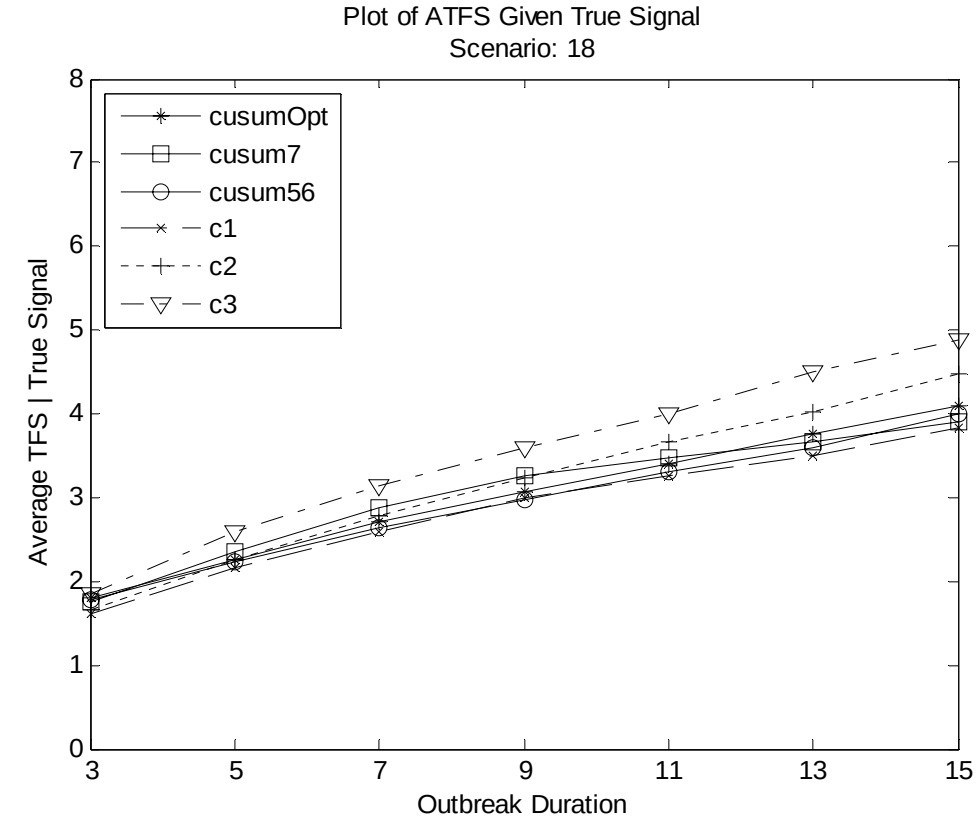

Plot of Fraction Missed
Scenario: 15
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

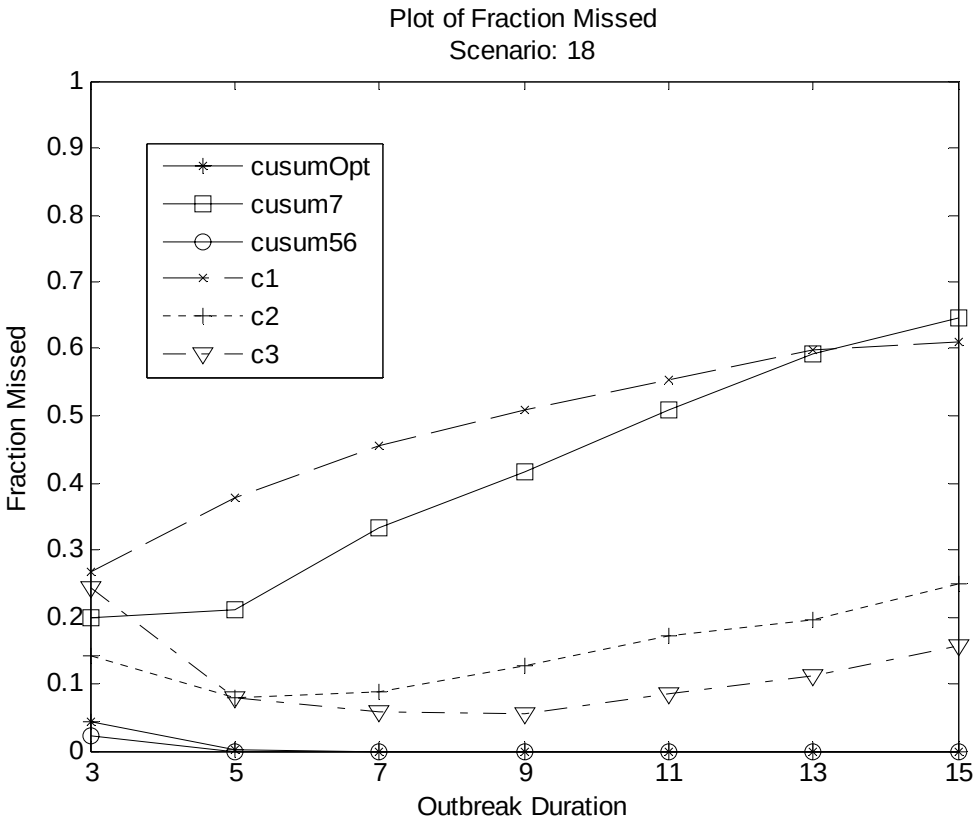

Plot of ATFS Given True Signal
Scenario: 16
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

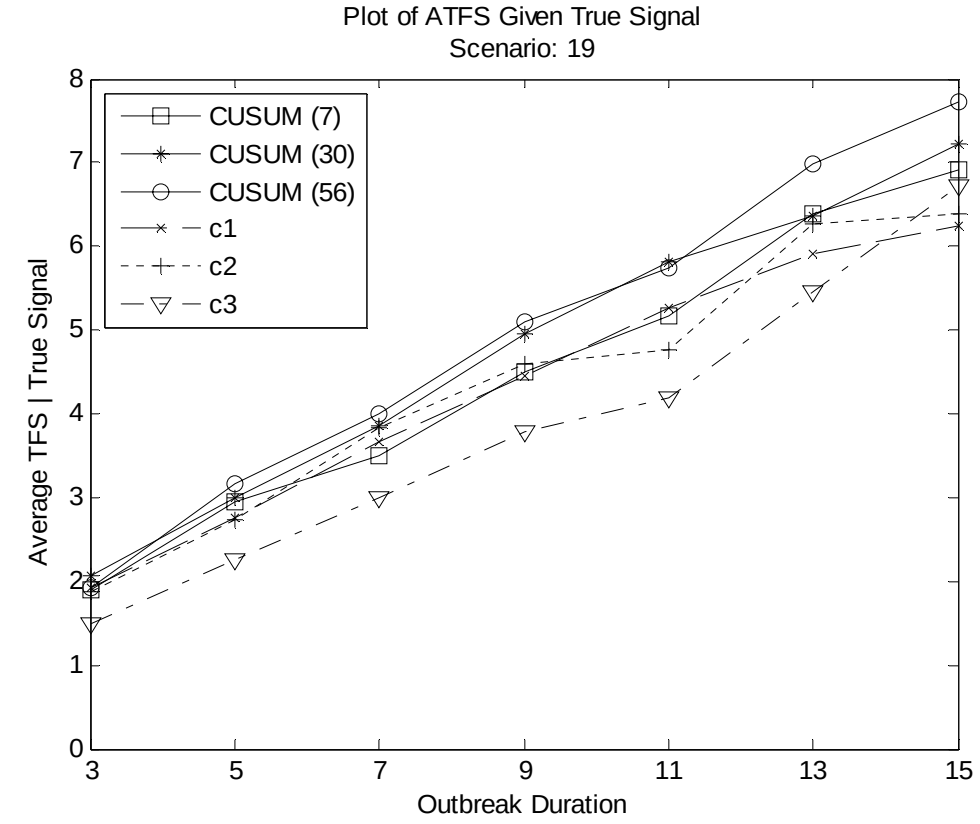

Plot of Fraction Missed
Scenario: 16
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

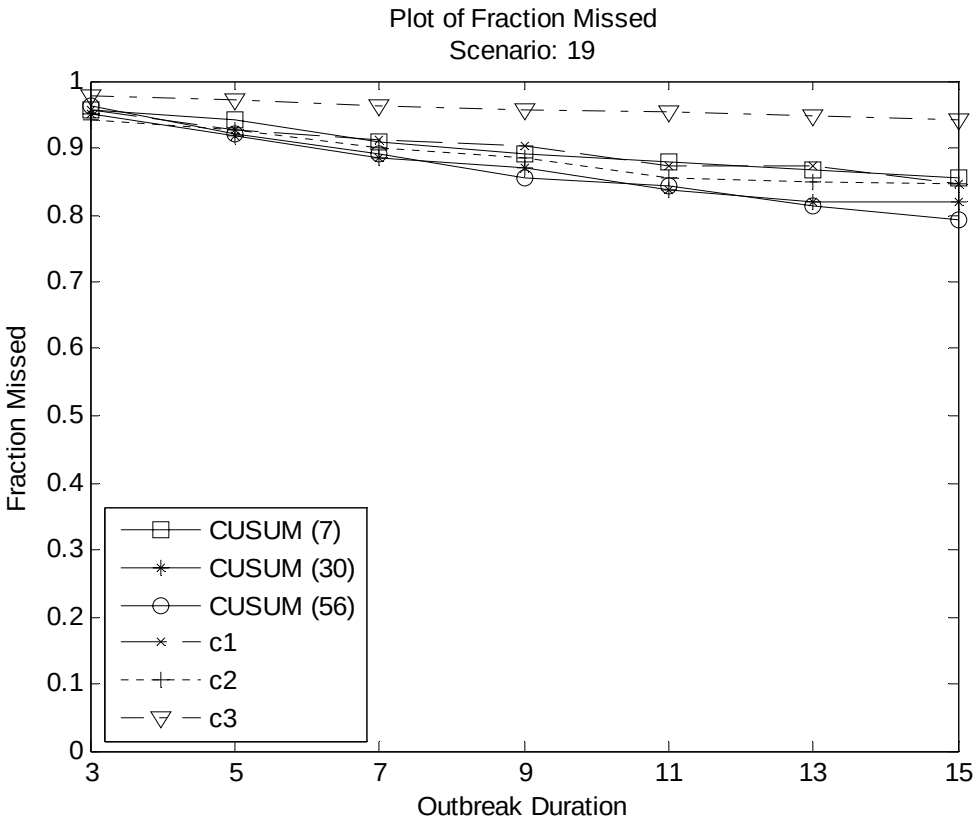

Plot of ATFS Given True Signal
Scenario: 17
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

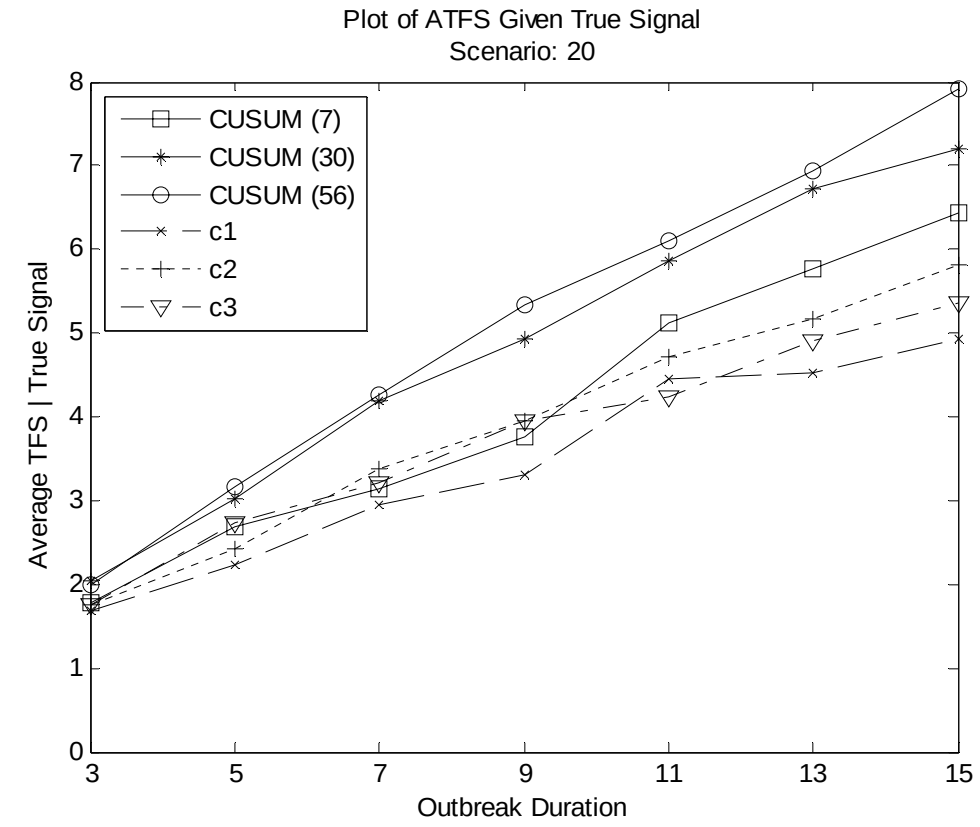

Plot of Fraction Missed
Scenario: 17
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

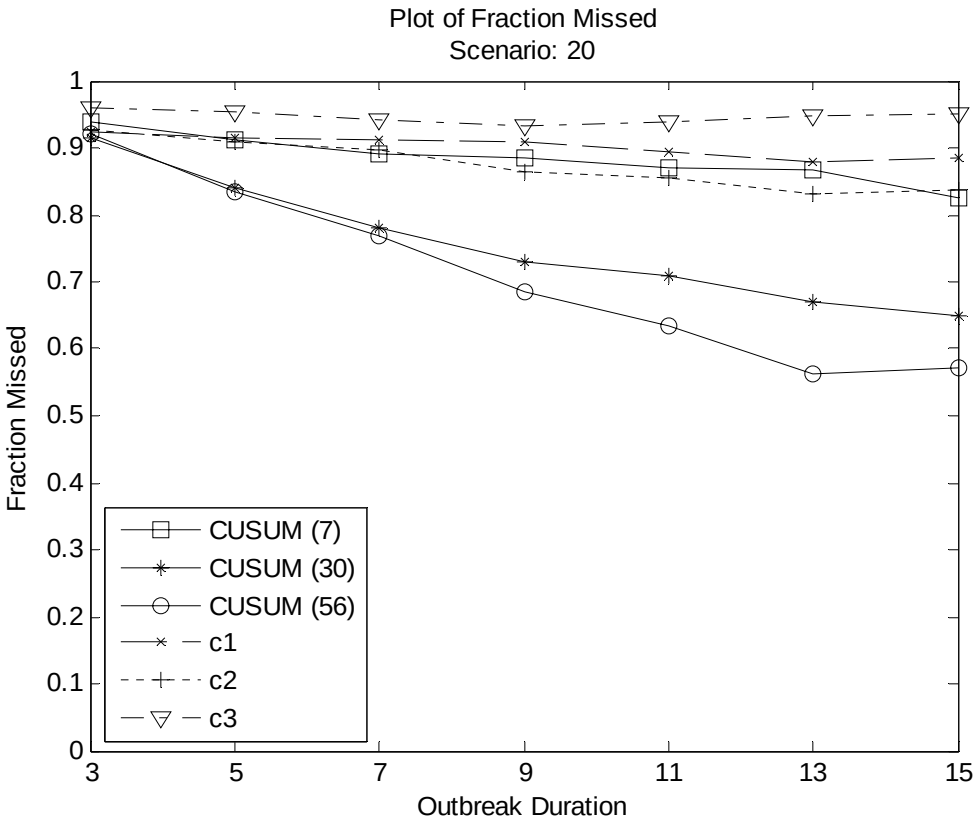

Plot of ATFS Given True Signal
Scenario: 18
cusumOpt
cusum7
cusum56
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

