The Johns Hopkins University Applied Physics Laboratory (APL) Global Health Surveillance program focuses on all phases of the surveillance timeline—from prediction of a potential disease outbreak through response and communication efforts. SAGES (Suite for Automated Global Electronic bioSurveillance), which builds on years of experience with the ESSENCE system, is the cornerstone of APL’s work in global health diplomacy initiatives. This issue of the Johns Hopkins APL Technical Digest describes the development, implementation, and adaptation of the SAGES suite of tools; the process and challenges of making the tools open source; and potential new analytic models for early detection of disease outbreak.
<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>648</td>
<td>Global Health Surveillance—Guest Editor’s Introduction</td>
<td>Sheri H. Lewis</td>
</tr>
<tr>
<td>679</td>
<td>Development of Mobile Health Capabilities for Remote Data Collection in Resource-Limited Settings</td>
<td>Adjoa M. Poku and Aaron T. Katz</td>
</tr>
<tr>
<td>685</td>
<td>A Tailored Approach to Implementing Open-Source Electronic Disease Surveillance Tools</td>
<td>Shraddha V. Patel</td>
</tr>
<tr>
<td>690</td>
<td>An Overview of Open-Source Software Licenses and the Value of Open-Source Software to Public Health Initiatives</td>
<td>Erin N. Hahn</td>
</tr>
<tr>
<td>699</td>
<td>Releasing Tools for International Disease Surveillance as Open-Source Software: A Case Study</td>
<td>Raj J. Ashar</td>
</tr>
<tr>
<td>735</td>
<td>A Scalable Data Mining Approach for Providing Public Health with Disease Incidence Predictions Weeks in Advance</td>
<td>Anna L. Buczak, Erhan Guven, Steven M. Babin, Erin N. Hahn, David W. George, Yevgeniy Elbert, Liane C. Ramac-Thomas, Benjamin D. Baugher, Jacqueline S. Coberly, and Sheri H. Lewis</td>
</tr>
<tr>
<td>745</td>
<td>Public Health Applications in the Cloud</td>
<td>Wayne A. Loschen, Miles A. Stewart, and Joseph S. Lombardo</td>
</tr>
</tbody>
</table>
Global Health Surveillance—Guest Editor’s Introduction

Sheri H. Lewis

Improving the overall health of a given population has far-reaching effects not only on the economy and stability of that particular population but also on a global scale. Disease surveillance, a critical component in understanding and improving global health, is undergoing a revolution driven by advances in information technology. Recent years have seen vast improvements in the collection, analysis, visualization, and reporting of public health data. At the Johns Hopkins University Applied Physics Laboratory (APL), teams of software engineers, analysts, and epidemiologists have been working for more than 15 years to develop advanced electronic disease surveillance technologies. This issue of the Johns Hopkins APL Technical Digest describes the development and implementation of these technologies, the process and challenges of making the tools open source, and potential new analytic models for early detection of disease outbreak.

INTRODUCTION

People are particularly fearful of a pandemic caused by the global spread of a novel naturally occurring disease such as Middle East respiratory syndrome coronavirus or a pandemic influenza A strain. This fear is understandable given that novel diseases represent the unknown: How quickly and easily does the disease spread from person to person? Who is most susceptible to the disease? What is the likelihood that I or my family will die from the disease if we contract it? These fears are not unfounded, as infectious diseases know no geographic boundaries, and globalization has connected people and parts of the world today more than in any other time in our history. Social and mainstream media can amplify these fears. Many of us have the ability to search the Internet to find more information on a new disease than we can possibly comprehend and to see the ample chatter on social media sites.

Fortunately, the World Health Organization (WHO) sets requirements for member countries to report certain diseases. Before the severe acute respiratory syndrome...
(SARS) outbreak in 2003, the WHO International Health Regulations (IHR) required member countries to report only outbreaks of cholera, yellow fever, and plague. However, as witnessed during the SARS outbreak, a disease that was first identified in China and Hong Kong quickly spread to the United States and Canada, and the existing IHR did not adequately address the growing threat of emerging diseases in a globalized world. Not only did health authorities have to deal with challenging decisions related to slowing the spread of disease (such as by closing certain venues), but few jurisdictions had a way to easily assess the health of their populations. As a result, in 2005, the WHO adopted new IHR. These new IHR, commonly referenced as IHR 2005, require member countries to report any disease that may constitute a public health emergency of international concern. Additionally, the IHR stipulate, albeit at a high level, that countries will improve their capacity to detect, assess, communicate, and respond to public health threats. These new IHR 2005 went into effect in 2012, but to date many countries are still having difficulties complying.