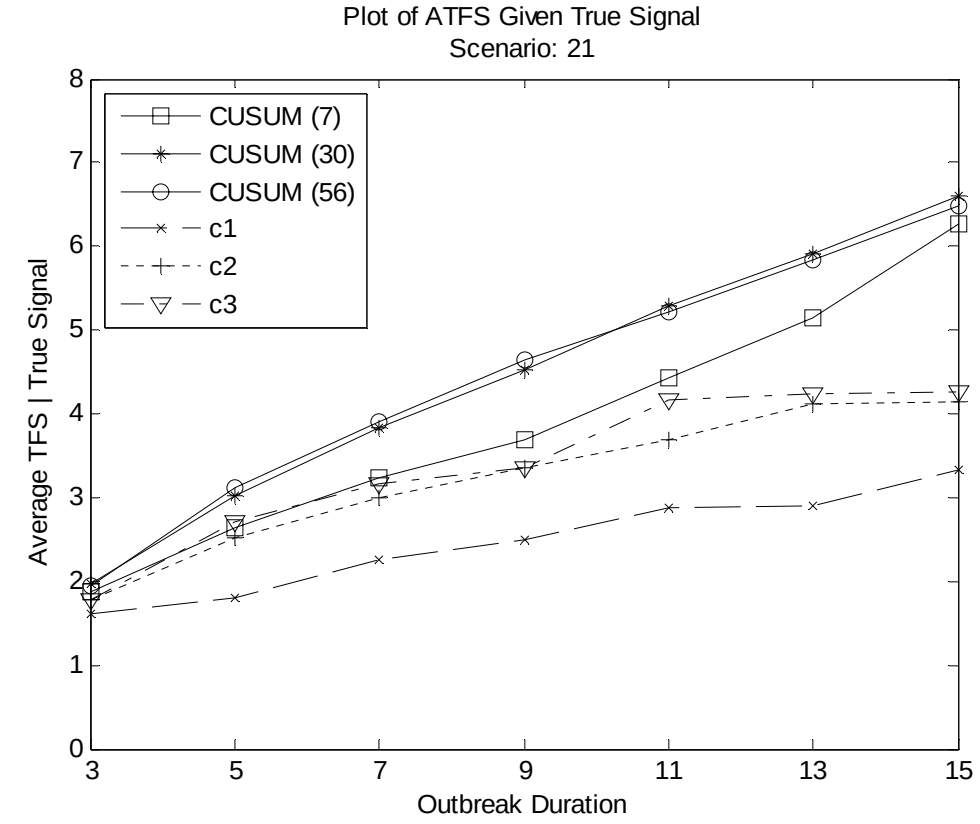

Plot of Fraction Missed
Scenario: 18
cusumOpt
cusum7
cusum56
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

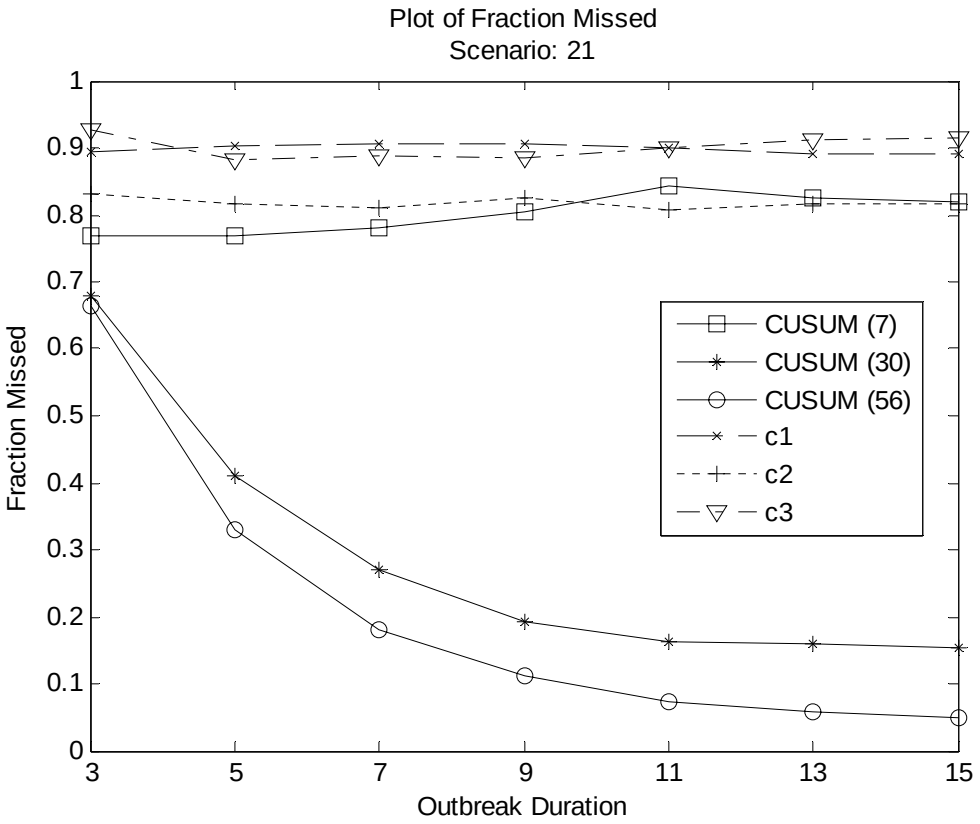

Plot of ATFS Given True Signal
Scenario: 19
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

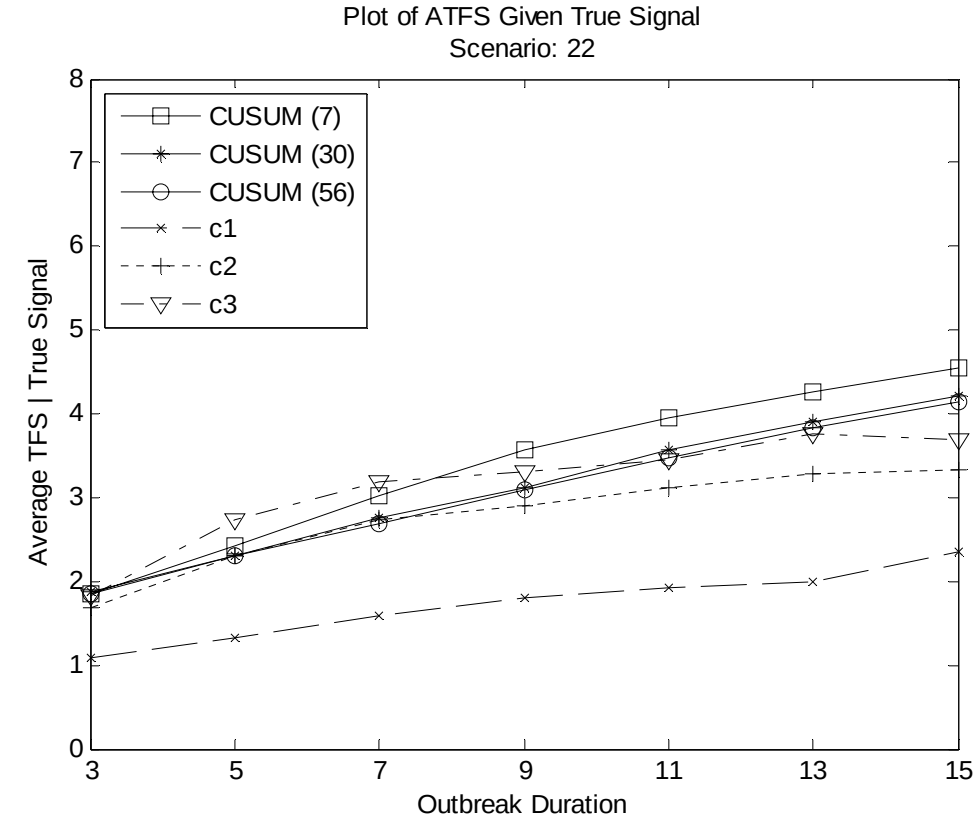

Plot of Fraction Missed
Scenario: 19
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

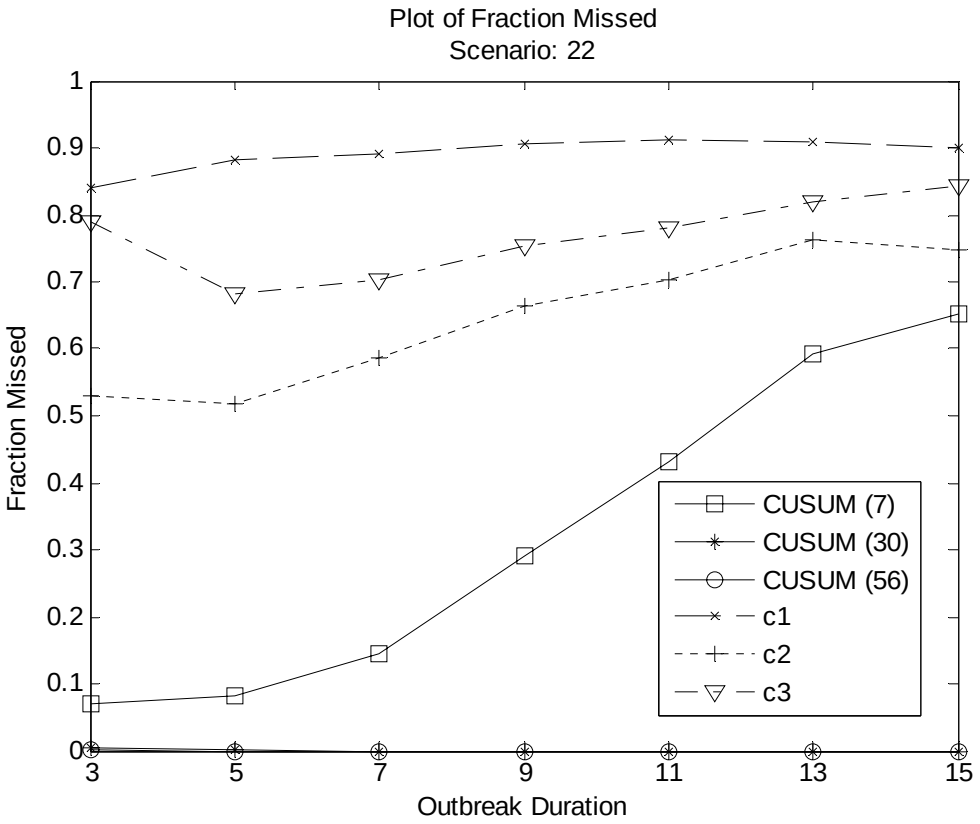

Plot of ATFS Given True Signal
Scenario: 20
Average TFS | True Signal
Outbreak Duration
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

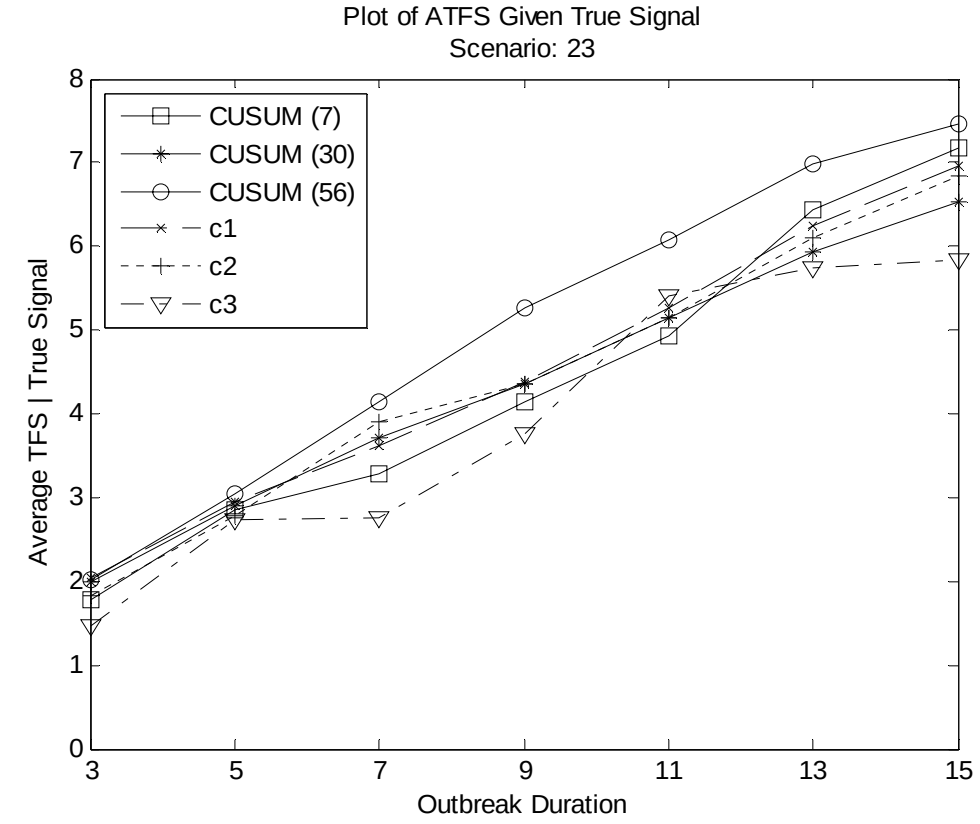

Plot of Fraction Missed
Scenario: 20
Fraction Missed
Outbreak Duration
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

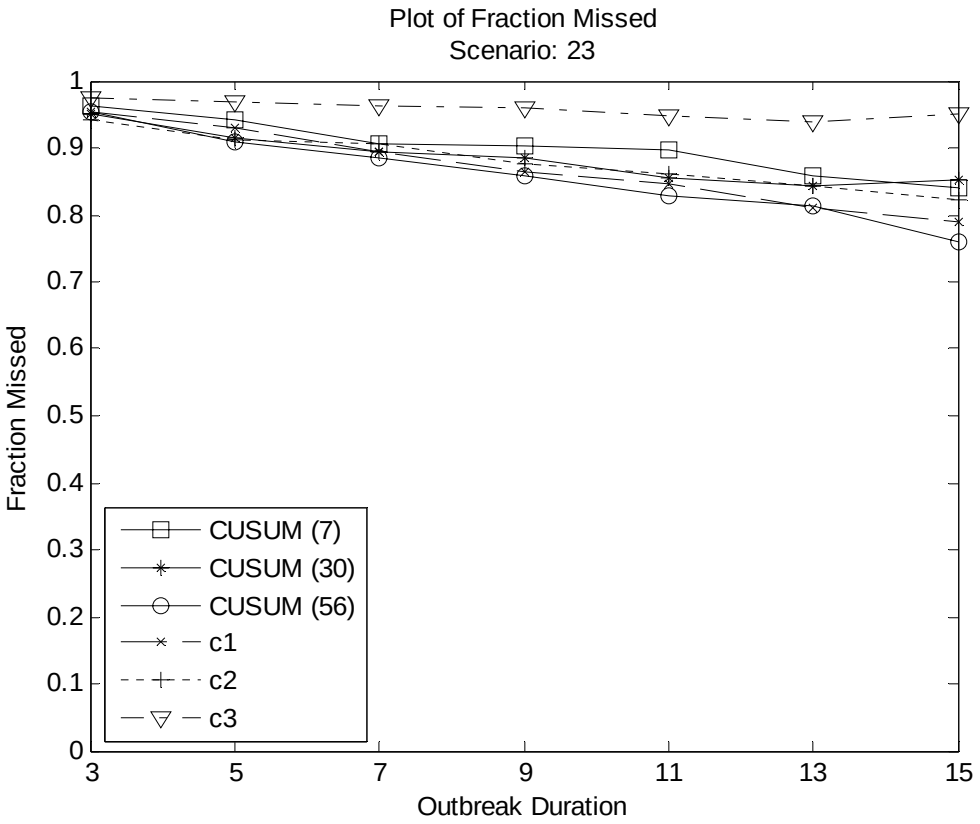

Plot of ATFS Given True Signal
Scenario: 21
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

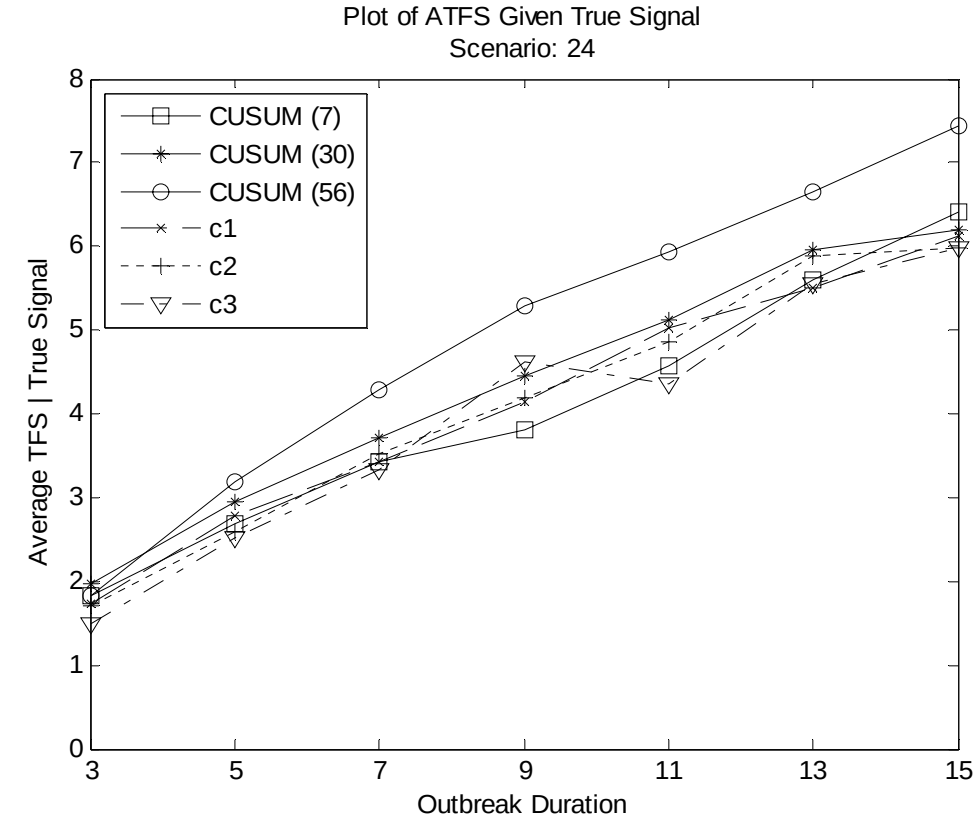

Plot of Fraction Missed
Scenario: 21
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

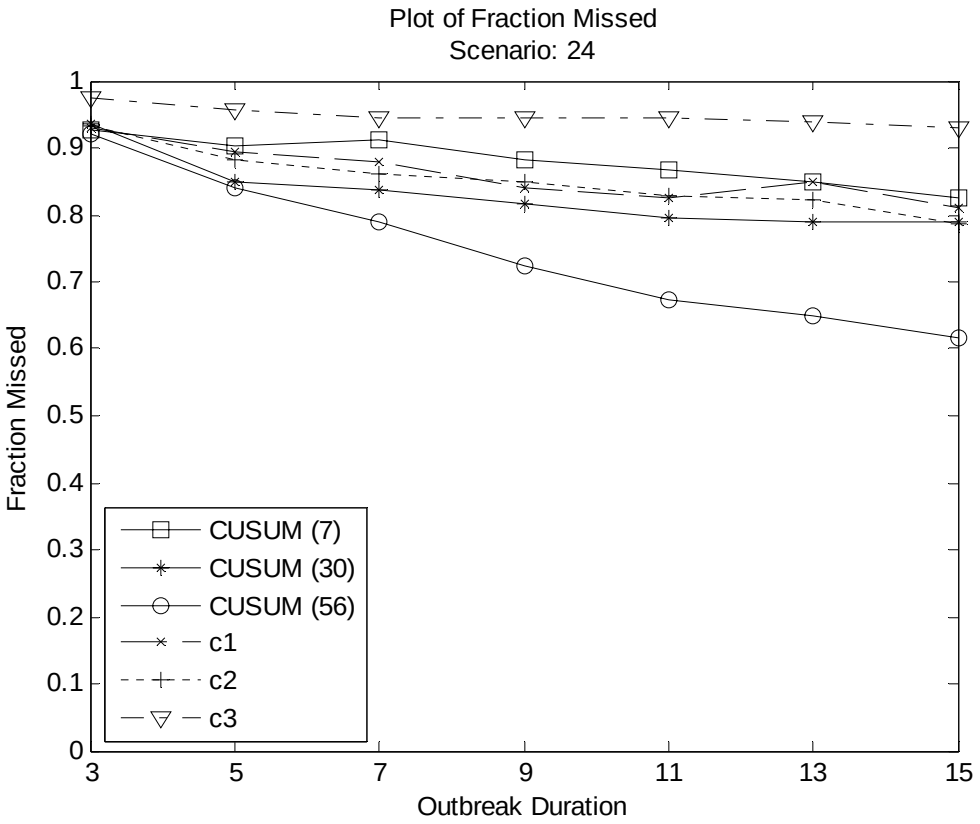

Plot of ATFS Given True Signal
Scenario: 22
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

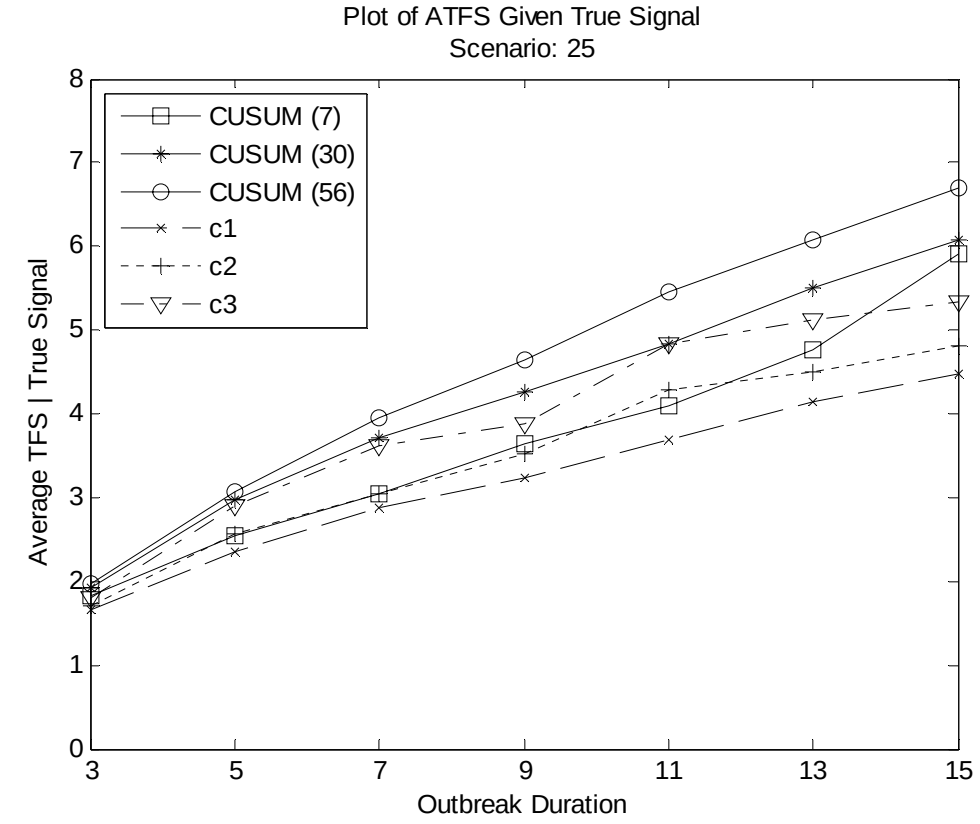

Plot of Fraction Missed
Scenario: 22
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

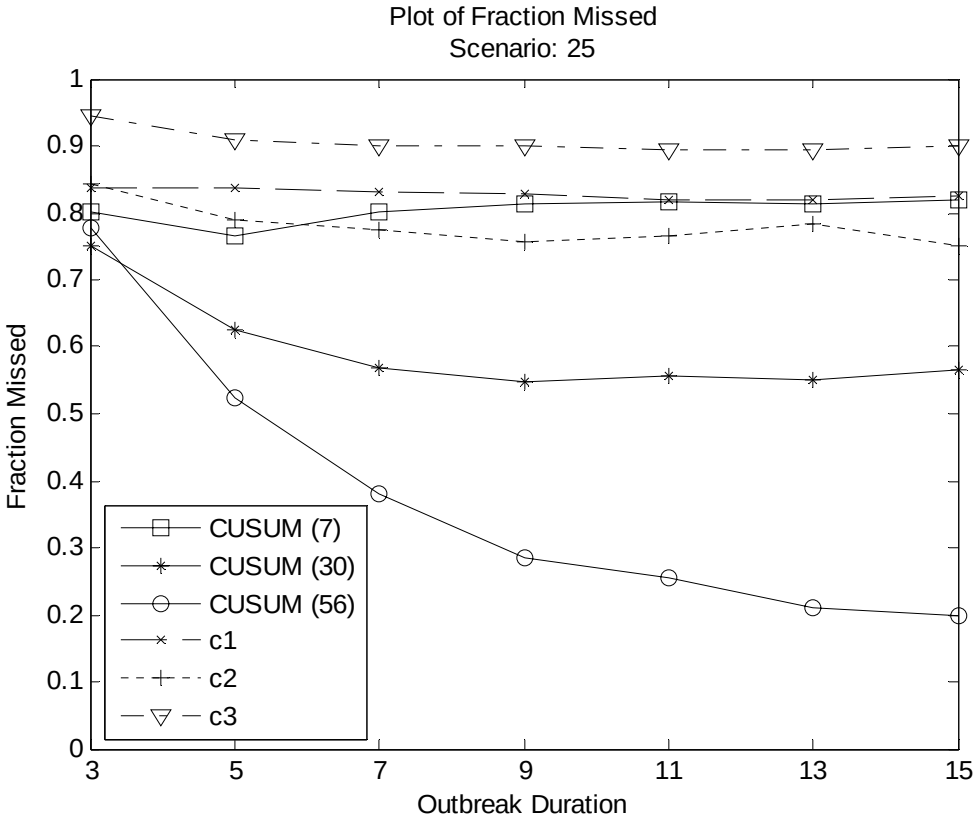

Plot of ATFS Given True Signal
Scenario: 23
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

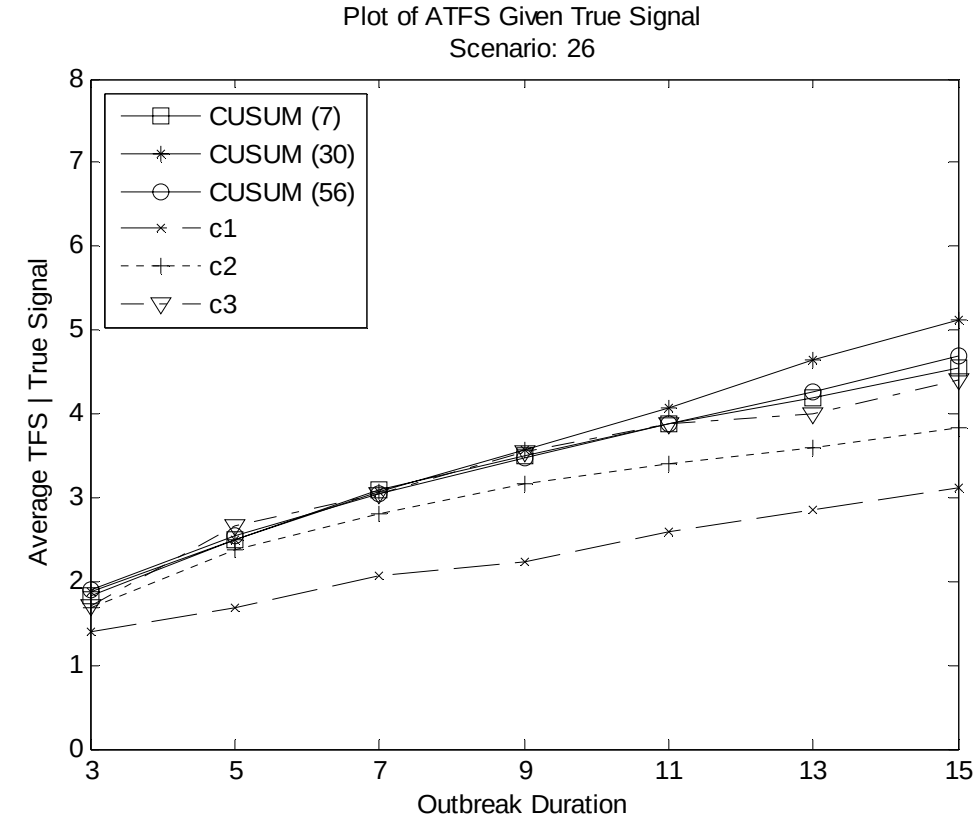

Plot of Fraction Missed
Scenario: 23
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

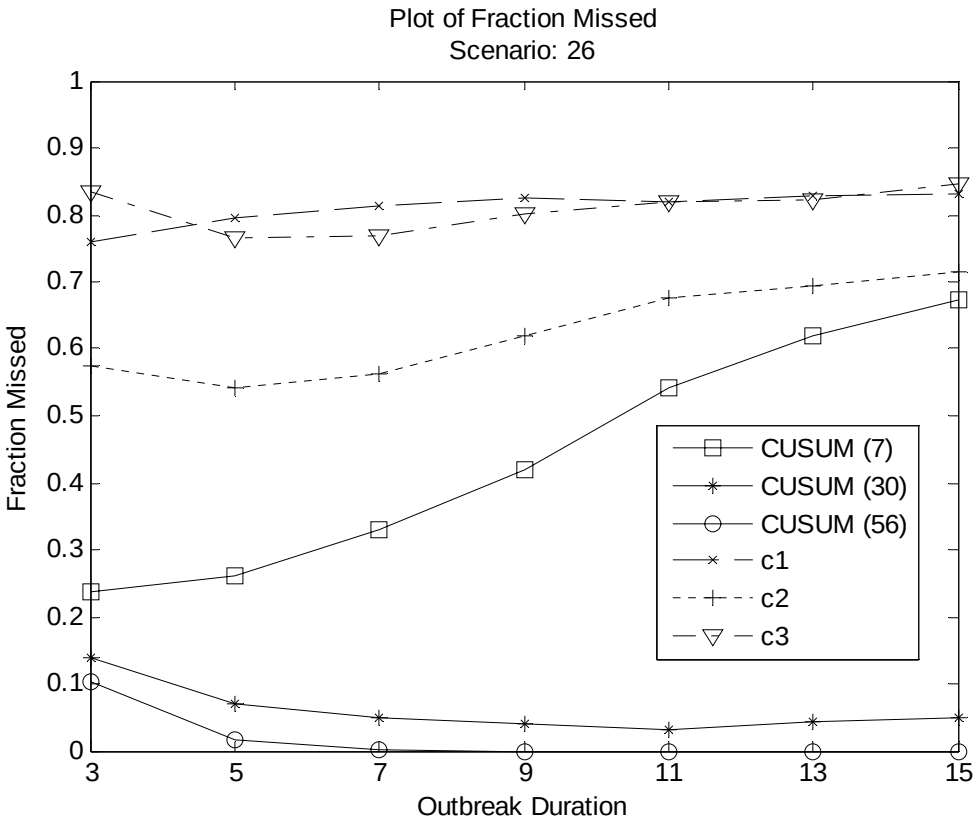

Plot of ATFS Given True Signal
Scenario: 24
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

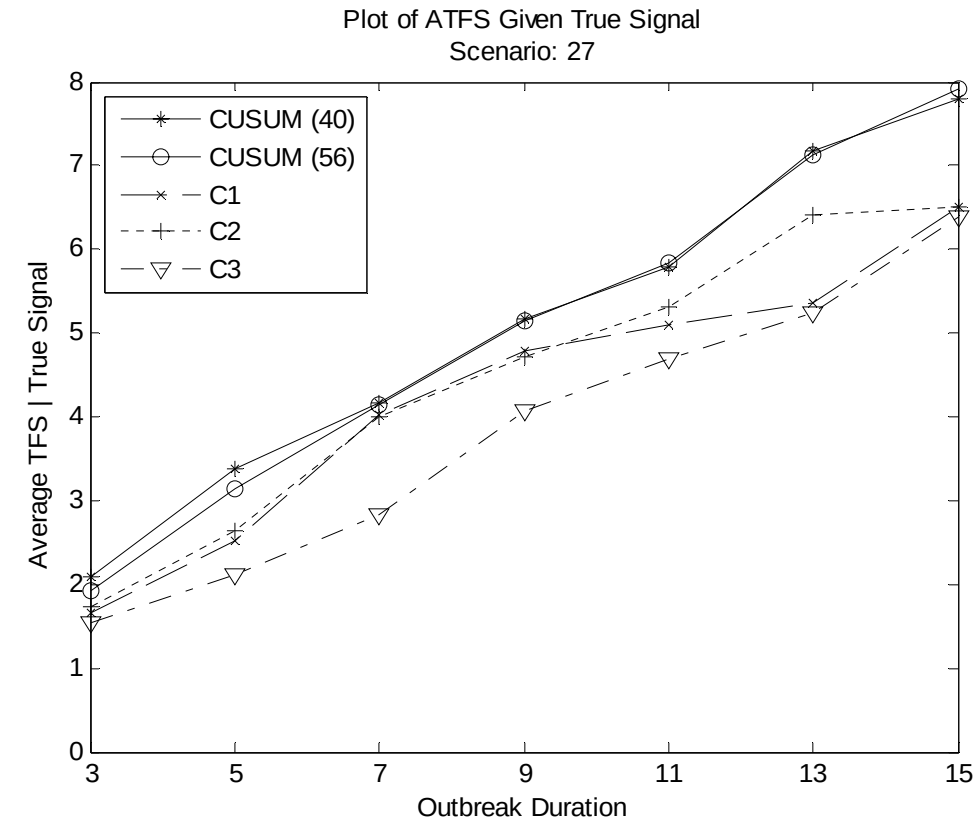

Plot of Fraction Missed
Scenario: 24
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