Furthermore, in February 2014, numerous countries, including the United States, committed to the Global Health Security Agenda (GHSA), whose goal is to accelerate progress in the prevention of, detection of, and response to infectious disease threats over a five-year period. The GHSA specifically targets infectious diseases and includes the topics of novel disease propagation and globalization of trade and travel, in addition to highlighting concerns about increasing antimicrobial drug resistance as well as the threat of disease from the accidental release, theft, or illicit use of a disease agent.

The latter point—accidental release, theft, or illicit use of a disease agent—also draws attention to the growing field of “do-it-yourself (DIY) biology” as well as to research ethics. DIY biology, while often well intentioned and practiced as an after-hours hobby by many trained researchers in academia and corporations, is an unregulated or self-regulated activity. Some in the scientific community fear this DIY work could result in inadvertent or malicious development and release of a biological weapon, heightening the need for increased surveillance capacity.

Also of concern is the ability of scientists to replicate diseases in such a way that modifies their lethality. Although it is truly awe inspiring that science has come so far, charged discussions result when researchers want to share their findings worldwide by publishing their results in peer-reviewed scientific publications, which is the norm for those working in the field. In recent years there has been debate over the ethics of publishing research that exposes the “recipes” for replicating viruses and changing the virulence of viruses such as avian influenza (H5N1). Many fear that such publications could provide those with nefarious intent with the information needed to develop a “superbug.” As scientists continue to make advances in the rapid identification and experimentation of novel pathogens, this debate will likely continue for years at the highest levels of government and academia.

The scenarios described above underscore the importance of the global health surveillance program at the Johns Hopkins University Applied Physics Laboratory (APL). The program operates within the space of predicting, detecting, and responding to infectious disease within the human population, both domestically and internationally. That being said, effective electronic disease surveillance systems represent just one part of a much larger effort to protect the global population against the threat of emerging and re-emerging infectious diseases. APL’s role in this domain is to combine public health expertise with information technology (IT) and analytics to achieve effective and sustainable solutions for public health officials worldwide.

At APL, teams of software engineers, analysts, and epidemiologists have been working for more than 15 years to develop advanced electronic disease surveillance technologies. As has been discussed in previous issues of the *Johns Hopkins APL Technical Digest*, public health entities the world over have lacked the benefit of IT solutions until as recently as 10 years ago. Today in the United States, public health professionals in most states have the capability to collect, analyze, and visualize data to assess the health of their communities. Additionally, there are public health “super users” who are taking electronic disease surveillance to the next level and using the data to investigate many types of public health issues such as chronic disease, injury, and mental health. However, in many places around the world, particularly in resource-limited environments, public health officials are on the front line of defense against the next epidemic—without the benefit of state-of-the-art electronic disease surveillance capabilities.

**THE ARTICLES**

In this issue of the Digest, we highlight the ever-increasing role APL is playing in the area of global health surveillance.

The first article, by Feighner et al., highlights the Suite for Automated Global Electronic bioSurveillance (SAGES), the cornerstone of the current global health surveillance program that builds on the successful Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE) program. This article describes not only the technology being used to build the capacity of U.S. partner countries but also the process by which APL works with its long-term sponsor, the Global Emerging Infections Surveillance
and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS). AFHSC-GEIS is committed to improving infectious disease surveillance worldwide for all DoD health care beneficiaries as well as for the global community. In this role it is guided by four goals: (i) conduct surveillance and outbreak response activities, (ii) improve the capacity of partner countries to perform disease surveillance, (iii) support research initiatives that will result in new capabilities in support of force health protection, and (iv) assess and communicate through its worldwide network.