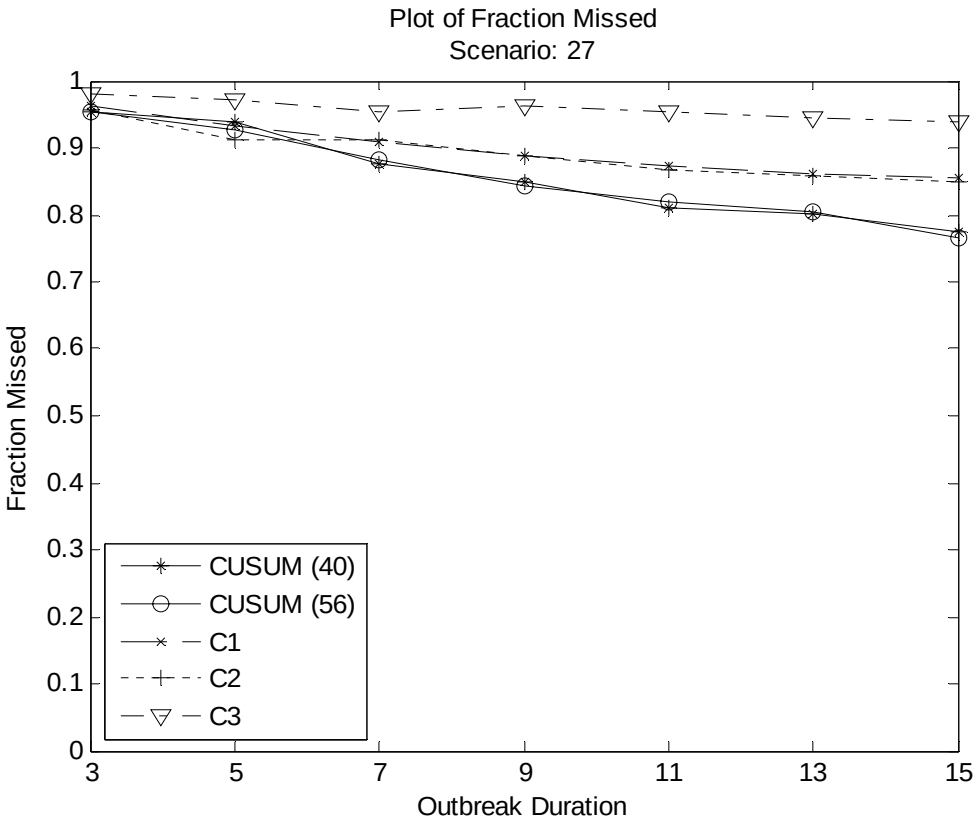

Plot of ATFS Given True Signal
Scenario: 25
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

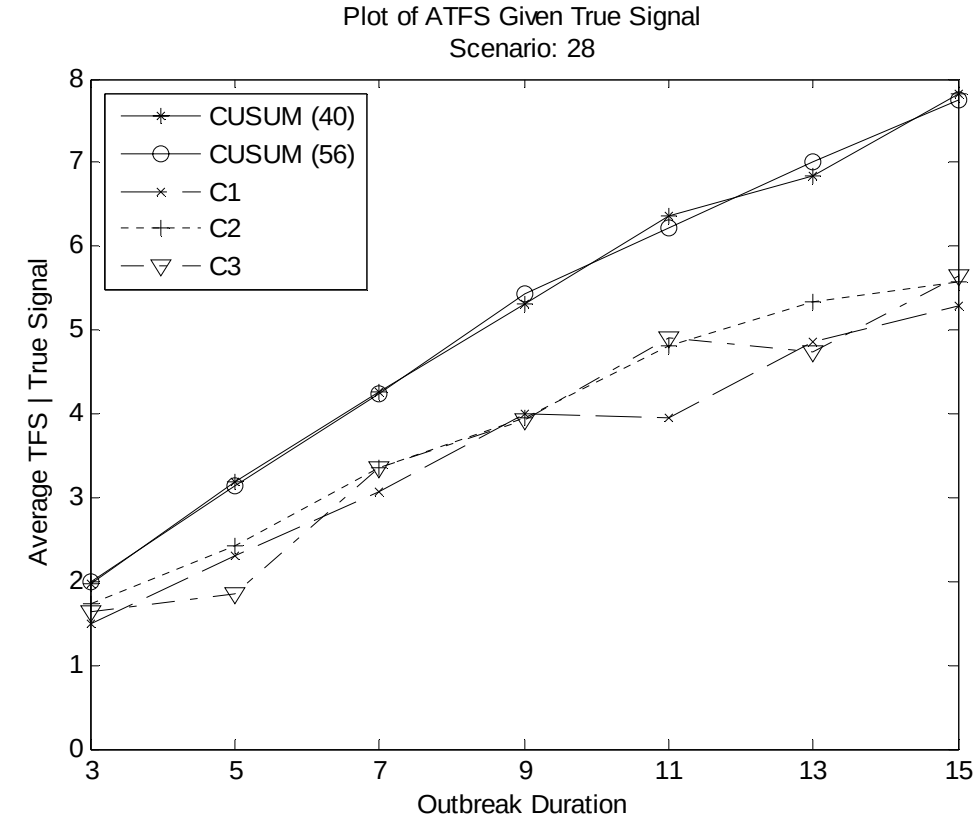

Plot of Fraction Missed
Scenario: 25
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

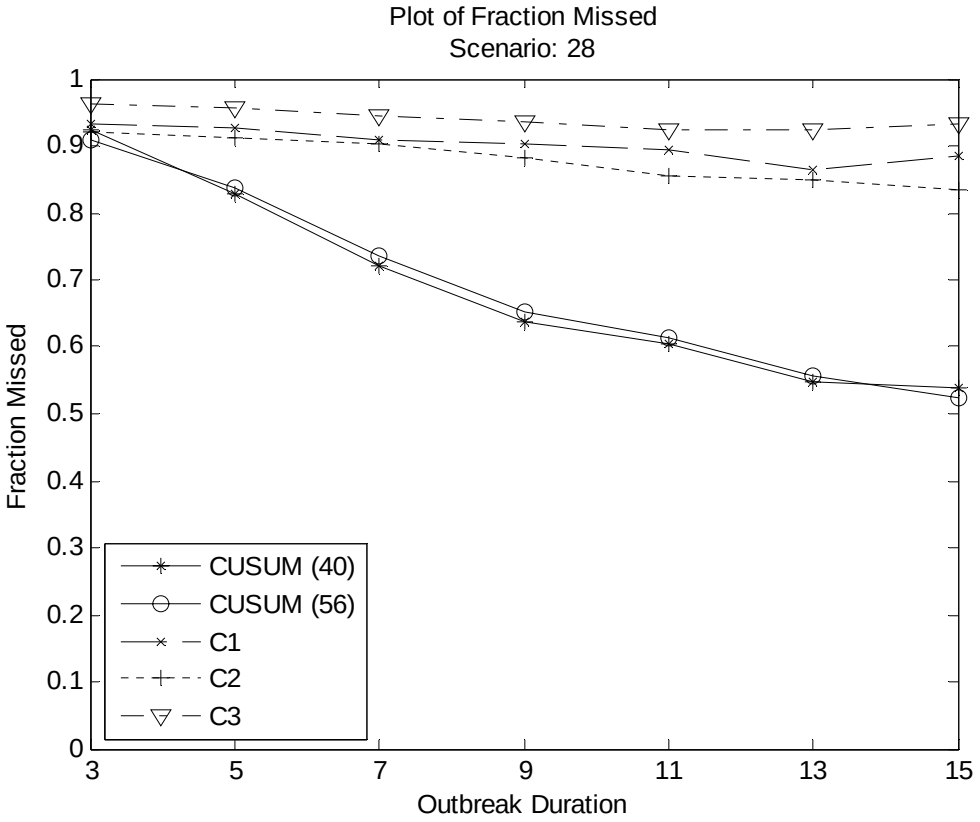

Plot of ATFS Given True Signal
Scenario: 26
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

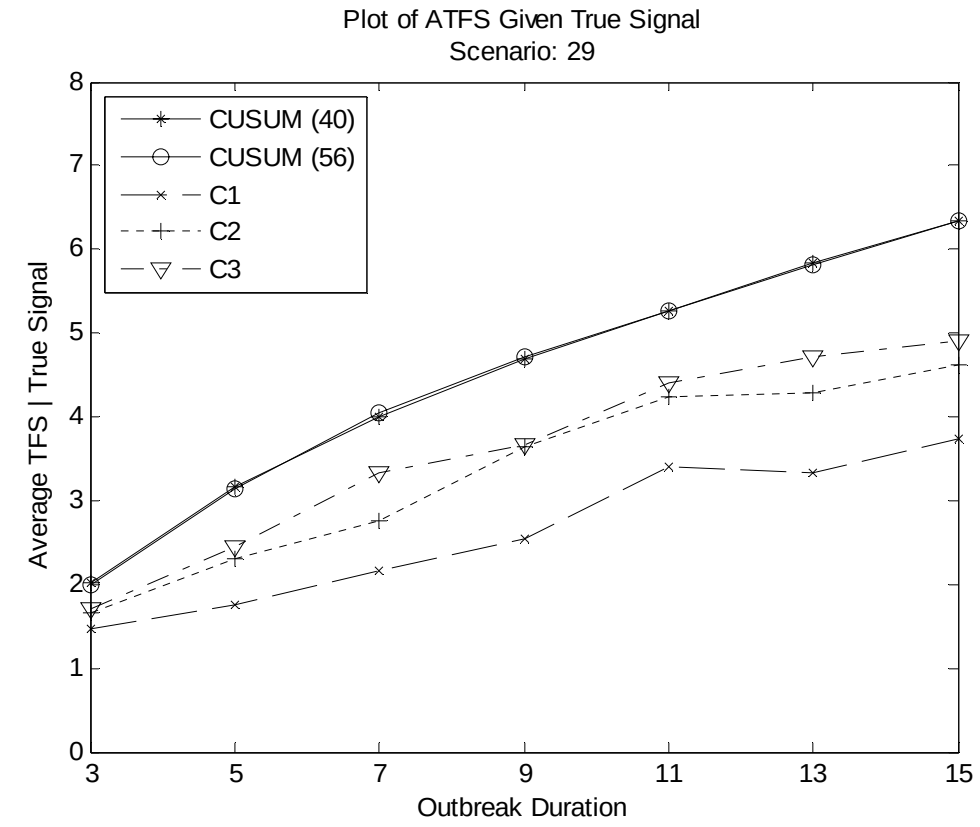

Plot of Fraction Missed
Scenario: 26
CUSUM (7)
CUSUM (30)
CUSUM (56)
c1
c2
c3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

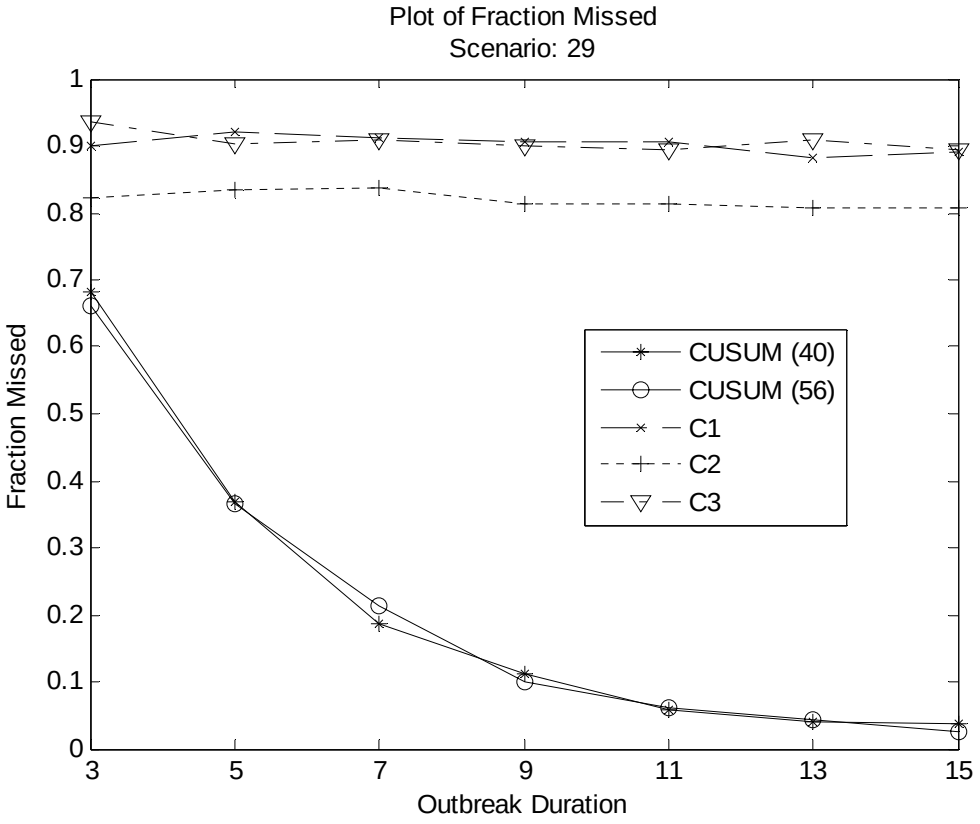

Plot of ATFS Given True Signal
Scenario: 27
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

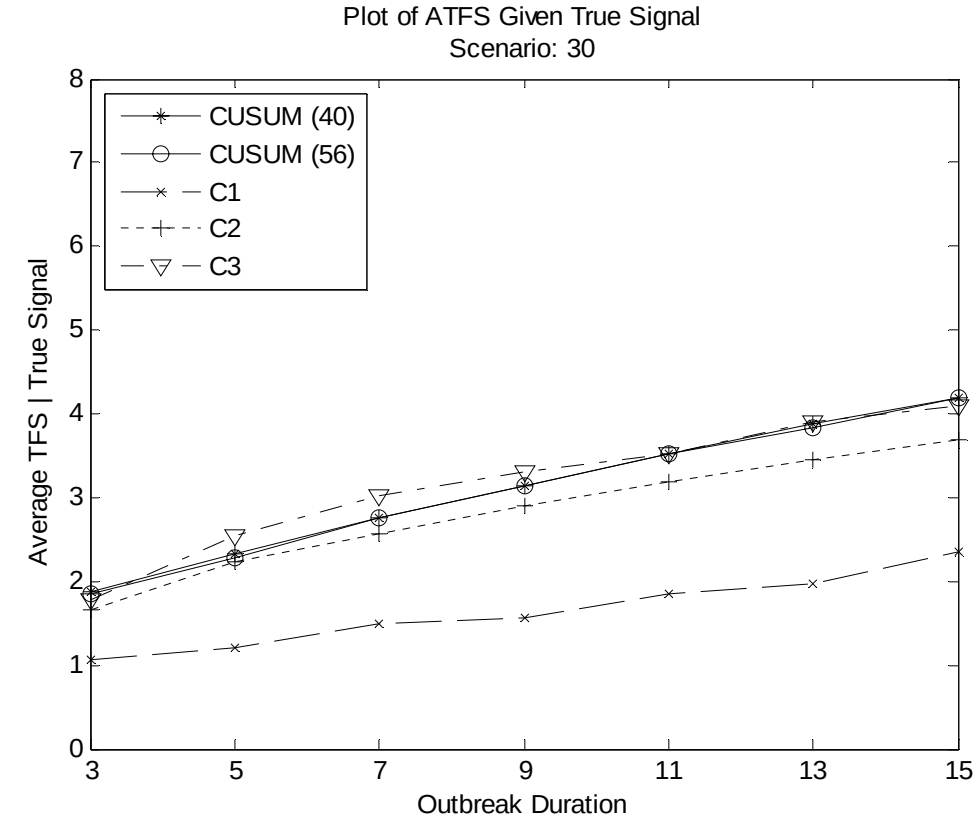

Plot of Fraction Missed
Scenario: 27
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

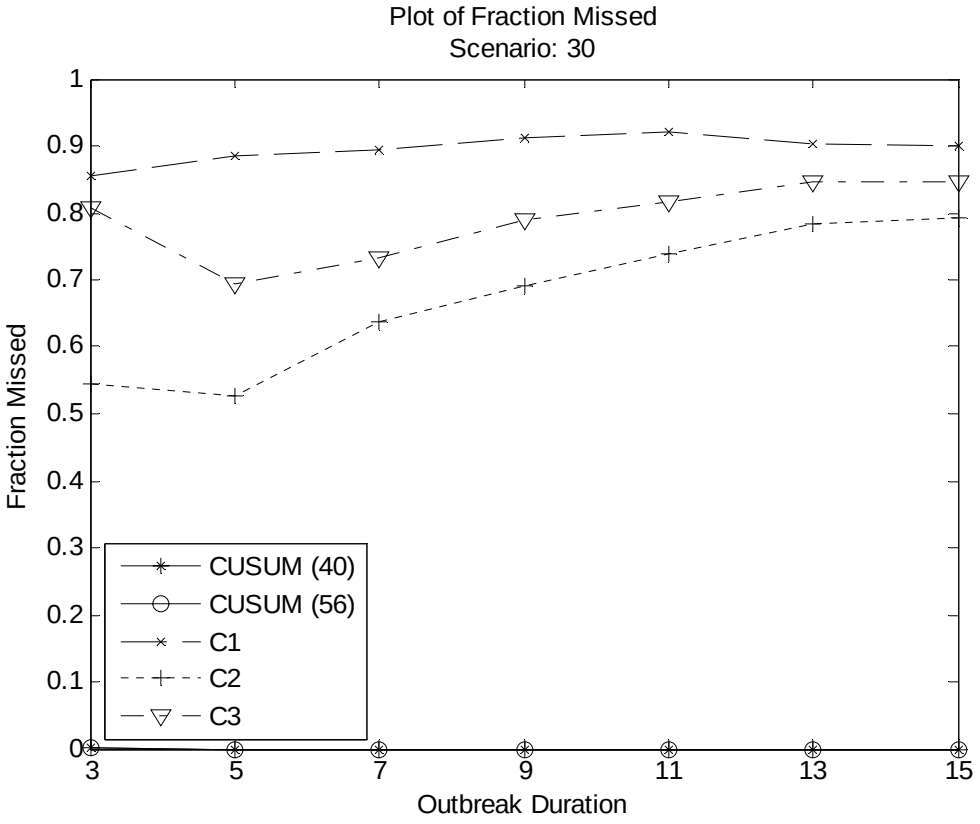

Plot of ATFS Given True Signal
Scenario: 28
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

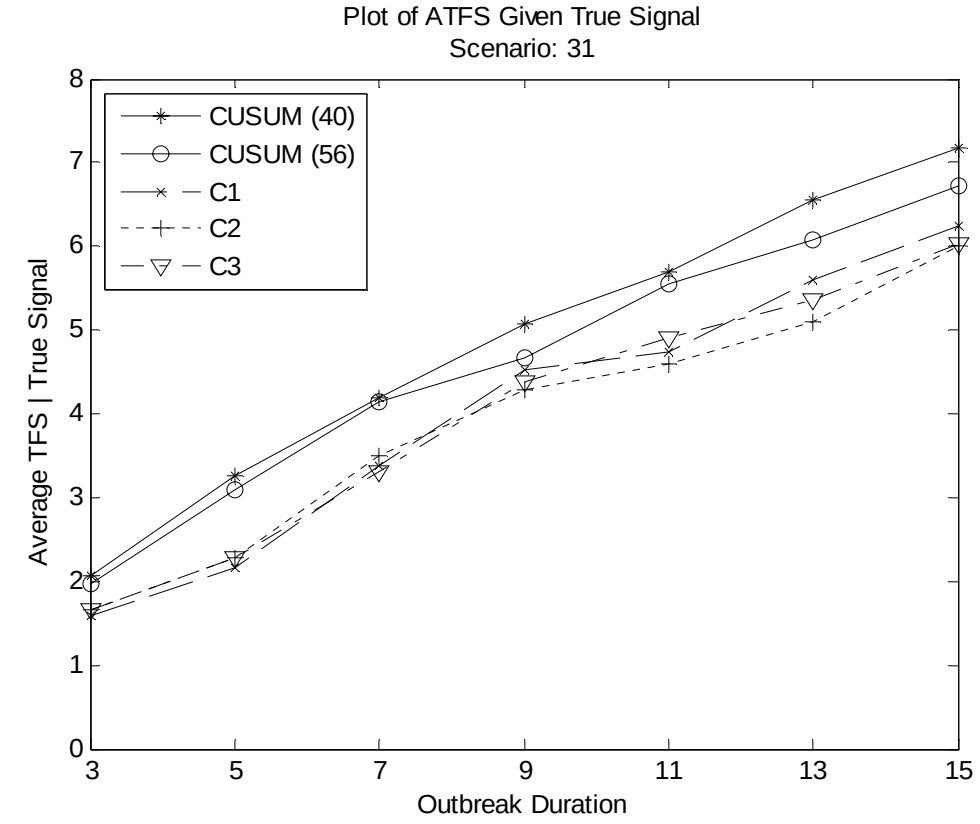

Plot of Fraction Missed
Scenario: 28
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

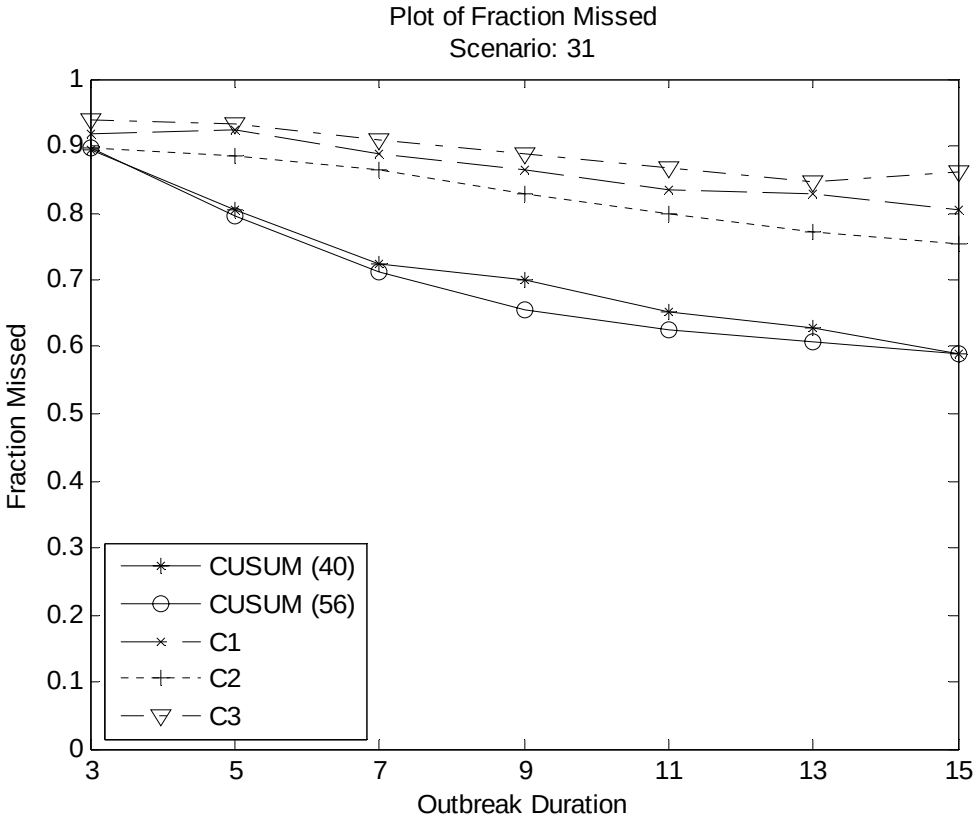

Plot of ATFS Given True Signal
Scenario: 29
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

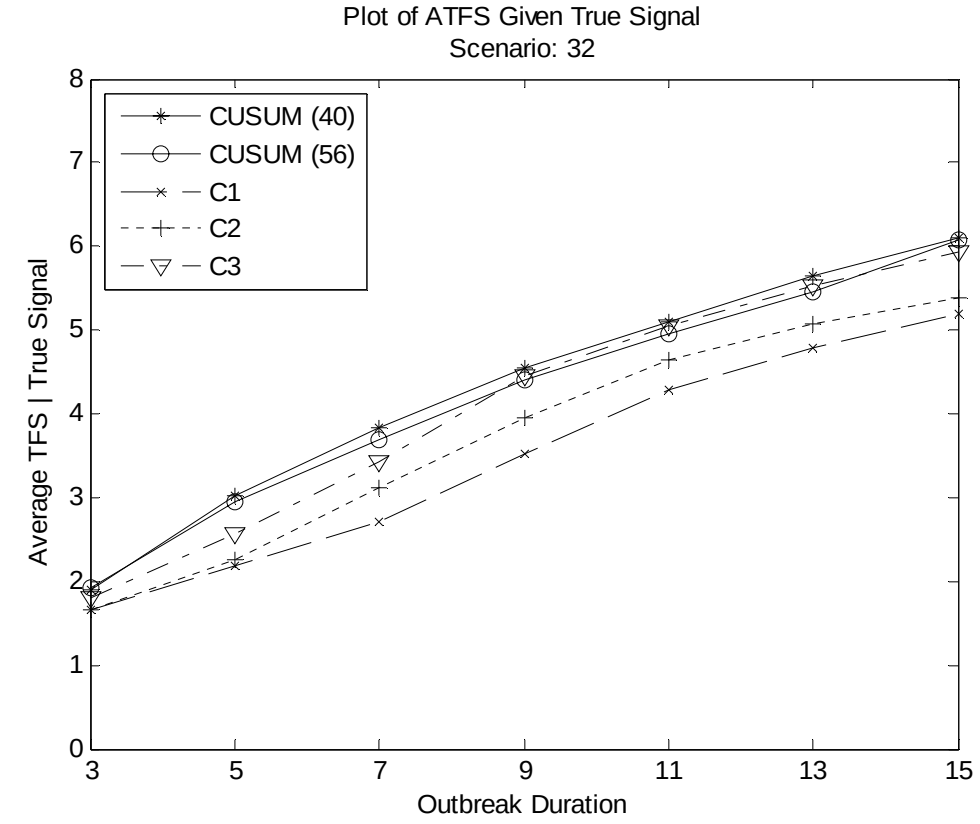

Plot of Fraction Missed
Scenario: 29
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

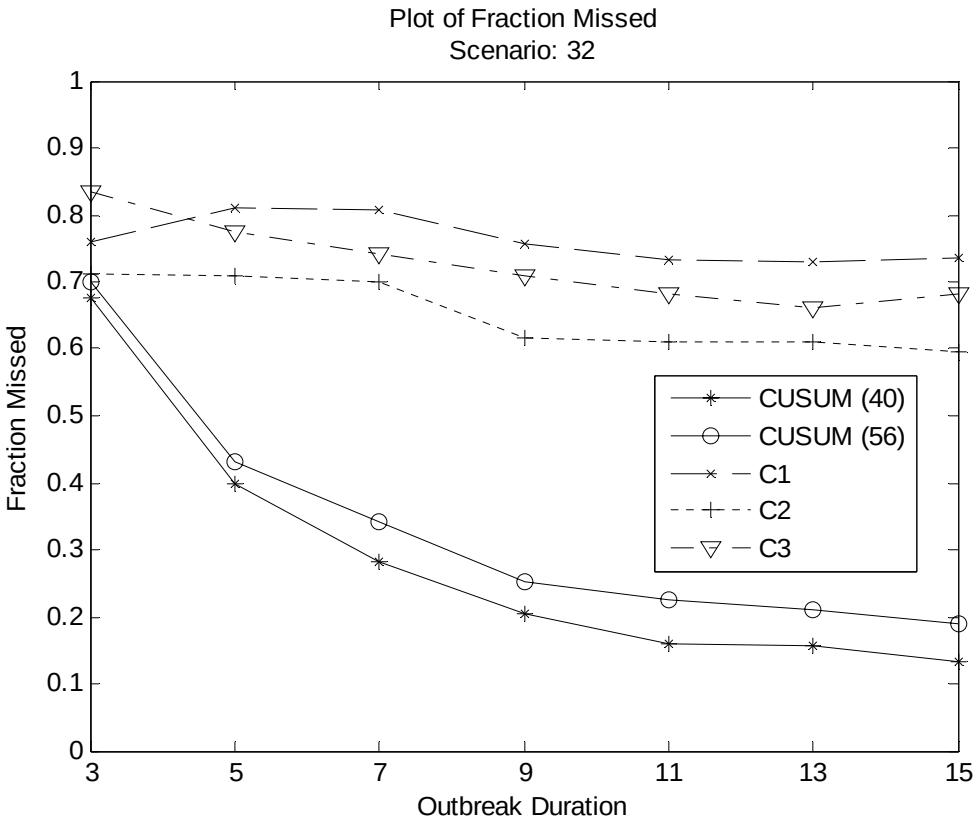

Plot of ATFS Given True Signal
Scenario: 30
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15

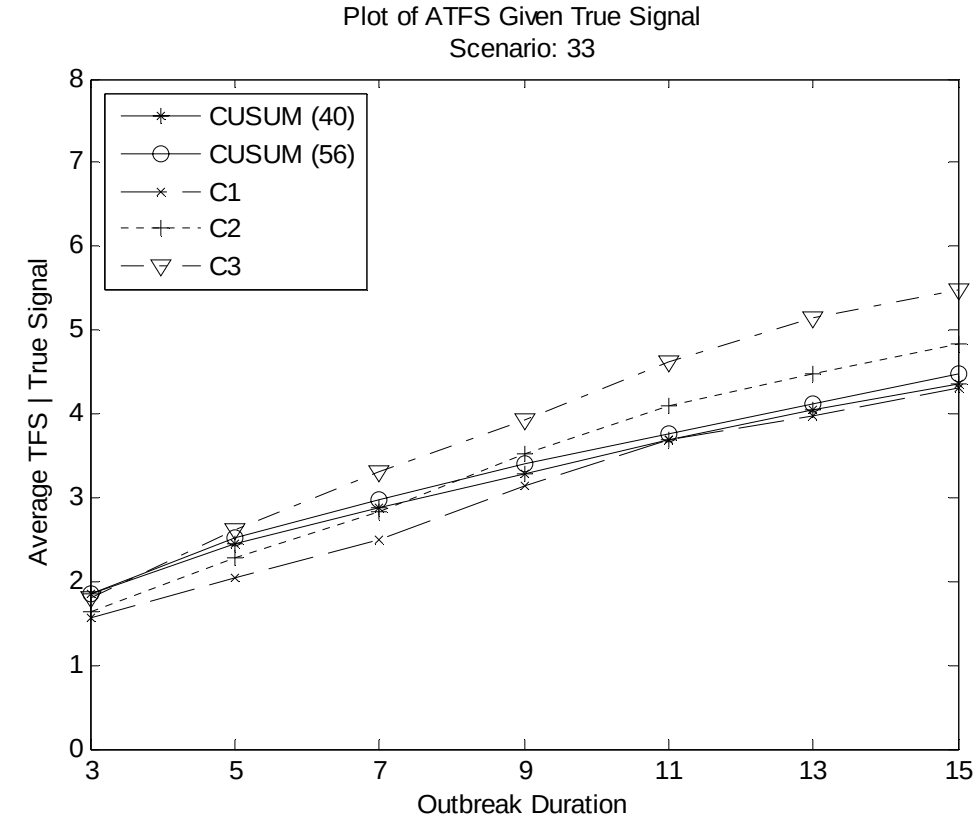

Plot of Fraction Missed
Scenario: 30
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

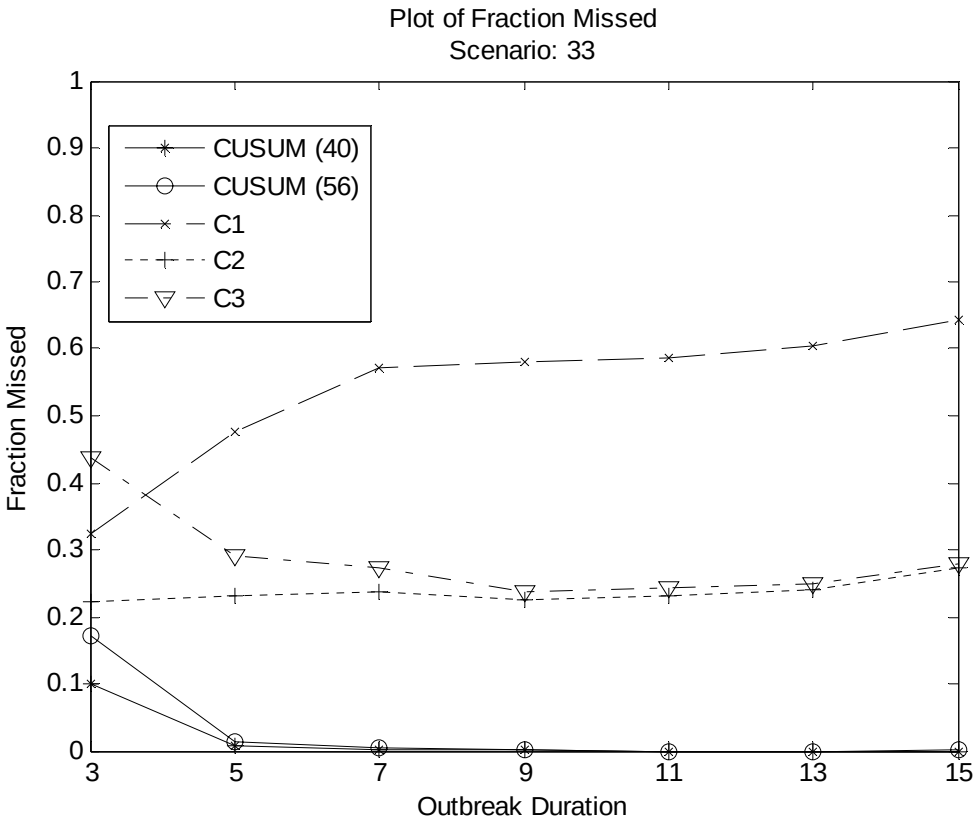

Plot of ATFS Given True Signal
Scenario: 31
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15


Plot of Fraction Missed
Scenario: 31
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15


Plot of ATFS Given True Signal
Scenario: 32
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15


Plot of Fraction Missed
Scenario: 32
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15


Plot of ATFS Given True Signal
Scenario: 33
CUSUM (40)
CUSUM (56)
C1
C2
C3
Average TFS | True Signal
Outbreak Duration
0
1
2
3
4
5
6
7
8
3
5
7
9
11
13
15


Plot of Fraction Missed
Scenario: 33
CUSUM (40)
CUSUM (56)
C1
C2
C3
Fraction Missed
Outbreak Duration
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
3
5
7
9
11
13
15

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: MATLAB SIMULATION CODE

Optimal *n* Code

```
clear
clc
%INITIALIZE VALUES
numYears =30;
numSimDays = 365*numYears; % number of days of data to create
lookBack(1) = 0; % number days to regress over
numWeeks = 20; % maximum number of weeks to regress over (n increases by
7 up to 7*numWeeks)
baseline = 0;
amplitude = 6;
sigma = 0.7;
sigmaZ=2.8;


% Steadily increase # of days to regress over
for (i = 2:1:numWeeks)

lookBack(i) = lookBack(i-1) + 7;

% X matrices for regressions; First Column: ones, Second Column: day #
matX1 = [ ones(lookBack(i),1) (1:1:lookBack(i))' ];  %linear model
matX2 = [ ones(lookBack(i),1) (1:1:lookBack(i))' ((1:1:lookBack(i)).^2)'
]; %model with square term

% DATA SIMULATION-----------------------------
for (hospCountsDay = 1:1:numSimDays)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
```

```
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*hospCountsDay/365))+ baseline;  % This is the
seasonal mean of the process
%rand_var=randn*sigma;  % For large counts
rand_var=lognrnd(1.0,sigma);  % For small counts

hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var));  %  an
observation on day "hospCountsDay"

% If enough data is available for this "lookBack" number of regress
days, regress
if (hospCountsDay >= (lookBack(i)+1))

% Vector "countLookBack" holds the previous "lookBack" # of days of
count values
 countLookBack          =          [          hospCounts((hospCountsDay-
lookBack(i)):1:(hospCountsDay-1))];

% DAILY REGRESSION CALCULATION----------------
% Regress from day hospCountsDay back "lookBack" number of days (use a
lag later?)

b = regress(countLookBack', matX1); % linear model; b = regress(X,Y)
where  X = days, Y = values(counts)
a = matX2\countLookBack'; % squared term model

% PREDICT "TOMORROW'S" COUNT-------------------
tomorrowCount = [1 (lookBack(i)+1)]; % 1 for intercept, (lookBack + 1)
for tomorrow's day #
predCount(hospCountsDay) = tomorrowCount*b; % linear model

tomorrowCount2 = [1 (lookBack(i)+1) (lookBack(i) + 1).^2]; % 1 for
intercept, (lookBack + 1) for tomorrow
predCount2(hospCountsDay) = tomorrowCount2*a; % squared term model

% Calculate residual values ( resid = predicted - actual)
residual(hospCountsDay)      =      hospCounts(hospCountsDay      )      -
predCount(hospCountsDay); % linear model
sqdResidual(hospCountsDay) = residual(hospCountsDay)^2;
```

```
residual2(hospCountsDay)     =     hospCounts(hospCountsDay     )     -
predCount2(hospCountsDay); % squared term model
sqdResidual2(hospCountsDay) = residual2(hospCountsDay)^2;

end % end if-statement
end % end hospCountsDay's for-loop

avgSqdResidual(i)    =    mean(sqdResidual((lookBack(i)+1):numSimDays));
%/(numSimDays - lookBack + 1) LINEAR MDL
avgSqdResidual2(i)   =   mean(sqdResidual2((lookBack(i)+1):numSimDays));
%/(numSimDays - lookBack + 1) SQUARED MDL

end % end i's for-loop
%
%Plot n vs avgSqdResidual
    plot(lookBack(2:numWeeks), avgSqdResidual(2:numWeeks), 'b')
    xlabel('# regress days');
    ylabel('avgSqdResidual');

hold on

% %Plot n vs avgSqdResidual2 (squared case)
      plot(lookBack(2:numWeeks), avgSqdResidual2(2:numWeeks), 'g')
%     xlabel('# regress days');
%     ylabel('avgSqdResidual');
      legend('Linear', 'Square');
%  %Plot hospital counts
%subplot(1,2,2), plot((1:1:1000), hospCounts(1:1000))
%     xlabel('day');
%     ylabel('count');
```

Optimal *h* Code

```
clear
clc
% want h of each scenario to give ATFS of 100, with SD of about 1.0,
then fix these.
% What are appropriate magnitudes and slopes of outbreaks
% for a given scenario (with n), and optimal h, plot ATFS vs slope of
% outbreak for CUSUM, C1, C2, C3 to compare them all
numLoops = 1000;
baseline = 90;
amplitude = 80;
sigma = 10;
sigmaZ = 2.8;
k = sigma/2;  % for large-count CUSUM
%k = sigmaZ/2;  % for small-count CUSUM
%k = sigma*0.65;  % for large-count CUSUM w/ lookBack=7
%k = sigmaZ*0.65;  % for small-count CUSUM w/ lookBack=7
h = 41.9; % (cusum)
cusum = 0;
runLengthCounter = 0;
TFS(1) = 0 ;
alarmCount = 0;
lookBack = 56;

% START CUSUM METHOD_____________________________________________
for(dummy=1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:(lookBack + 1))
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(j,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(j,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
```

```
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(j,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var));  % an observation
on day "hospCountsDay"
end

hospCountsDay = lookBack + 2;

while (cusum < h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts
```

```
hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var));

matX1 = [ ones(lookBack,1) (1:1:lookBack)' ];
matX2 = zeros(lookBack,6);
for i=1:lookBack;
    columnX2=mod(hospCountsDay-i,7)+1;
    if(columnX2<7)
        matX2(lookBack+1-i,columnX2)=1;
    end;
end;
matX =[matX1 matX2];

countLookBack = [ hospCounts((hospCountsDay-lookBack):1:(hospCountsDay-
1))];
b = regress(countLookBack', matX);

daysInd=[0, 0, 0, 0, 0, 0];
columnInd=mod(hospCountsDay,7)+1;
if(columnInd<7)
    daysInd(columnInd)=1;
end;

tomorrowCount = [1 (lookBack+1) daysInd]; % 1 for intercept, (lookBack +
1) for tomorrow's day #
predCount(hospCountsDay) = tomorrowCount*b; %it's like you're predicting
the mean of the process
residual(hospCountsDay)     =     hospCounts(hospCountsDay     )     -
predCount(hospCountsDay);

cusum = max(0, (residual(hospCountsDay) - k + cusum));

runLengthCounter = runLengthCounter + 1;
hospCountsDay = hospCountsDay + 1;
end % end while (cusum < h) loop

alarmCount = alarmCount + 1;
TFS(alarmCount) = runLengthCounter; % after "alarm", you have a new TFS
value
runLengthCounter = 0; % after "alarm", reset runLengthCounter
cusum = 0; % after "alarm", reset runLengthCounter

end % end i's for-loop
averageTFS = mean(TFS)
stdErrTFS = std(TFS)/(numLoops^0.5)

%
% START C1 METHOD_____________________________________________
alarmCountC1 = 0;
runLengthCounterC1 = 0;
TFSc1(1)=0;
c1h = 2.7;
c1Statistic = 0;
sdMovingAvg=100;
```

```
for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:7)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

hospCounts(j)=max(0,ceil(meanie+rand_var+dayeffect));  %  an  observation
on day "hospCountsDay"
end

hospCountsDay = 8;

while (c1Statistic < c1h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
```

```
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts
hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var));

movingAvg(hospCountsDay)      =      (hospCounts(hospCountsDay-7)      +
hospCounts(hospCountsDay-6)     +      hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4)     +      hospCounts(hospCountsDay-3)     +
hospCounts(hospCountsDay-2) + hospCounts(hospCountsDay-1))/7;

if (length(movingAvg) >= 14) %need 7 (nonzero) values for an average
if (sdMovingAvg>0)
    oldsdMovingAvg=sdMovingAvg;
end
    sdMovingAvg        =        ((((hospCounts(hospCountsDay-7)        -
movingAvg(hospCountsDay-7))^2)    +    ((hospCounts(hospCountsDay-6)    -
movingAvg(hospCountsDay-6))^2)    +    ((hospCounts(hospCountsDay-5)    -
movingAvg(hospCountsDay-5))^2)    +    ((hospCounts(hospCountsDay-4)    -
movingAvg(hospCountsDay-4))^2)    +    ((hospCounts(hospCountsDay-3)    -
movingAvg(hospCountsDay-3))^2)    +    ((hospCounts(hospCountsDay-2)    -
movingAvg(hospCountsDay-2))^2)    +    ((hospCounts(hospCountsDay-1)    -
movingAvg(hospCountsDay-1))^2))/6)^0.5;
if (sdMovingAvg ==0)
    sdMovingAvg=oldsdMovingAvg;
end
c1Statistic           =           (hospCounts(hospCountsDay)           -
movingAvg(hospCountsDay))/sdMovingAvg;
runLengthCounterC1 = runLengthCounterC1 + 1;
end
```

```
hospCountsDay = hospCountsDay + 1;
end % while loop when "alarm" occurs

alarmCountC1 = alarmCountC1 + 1;
TFSc1(alarmCountC1) = runLengthCounterC1; % after "alarm", you have a
new TFS value
runLengthCounterC1 = 0; % after "alvarm", reset runLengthCounter
c1Statistic = 0;

end % dummy's for loop
averageTFSc1 = mean(TFSc1)
stdErrTFSc1 = std(TFSc1)/(numLoops^0.5)

% START C2 METHOD_____________________________________________
alarmCountC2 = 0;
runLengthCounterC2 = 0;
TFSc2(1) = 0;
c2h = 2.61;
c2Statistic = 0;
c2StatTodayMinus2 = 0;
c2StatTodayMinus1 = 0;
c2StatToday = 0;
sdMovingAvg=100;

for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:9)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
```

```
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
  elseif (j==8)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==9)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
  end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var));  % an  observation
on day "hospCountsDay"
end

hospCountsDay = 10;

    while (c2Statistic < c2h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
```

```
    end

    meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+
baseline;
    rand_var=randn*sigma;  % For large counts
    %rand_var=lognrnd(1.0,sigma);  % For small counts
    hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var));

    movingAvg(hospCountsDay)     =     (hospCounts(hospCountsDay-9)     +
hospCounts(hospCountsDay-8)     +     hospCounts(hospCountsDay-7)     +
hospCounts(hospCountsDay-6)     +     hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4) + hospCounts(hospCountsDay-3))/7;

        if (length(movingAvg) >= 16) %need 7 (16-9) days for an average
            if (sdMovingAvg>0)
                oldsdMovingAvg=sdMovingAvg;
            end
        sdMovingAvg       =       ((((hospCounts(hospCountsDay-9)       -
movingAvg(hospCountsDay-9))^2)    +    ((hospCounts(hospCountsDay-8)    -
movingAvg(hospCountsDay-8))^2)    +    ((hospCounts(hospCountsDay-7)    -
movingAvg(hospCountsDay-7))^2)    +    ((hospCounts(hospCountsDay-6)    -
movingAvg(hospCountsDay-6))^2)    +    ((hospCounts(hospCountsDay-5)    -
movingAvg(hospCountsDay-5))^2)    +    ((hospCounts(hospCountsDay-4)    -
movingAvg(hospCountsDay-4))^2)    +    ((hospCounts(hospCountsDay-3)    -
movingAvg(hospCountsDay-3))^2))/6)^0.5;
            if (sdMovingAvg==0)
                sdMovingAvg=oldsdMovingAvg;
            end

        c2Statistic         =         (hospCounts(hospCountsDay)         -
movingAvg(hospCountsDay))/sdMovingAvg;
        c2StatTodayMinus2 = c2StatTodayMinus1; % values on right are one
day old right now
        c2StatTodayMinus1 = c2StatToday;
        c2StatToday = c2Statistic;

        runLengthCounterC2 = runLengthCounterC2 + 1;
        end
    hospCountsDay = hospCountsDay + 1;
    end % while loop when "alarm" occurs

alarmCountC2 = alarmCountC2 + 1;
TFSc2(alarmCountC2) = runLengthCounterC2; % after "alarm", you have a
new TFS value
runLengthCounterC2 = 0; % after "alarm", reset runLengthCounter
c2Statistic = 0;

end % dummy's for loop
averageTFSc2 = mean(TFSc2)
stdErrTFSc2 = std(TFSc2)/(numLoops^0.5)


% START C3 METHOD_____________________________________________
alarmCountC3 = 0;
```

```
runLengthCounterC3 = 0;
TFSc3(1) = 0;
c3h = 3.38;
c2Statistic = 0;
c2StatTodayMinus2 = 0;
c2StatTodayMinus1 = 0;
c2StatToday = 0;
c3Statistic = 0;
sdMovingAvg=100;

for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:9)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;   % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
  elseif (j==8)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==9)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
```

```
        end
  end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var));  % an observation
on day "hospCountsDay"
end

hospCountsDay = 10;

    while (c3Statistic < c3h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

    meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+
baseline;
    rand_var=randn*sigma;  % For large counts
    %rand_var=lognrnd(1.0,sigma);  % For small counts
    hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var));

    movingAvg(hospCountsDay)     =     (hospCounts(hospCountsDay-9)     +
hospCounts(hospCountsDay-8)     +     hospCounts(hospCountsDay-7)     +
hospCounts(hospCountsDay-6)     +     hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4) + hospCounts(hospCountsDay-3))/7;

        if (length(movingAvg) >= 16) %need 7 (16-9) days for an average
```

```
            if (sdMovingAvg>0)
                oldsdMovingAvg=sdMovingAvg;
            end
        sdMovingAvg      =      ((((hospCounts(hospCountsDay-9)      -
movingAvg(hospCountsDay-9))^2)    +    ((hospCounts(hospCountsDay-8)    -
movingAvg(hospCountsDay-8))^2)    +    ((hospCounts(hospCountsDay-7)    -
movingAvg(hospCountsDay-7))^2)    +    ((hospCounts(hospCountsDay-6)    -
movingAvg(hospCountsDay-6))^2)    +    ((hospCounts(hospCountsDay-5)    -
movingAvg(hospCountsDay-5))^2)    +    ((hospCounts(hospCountsDay-4)    -
movingAvg(hospCountsDay-4))^2)    +    ((hospCounts(hospCountsDay-3)    -
movingAvg(hospCountsDay-3))^2))/6)^0.5;
            if (sdMovingAvg==0)
                sdMovingAvg=oldsdMovingAvg;
            end

        c2Statistic        =        (hospCounts(hospCountsDay)        -
movingAvg(hospCountsDay))/sdMovingAvg;
        c2StatTodayMinus2 = c2StatTodayMinus1; % values on right are one
day old right now
        c2StatTodayMinus1 = c2StatToday;
        c2StatToday = c2Statistic;

            if (length(movingAvg) >= 19) %need 3 C2 values for C3 (and
all 3 c2Stat values are != 0)
            c3Statistic   =   max(0,   (c2StatToday)   -   1)   +   max(0,
(c2StatTodayMinus1) - 1) + max(0, (c2StatTodayMinus2) - 1);
            runLengthCounterC3 = runLengthCounterC3 + 1;

            end
%        runLengthCounterC2 = runLengthCounterC2 + 1;
        end
    hospCountsDay = hospCountsDay + 1;
    end % while loop when "alarm" occurs

alarmCountC3 = alarmCountC3 + 1;
TFSc3(alarmCountC3) = runLengthCounterC3; % after "alarm", you have a
new TFS value
runLengthCounterC3 = 0; % after "alarm", reset runLengthCounter
c3Statistic = 0;

end % dummy's for loop
averageTFSc3 = mean(TFSc3)
stdErrTFSc3 = std(TFSc3)/(alarmCountC3^0.5)
```

Comparison Metrics Code

```
clear
clc

numLoops = 2000;
duration = [3 5 7 9 11 13 15];

% Values that change depending on the scenario under consideration:
scenarioNumber = 31; %defines a specific baseline/amplitude/sigma/outbreak percent combination
baseline = 90;
amplitude = 80;
sigma = 10;
%magnitude = 2; %for small counts: [SD=0.5 : outbreak = 1, 2, 4] [SE=0.7 : outbreak = 2, 4, 8]
magnitude = .1*baseline; % % Large disease outbreak: 10, 25, 50% of baseline

MYlookback_optimal = 40; % is # of days that our optimal CUSUM uses to look over
MY_h_optimal = 39;
MY_h_56day = 41.9;
MYc1h = 2.7;
MYc2h = 2.61;
MYc3h = 3.38;

% START CUSUM METHOD - "Sub-OPTIMAL FOR SCENARIO" day lookback_____________________________________
k = sigma/2;
cusum = 0;
runLengthCounter = 0;
TFS(1) = 0 ;

lookBack = MYlookback_optimal; % (cusum Optimal)
h = MY_h_optimal; % (cusum Optimal)
alarmCount = 0;

for (durationIndex=1:1:7)

for(dummy=1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:(lookBack + 1))
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % Seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(j,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
```

```
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(j,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(j,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var)); % an observation
on day "hospCountsDay"
end

hospCountsDay = lookBack + 2;

while (cusum < h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
```

```
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

if       (hospCountsDay       >       100       &&       hospCountsDay
<=(100+(duration(durationIndex)+1)/2))
    outbreak                                              =(hospCountsDay-
100)*magnitude/((duration(durationIndex)+1)/2);
elseif    (hospCountsDay    >    (100+(duration(durationIndex)+1)/2)    &&
hospCountsDay < (100+duration(durationIndex)))
    outbreak       =       magnitude       -       (hospCountsDay-100-
((duration(durationIndex)+1)/2))*magnitude/((duration(durationIndex)+1)/
2);
else
    outbreak=0;
end

hospCounts(hospCountsDay) = max(0,ceil(meanie + dayeffect + rand_var +
outbreak));

matX1 = [ ones(lookBack,1) (1:1:lookBack)' ];
matX2 = zeros(lookBack,6);
for i=1:lookBack;
    columnX2=mod(hospCountsDay-i,7)+1;
    if(columnX2<7)
        matX2(lookBack+1-i,columnX2)=1;
    end;
end;
matX =[matX1 matX2];

countLookBack = [ hospCounts((hospCountsDay-lookBack):1:(hospCountsDay-
1)) ];
b = regress(countLookBack', matX);

daysInd=[0, 0, 0, 0, 0, 0];
columnInd=mod(hospCountsDay,7)+1;
if(columnInd<7)
    daysInd(columnInd)=1;
end;

tomorrowCount = [1 (lookBack+1) daysInd]; % 1 for intercept, (lookBack +
1) for tomorrow's day #
```

```
predCount(hospCountsDay) = tomorrowCount*b; %it's like you're predicting
the mean of the process
residual(hospCountsDay)     =     hospCounts(hospCountsDay     )     -
predCount(hospCountsDay);

cusum = max(0, (residual(hospCountsDay) - k + cusum));

if (hospCountsDay <= 100 && cusum >= h)
    cusum=0; % False alarm. Don't signal.
elseif (hospCountsDay > 100)
    runLengthCounter = runLengthCounter + 1;
end

hospCountsDay = hospCountsDay + 1;
end % end while (cusum < h) loop. An alarm has occurred.

alarmCount = alarmCount + 1;

TFSallCusumOpt(alarmCount) = runLengthCounter;

if (runLengthCounter <= duration(durationIndex))
    TFScusumOpt(alarmCount) = runLengthCounter;
else
    TFScusumOpt(alarmCount) = -99; %did not catch outbreak by its end
end

runLengthCounter = 0; % after "alarm", reset runLengthCounter and cusum
cusum = 0;
end % end dummy for-loop

alarmCount=0; % reset previous alarmCount, reset runningSum stuff
runningSumTFS=0;
runningSumTFSCounter=0;
runningSumTFSall=0;
runningSqdSumTFScusumOpt = 0;

for (dummy2 = 1:1:numLoops)
    if (TFScusumOpt(dummy2) > 0)
        runningSumTFS  =  runningSumTFS  +  TFScusumOpt(dummy2);  %only
adding positive TFS values
        runningSumTFSCounter = runningSumTFSCounter + 1; %number of on-
time signals
    end % end if TFS(dummy2)>0
runningSumTFSall = runningSumTFSall + TFSallCusumOpt(dummy2);
end % end dummy2 for-loop

averageTFScusumOpt(durationIndex) = runningSumTFS/runningSumTFSCounter;

for (dummy23 = 1:1:numLoops)
    if (TFScusumOpt(dummy23) > 0)
        runningSqdSumTFScusumOpt     =     runningSqdSumTFScusumOpt     +
(averageTFScusumOpt(durationIndex) - TFScusumOpt(dummy23))^2;
    end
```

```
end
seAverageTFScusumOpt(durationIndex)                                    =
(sqrt(runningSqdSumTFScusumOpt/(runningSumTFSCounter                   -
1)))/sqrt(runningSumTFSCounter);

averageTFSallCusumOpt(durationIndex)   =   runningSumTFSall/numLoops;   %
numLoops = # total alarms
fractionMissedcusumOpt(durationIndex)        =        (numLoops        -
runningSumTFSCounter)/numLoops;

end % end durationIndex for-loop


%     START     CUSUM     METHOD     -     8     week     (56     day)
lookback_____________________________________________
k = sigma/2;
h = MY_h_56day; % (cusum 56 day)
lookBack = 56;
cusum = 0;
alarmCount = 0;
runLengthCounter = 0;

for (durationIndex=1:1:7)

for(dummy=1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:(lookBack + 1))
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(j,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(j,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(j,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
```

```
    elseif (mod(j,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(j,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var)); % an observation
on day "hospCountsDay"
end

hospCountsDay = lookBack + 2;

while (cusum < h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts
```

```
if       (hospCountsDay       >       100       &&       hospCountsDay
<=(100+(duration(durationIndex)+1)/2))
    outbreak                                          =(hospCountsDay-
100)*magnitude/((duration(durationIndex)+1)/2);
elseif    (hospCountsDay    >    (100+(duration(durationIndex)+1)/2)    &&
hospCountsDay < (100+duration(durationIndex)))
    outbreak       =       magnitude       -       (hospCountsDay-100-
((duration(durationIndex)+1)/2))*magnitude/((duration(durationIndex)+1)/
2);
else
    outbreak=0;
end

hospCounts(hospCountsDay) = max(0,ceil(meanie + dayeffect + rand_var +
outbreak));

matX1 = [ ones(lookBack,1) (1:1:lookBack)' ];
matX2 = zeros(lookBack,6);
for i=1:lookBack;
    columnX2=mod(hospCountsDay-i,7)+1;
    if(columnX2<7)
        matX2(lookBack+1-i,columnX2)=1;
    end;
end;
matX =[matX1 matX2];

countLookBack = [ hospCounts((hospCountsDay-lookBack):1:(hospCountsDay-
1)) ];
b = regress(countLookBack', matX);

daysInd=[0, 0, 0, 0, 0, 0];
columnInd=mod(hospCountsDay,7)+1;
if(columnInd<7)
    daysInd(columnInd)=1;
end;

tomorrowCount = [1 (lookBack+1) daysInd]; % 1 for intercept, (lookBack +
1) for tomorrow's day #
predCount(hospCountsDay) = tomorrowCount*b; %it's like you're predicting
the mean of the process
residual(hospCountsDay)     =     hospCounts(hospCountsDay     )     -
predCount(hospCountsDay);

cusum = max(0, (residual(hospCountsDay) - k + cusum));

if (hospCountsDay <= 100 && cusum >= h)
    cusum=0;
elseif (hospCountsDay > 100)
    runLengthCounter = runLengthCounter + 1;
end

hospCountsDay = hospCountsDay + 1;
end % end while (cusum < h) loop. An alarm has occurred.
```

```
alarmCount = alarmCount + 1;
TFSallCusum56(alarmCount) = runLengthCounter;

if (runLengthCounter <= duration(durationIndex))
    TFScusum56(alarmCount) = runLengthCounter;
else
    TFScusum56(alarmCount) = -99; %did not catch outbreak by its end
end

%TFS(alarmCount);
runLengthCounter = 0; % after "alarm", reset runLengthCounter and cusum
cusum = 0;
end % end dummy for-loop

alarmCount=0; % reset previous alarmCount, reset runningSum stuff
runningSumTFS=0;
runningSumTFSCounter=0;
runningSumTFSall=0;
runningSqdSumTFScusum56 = 0;

for (dummy2 = 1:1:numLoops)
    if (TFScusum56(dummy2) > 0)
        runningSumTFS = runningSumTFS + TFScusum56(dummy2); %only adding
positive TFS values
        runningSumTFSCounter = runningSumTFSCounter + 1;
    end % end if TFS(dummy2)>0
runningSumTFSall = runningSumTFSall + TFSallCusum56(dummy2);
end % end dummy2 for-loop

averageTFScusum56(durationIndex) = runningSumTFS/runningSumTFSCounter;

for (dummy23 = 1:1:numLoops)
    if (TFScusum56(dummy23) > 0)
        runningSqdSumTFScusum56     =     runningSqdSumTFScusum56     +
(averageTFScusum56(durationIndex) - TFScusum56(dummy23))^2;
    end
end
seAverageTFScusum56(durationIndex)                                      =
(sqrt(runningSqdSumTFScusum56/(runningSumTFSCounter                     -
1)))/sqrt(runningSumTFSCounter);

averageTFSallCusum56(durationIndex) = runningSumTFSall/numLoops;
fractionMissedcusum56(durationIndex)        =        (numLoops        -
runningSumTFSCounter)/numLoops;

end % end durationIndex for-loop

%START                                                                   C1
METHOD_____________________________________________________________
sdMovingAvg = 999999; %this will be reset before it is used (SD required
to be > 0 near line 330)

for (durationIndex=1:1:7)
```

```
runningSumTFSc1=0;
runningSumTFSCounterc1=0;

alarmCountC1 = 0;
runLengthCounterC1 = 0;
TFSc1(1)=0;
c1h = MYc1h;
c1Statistic = 0;

for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:7)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var));  %  an  observation
on day "hospCountsDay"
end

hospCountsDay = 8;
```

```
while (c1Statistic < c1h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

if       (hospCountsDay       >       100       &&       hospCountsDay
<=(100+(duration(durationIndex)+1)/2))
    outbreak                                              =(hospCountsDay-
100)*magnitude/((duration(durationIndex)+1)/2);
elseif    (hospCountsDay    >    (100+(duration(durationIndex)+1)/2)    &&
hospCountsDay < (100+duration(durationIndex)))
    outbreak       =       magnitude       -       (hospCountsDay-100-
((duration(durationIndex)+1)/2))*magnitude/((duration(durationIndex)+1)/
2);
else
    outbreak=0;
end

hospCounts(hospCountsDay) = max(0,ceil(meanie + dayeffect + rand_var +
outbreak));
```

```
movingAvg(hospCountsDay)      =      (hospCounts(hospCountsDay-7)      +
hospCounts(hospCountsDay-6)     +     hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4)     +     hospCounts(hospCountsDay-3)     +
hospCounts(hospCountsDay-2) + hospCounts(hospCountsDay-1))/7;

if (length(movingAvg) >= 14) %need 7 (nonzero) values for an average
    if (sdMovingAvg > 0)
        oldsdMovingAvg = sdMovingAvg;
    end
    sdMovingAvg        =        ((((hospCounts(hospCountsDay-7)        -
movingAvg(hospCountsDay-7))^2)    +    ((hospCounts(hospCountsDay-6)    -
movingAvg(hospCountsDay-6))^2)    +    ((hospCounts(hospCountsDay-5)    -
movingAvg(hospCountsDay-5))^2)    +    ((hospCounts(hospCountsDay-4)    -
movingAvg(hospCountsDay-4))^2)    +    ((hospCounts(hospCountsDay-3)    -
movingAvg(hospCountsDay-3))^2)    +    ((hospCounts(hospCountsDay-2)    -
movingAvg(hospCountsDay-2))^2)    +    ((hospCounts(hospCountsDay-1)    -
movingAvg(hospCountsDay-1))^2))/6)^0.5;
    if (sdMovingAvg==0)
        sdMovingAvg = oldsdMovingAvg;
    end
    c1Statistic         =         (hospCounts(hospCountsDay)         -
movingAvg(hospCountsDay))/sdMovingAvg;
    if (hospCountsDay <= 100 && c1Statistic >= c1h)
        c1Statistic=0;
    elseif (hospCountsDay > 100)
        runLengthCounterC1 = runLengthCounterC1 + 1;
    end
end

hospCountsDay = hospCountsDay + 1;
end % end while (c1Statistic < c1h) loop, an alarm has occurred

alarmCountC1 = alarmCountC1 + 1;
TFSallC1(alarmCountC1) = runLengthCounterC1;

if (runLengthCounterC1 <= duration(durationIndex))
    TFSc1(alarmCountC1) = runLengthCounterC1;
else
    TFSc1(alarmCountC1) = -99; %did not catch outbreak by its end
end

runLengthCounterC1 = 0; % after "alarm", reset runLengthCounter and
cusum
c1Statistic=0;

end % dummy's for loop

alarmCountC1=0; % reset previous alarmCount, reset runningSum stuff
runningSumTFSc1=0;
runningSumTFSCounterc1=0;
runningSumTFSall=0;
runningSqdSumTFSc1 = 0;

for (dummy2 = 1:1:numLoops)
```

```
    if (TFSc1(dummy2) > 0)
        runningSumTFSc1 = runningSumTFSc1 + TFSc1(dummy2);
        runningSumTFSCounterc1 = runningSumTFSCounterc1 + 1;
    end
runningSumTFSall = runningSumTFSall + TFSallC1(dummy2);
end

averageTFSc1(durationIndex) = runningSumTFSc1/runningSumTFSCounterc1;

for (dummy23 = 1:1:numLoops)
    if (TFSc1(dummy23) > 0)
        runningSqdSumTFSc1     =     (averageTFSc1(durationIndex)     -
TFSc1(dummy23))^2;
    end
end
seAverageTFSc1(durationIndex)                                          =
(sqrt(runningSqdSumTFSc1/(runningSumTFSCounterc1                       -
1)))/sqrt(runningSumTFSCounterc1);

averageTFSallC1(durationIndex) = runningSumTFSall/numLoops;
fractionMissedc1(durationIndex)          =          (numLoops          -
runningSumTFSCounterc1)/numLoops;

end % end durationIndex for-loop

% START C2 METHOD_____________________________________________
sdMovingAvg = 999999;
for (durationIndex=1:1:7)
runningSumTFSc2=0;
runningSumTFSCounterc2=0;
runningSumTFSc3=0;
runningSumTFSCounterc3=0;

alarmCountC2 = 0;
alarmCountC3 = 0;
runLengthCounterC2 = 0;
runLengthCounterC3 = 0;
TFSc2(1) = 0;
TFSc3(1) = 0;
c2h = MYc2h;
c3h = MYc3h;
c2Statistic = 0;
c2StatTodayMinus2 = 0;
c2StatTodayMinus1 = 0;
c2StatToday = 0;
c3Statistic = 0;

for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:9)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
```

```
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
  elseif (j==8)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==9)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
  end

hospCounts(j)=max(0,ceil(meanie+rand_var+dayeffect));  % an  observation
on day "hospCountsDay"
end

hospCountsDay = 10;

while (c2Statistic < c2h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
```

```
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

if       (hospCountsDay       >       100       &&       hospCountsDay
<=(100+(duration(durationIndex)+1)/2))
    outbreak                                              =(hospCountsDay-
100)*magnitude/((duration(durationIndex)+1)/2);
elseif    (hospCountsDay    >    (100+(duration(durationIndex)+1)/2)    &&
hospCountsDay < (100+duration(durationIndex)))
    outbreak       =       magnitude       -       (hospCountsDay-100-
((duration(durationIndex)+1)/2))*magnitude/((duration(durationIndex)+1)/
2);
else
    outbreak=0;
end

hospCounts(hospCountsDay)=max(0,ceil(meanie+dayeffect+rand_var+outbreak)
);
movingAvg(hospCountsDay)      =      (hospCounts(hospCountsDay-9)      +
hospCounts(hospCountsDay-8)     +     hospCounts(hospCountsDay-7)     +
hospCounts(hospCountsDay-6)     +     hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4) + hospCounts(hospCountsDay-3))/7;

if (length(movingAvg) >= 16) %need 7 (16-9) days for an average
    if (sdMovingAvg > 0)
        oldsdMovingAvg = sdMovingAvg;
    end
```

```
sdMovingAvg = ((((hospCounts(hospCountsDay-9) - movingAvg(hospCountsDay-
9))^2) + ((hospCounts(hospCountsDay-8) - movingAvg(hospCountsDay-8))^2)
+  ((hospCounts(hospCountsDay-7)  -  movingAvg(hospCountsDay-7))^2)  +
((hospCounts(hospCountsDay-6)   -   movingAvg(hospCountsDay-6))^2)   +
((hospCounts(hospCountsDay-5)   -   movingAvg(hospCountsDay-5))^2)   +
((hospCounts(hospCountsDay-4)   -   movingAvg(hospCountsDay-4))^2)   +
((hospCounts(hospCountsDay-3) - movingAvg(hospCountsDay-3))^2))/6)^0.5;
    if (sdMovingAvg == 0)
        sdMovingAvg = oldsdMovingAvg;
    end
c2Statistic          =          (hospCounts(hospCountsDay)          -
movingAvg(hospCountsDay))/sdMovingAvg;
c2StatTodayMinus2 = c2StatTodayMinus1; % values on right are one day old
right now
c2StatTodayMinus1 = c2StatToday;
c2StatToday = c2Statistic;

    if (hospCountsDay <= 100 && c2Statistic >= c2h)
        c2Statistic=0;
    elseif (hospCountsDay > 100)
        runLengthCounterC2 = runLengthCounterC2 + 1;
    end
end

hospCountsDay = hospCountsDay + 1;
end % end while (c2Statistic < c2h) loop, an alarm in C2 has occurred

alarmCountC2 = alarmCountC2 + 1;
TFSallC2(alarmCountC2) = runLengthCounterC2;

if (runLengthCounterC2 <= duration(durationIndex))
    TFSc2(alarmCountC2) = runLengthCounterC2;
else
    TFSc2(alarmCountC2) = -99; %did not catch outbreak by its end
end

runLengthCounterC2  =  0;  %  after  "alarm",  reset  runLengthCounter  and
cusum
c2Statistic=0;

end % dummy's for loop

alarmCountC2=0; % reset previous alarmCount, reset runningSum stuff
runningSumTFSc2=0;
runningSumTFSCounterc2=0;
runningSumTFSallC2=0;

for (dummy2 = 1:1:numLoops)
    if (TFSc2(dummy2) > 0)
        runningSumTFSc2 = runningSumTFSc2 + TFSc2(dummy2);
        runningSumTFSCounterc2 = runningSumTFSCounterc2 + 1;
    end
runningSumTFSallC2 = runningSumTFSallC2 + TFSallC2(dummy2);
end
```

```
averageTFSc2(durationIndex) = runningSumTFSc2/runningSumTFSCounterc2;

for (dummy1 = 1:1:numLoops)
    if (TFSc2(dummy1) > 0)
        runningSqdSumTFSc2     =     (averageTFSc2(durationIndex)     -
TFSc2(dummy1))^2;
    end
end
seAverageTFSc2(durationIndex)                                          =
(sqrt(runningSqdSumTFSc2/(runningSumTFSCounterc2                       -
1)))/sqrt(runningSumTFSCounterc2);

averageTFSallC2(durationIndex) = runningSumTFSallC2/numLoops;
fractionMissedc2(durationIndex)          =          (numLoops          -
runningSumTFSCounterc2)/numLoops;

end % end durationIndex for-loop

TFSc2;
TFSallC2;
averageTFSc2
seAverageTFSc2 % for the given true signal
fractionMissedc2
averageTFSallC2

% START C3 METHOD_____________________________________________
for (durationIndex=1:1:7)
% runningSumTFSc2=0;
% runningSumTFSCounterc2=0;
runningSumTFSc3=0;
runningSumTFSCounterc3=0;

% alarmCountC2 = 0;
alarmCountC3 = 0;
% runLengthCounterC2 = 0;
runLengthCounterC3 = 0;
% TFSc2(1) = 0;
TFSc3(1) = 0;
% c2h = MYc2h;
c3h = MYc3h;
c2Statistic = 0;
c2StatTodayMinus2 = 0;
c2StatTodayMinus1 = 0;
c2StatToday = 0;
c3Statistic = 0;

for (dummy = 1:1:numLoops)
randStartDay = ceil(rand*365);

for (j = 1:1:9)
meanie=amplitude*(sin(2*pi*(j+randStartDay)/365))+ baseline;  % This is
the seasonal mean of the process
rand_var=randn*sigma;  % For large counts
```

```
%rand_var=lognrnd(1.0,sigma);  % For small counts

    %day effect: Monday=1 through Sunday=7
    if (j==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (j==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (j==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (j==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (j==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
  elseif (j==8)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
    elseif (j==9)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
  end

hospCounts(j)=max(0,ceil(meanie+dayeffect+rand_var));  % an observation
on day "hospCountsDay"
end

hospCountsDay = 10;

while (c3Statistic < c3h)

    %day effect: Monday=1 through Sunday=7 (in mod calcs)
    if (mod(hospCountsDay,7)+1==1)
        if (sigma>1) dayeffect= (0.1)*sigma;
        else dayeffect= (0.1)*sigmaZ;
        end
```

```
    elseif (mod(hospCountsDay,7)+1==2)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==3)
        if (sigma>1) dayeffect= (0.3)*sigma;
        else dayeffect= (0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==4)
        if (sigma>1) dayeffect= (0.2)*sigma;
        else dayeffect= (0.2)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==5)
        if (sigma>1) dayeffect= (0.0)*sigma;
        else dayeffect= (0.0)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==6)
        if (sigma>1) dayeffect= (-0.3)*sigma;
        else dayeffect= (-0.3)*sigmaZ;
        end
    elseif (mod(hospCountsDay,7)+1==7)
        if (sigma>1) dayeffect= (-0.5)*sigma;
        else dayeffect= (-0.5)*sigmaZ;
        end
    end

meanie=amplitude*(sin(2*pi*(randStartDay+hospCountsDay)/365))+ baseline;
rand_var=randn*sigma;  % For large counts
%rand_var=lognrnd(1.0,sigma);  % For small counts

if       (hospCountsDay       >       100       &&       hospCountsDay
<=(100+(duration(durationIndex)+1)/2))
    outbreak                                              =(hospCountsDay-
100)*magnitude/((duration(durationIndex)+1)/2);
elseif    (hospCountsDay    >    (100+(duration(durationIndex)+1)/2)    &&
hospCountsDay < (100+duration(durationIndex)))
    outbreak       =       magnitude       -       (hospCountsDay-100-
((duration(durationIndex)+1)/2))*magnitude/((duration(durationIndex)+1)/
2);
else
    outbreak=0;
end

hospCounts(hospCountsDay)                                                 =
max(0,ceil(meanie+dayeffect+rand_var+outbreak));
movingAvg(hospCountsDay)      =      (hospCounts(hospCountsDay-9)      +
hospCounts(hospCountsDay-8)     +     hospCounts(hospCountsDay-7)     +
hospCounts(hospCountsDay-6)     +     hospCounts(hospCountsDay-5)     +
hospCounts(hospCountsDay-4) + hospCounts(hospCountsDay-3))/7;

if (length(movingAvg) >= 16) %need 7 (16-9) days for an average
    if (sdMovingAvg > 0)
        oldsdMovingAvg = sdMovingAvg;
    end
```

```
sdMovingAvg = ((((hospCounts(hospCountsDay-9) - movingAvg(hospCountsDay-
9))^2) + ((hospCounts(hospCountsDay-8) - movingAvg(hospCountsDay-8))^2)
+  ((hospCounts(hospCountsDay-7)  -  movingAvg(hospCountsDay-7))^2)  +
((hospCounts(hospCountsDay-6)   -   movingAvg(hospCountsDay-6))^2)   +
((hospCounts(hospCountsDay-5)   -   movingAvg(hospCountsDay-5))^2)   +
((hospCounts(hospCountsDay-4)   -   movingAvg(hospCountsDay-4))^2)   +
((hospCounts(hospCountsDay-3) - movingAvg(hospCountsDay-3))^2))/6)^0.5;
    if (sdMovingAvg == 0)
        sdMovingAvg = oldsdMovingAvg;
    end
c2Statistic          =          (hospCounts(hospCountsDay)          -
movingAvg(hospCountsDay))/sdMovingAvg;
c2StatTodayMinus2 = c2StatTodayMinus1; % values on right are one day old
right now
c2StatTodayMinus1 = c2StatToday;
c2StatToday = c2Statistic;
end

if (length(movingAvg) >= 19) %need 3 C2 values for C3 (and all 3 c2Stat
values are != 0)
c3Statistic = max(0, (c2StatToday) - 1) + max(0, (c2StatTodayMinus1) -
1) + max(0, (c2StatTodayMinus2) - 1);
    if (hospCountsDay <= 100 && c3Statistic >= c3h)
        c3Statistic = 0;
    elseif (hospCountsDay > 100)
        runLengthCounterC3 = runLengthCounterC3 + 1;
    end
end

hospCountsDay = hospCountsDay + 1;
end % end while (c3Statistic < c3h) loop, an alarm in C2 has occurred

alarmCountC3 = alarmCountC3 + 1;
TFSallC3(alarmCountC3) = runLengthCounterC3;

if (runLengthCounterC3 <= duration(durationIndex))
    TFSc3(alarmCountC3) = runLengthCounterC3;
else
    TFSc3(alarmCountC3) = -99; %did not catch outbreak by its end
end

runLengthCounterC3 = 0; % after "alarm", reset runLengthCounter and
cusum
c3Statistic=0;

end % dummy's for loop, you have all your alarms in C3

alarmCountC3=0;

runningSumTFSc3=0;
runningSumTFSCounterc3=0;
runningSumTFSallC3=0;

for (dummy3 = 1:1:length(TFSc3))
```

```
    if (TFSc3(dummy3) > 0)
        runningSumTFSc3 = runningSumTFSc3 + TFSc3(dummy3);
        runningSumTFSCounterc3 = runningSumTFSCounterc3 + 1;
    end
runningSumTFSallC3 = runningSumTFSallC3 + TFSallC3(dummy3);
end

averageTFSc3(durationIndex) = runningSumTFSc3/runningSumTFSCounterc3;

for (dummy23 = 1:1:length(TFSc3))
    if (TFSc3(dummy23) > 0)
        runningSqdSumTFSc3     =     (averageTFSc3(durationIndex)     -
TFSc3(dummy23))^2;
    end
end
seAverageTFSc3(durationIndex)                                          =
(sqrt(runningSqdSumTFSc3/(runningSumTFSCounterc3                       -
1)))/sqrt(runningSumTFSCounterc3);

averageTFSallC3(durationIndex) = runningSumTFSallC3/numLoops;
fractionMissedc3(durationIndex)          =          (numLoops          -
runningSumTFSCounterc3)/numLoops;

end % end durationIndex for-loop

% OUTPUT-------------------------------------------------------------------
---
averageTFScusumOpt
seAverageTFScusumOpt
fractionMissedcusumOpt
averageTFSallCusumOpt

averageTFScusum56
seAverageTFScusum56
fractionMissedcusum56
averageTFSallCusum56

averageTFSc1
seAverageTFSc1
fractionMissedc1
averageTFSallC1

averageTFSc2
seAverageTFSc2 % for the given true signal
fractionMissedc2
averageTFSallC2

averageTFSc3
seAverageTFSc3 % for the given true signal
fractionMissedc3
averageTFSallC3
```

```
%SAVING COMMANDS (vectors saved together as a MATLAB file, and all are
loaded into workspace upon opening the file)
save(['scenario_'           num2str(scenarioNumber)           '_data'],
'averageTFScusumOpt',...
'seAverageTFScusumOpt',                         'fractionMissedcusumOpt',
'averageTFSallCusumOpt',...
'averageTFScusum56',   'seAverageTFScusum56',   'fractionMissedcusum56',
'averageTFSallCusum56',...
'averageTFSc1',          'seAverageTFSc1',          'fractionMissedc1',
'averageTFSallC1',...
'averageTFSc2',          'seAverageTFSc2',          'fractionMissedc2',
'averageTFSallC2',...
'averageTFSc3',          'seAverageTFSc3',          'fractionMissedc3',
'averageTFSallC3');

%PLOTTING COMMANDS

plotATFSgivenTrueSignal     =     figure('Name','ATFS     Given     True
Signal','NumberTitle','off');
plot(duration, averageTFScusumOpt, '-*k', duration, averageTFScusum56,
'-ok', duration, averageTFSc1, '--xk', duration, averageTFSc2, ':+k',
duration, averageTFSc3, '-.vk');
title({'Plot    of    ATFS    Given    True    Signal';    ['Scenario:    '
num2str(scenarioNumber)]});
xlabel('Outbreak Duration');
ylabel('Average TFS | True Signal');
set(gca, 'Xtick', 3:2:15)
axis('tight')
legend('CUSUM   (40)',   'CUSUM   (56)',   'C1','C2',   'C3',   'Location',
'NorthWest')
%axis([xmin, xmax, ymin, ymax])
axis([3, 15, 0, 8])
saveas(plotATFSgivenTrueSignal,['H:\THESIScode\FinalOutputAndPlots\plotA
TFSgivenTrueSignal_S' num2str(scenarioNumber) '.fig']);

plotFractionMissed              =              figure('Name','Fraction
Missed','NumberTitle','off');
plot(duration,       fractionMissedcusumOpt,       '-*k',       duration,
fractionMissedcusum56,   '-ok',   duration,   fractionMissedc1,   '--xk',
duration, fractionMissedc2, ':+k', duration, fractionMissedc3, '-.vk');
title({'Plot      of      Fraction      Missed';      ['Scenario:      '
num2str(scenarioNumber)]});
xlabel('Outbreak Duration');
ylabel('Fraction Missed');
set(gca, 'Xtick', 3:2:15)
axis('tight')
legend('CUSUM   (40)',   'CUSUM   (56)',   'C1','C2',   'C3',   'Location',
'SouthWest')
axis([3, 15, 0, 1])
saveas(plotFractionMissed,['H:\THESIScode\FinalOutputAndPlots\plotFracti
onMissed_S' num2str(scenarioNumber) '.fig']);
```

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Associate Professor R.D. Fricker, Jr.
   Naval Postgraduate School
   Monterey, CA

4. Chairman David Olwell
   Naval Postgraduate School
   Monterey, CA

5. Professor Annette Neu
   Naval Postgraduate School
   Monterey, CA

6. Professor Ellen Gordon
   Naval Postgraduate School
   Monterey, CA

7. Dr. Jerome Tokars
   Centers for Disease Control and Prevention
   Atlanta, GA

8. Dr. Henry Rolka
   Centers for Disease Control and Prevention
   Atlanta, GA

9. Ms. Lori Hutwagner
   Centers for Disease Control and Prevention
   Atlanta, GA

10. Ms. Heather Issvoran
    Naval Postgraduate School
    Monterey, CA