The main analysis and visualization tool within the SAGES tool kit is OpenESSENCE (OE), an open-source version of the ESSENCE system. OE combines advanced analytics with a flexible platform to collect and visualize data that vary greatly in both volume and granularity. First, Campbell et al. highlight the OE features and functionality that public health users have deemed critical for interpreting health data at both the individual and aggregate levels. Burkom et al. then describe the analytics used in the OE platform and the work that went into refining them for partner countries whose data may differ vastly from those collected in countries such as the United States. The analytics enable early event detection and signal statistical anomalies that warrant further epidemiological evaluation without causing alerting fatigue to the public health monitors.

Data collection remains a challenge in resource-limited environments that may lack robust Internet capabilities or IT infrastructure. As a result, the SAGES tool kit includes technologies to enable data collection via cell phones. Poku and Katz describe the use of mobile technology in the SAGES platform and the larger movement in the mobile health, also known as mHealth, community.

Technology development is just one piece of the solution. Many of the complexities of disease surveillance involve implementation of the software with the partner countries. Just as in the U.S. public health system, the skill of the public health and IT workforce varies from site to site, and as a result, the technology implementation strategy must be adaptable and flexible to maximize the likelihood of success. Shraddha Patel describes the SAGES implementation process used by the APL and sponsor team.

Another element of the overall solution is making the tools available to the widest array of users who will not only use the tools but also improve them. To meet this need, the U.S. government is increasingly adopting the use of open-source products. By definition, open-source software refers to software products whose code is available for use and/or modification by anyone. Although this concept allows for greater flexibility and collaboration, it is not without challenges. Erin Hahn describes the drive behind open-source software, particularly in the health community, and the decisions developers and the sponsor community must consider. Open-source software is often desirable in resource-limited settings because it lacks licensing fees. Other desirable features of open-source software are the ability for users to add to the code base and its transparency, which reassures adopters that their data are not being released unknowingly to third parties. Raj Ashar describes the issues the sponsor and APL team encountered during the open-source release process for the SAGES software tools.

As was mentioned previously, social media is an increasingly popular mode of communication. It is one that holds great promise for the early detection of infectious disease. However, before data derived from social media can be effectively used in disease surveillance systems, work is needed to understand whether and how these data are correlated with more traditional health data. Coberly et al. describe a study in which they looked at Twitter data in the Philippines and correlated them with data gathered by a partner country.

Just as Coberly et al. are looking for ways to enhance current surveillance capabilities, the adaptability of a flexible analysis and visualization engine such as OE enhances public health professionals’ capabilities. Campbell et al. describe how they have applied the OE framework for the AFHSC-GEIS respiratory disease surveillance network. Although partner countries currently use OE to monitor for infectious disease outbreaks, the respiratory dashboard, developed on the same framework, is being used as a tool to upload respiratory diagnostic test data across the AFHSC-GEIS partner network to provide a global picture of respiratory disease.

Again looking to advance the tools and capabilities, in the final article of this issue Buczak et al. describe their work to improve public health monitors’ ability to detect a potential disease outbreak before it becomes a reality. The hope is that monitors will use these novel prediction methodologies in conjunction with electronic disease surveillance tools, thereby alerting health officials to more closely monitor for a potential disease risk and to take mitigating steps before a disease outbreak occurs.

Finally, Loschen et al. discuss the work done to date to use the cloud environment to host ESSENCE here in the United States. While cloud computing is not currently a viable solution in austere environments, it holds great potential for the future given how rapidly IT capacity is increasing worldwide.

**FINAL THOUGHTS**

Nations have long used health as a diplomatic tool. Improving the overall health of a given population has far-reaching effects on the economy and stability of a population, whether it is as small as a village or as large as an
entire country. Indeed, improving the health of one population has effects on populations worldwide. Through the efforts of many U.S. government agencies, some of which have been tackling this problem long before the GHSA, in concert with partner countries and international organizations, the face of global health diplomacy is slowly changing. The U.S. government, now more than ever, is strongly committed to global health diplomacy as a means to improve not only health but also global economic and political stability. It believes, as do members of the APL team, that a “public health emergency anywhere is a public health emergency everywhere.”

The Author

Sheri H. Lewis is the Global Health Surveillance Program Manager and a member of the Principal Professional Staff at APL. She works closely with both the U.S. and international public health communities on the development and implementation of electronic disease surveillance systems and mobile health applications. Her current efforts focus on the deployment of such systems in resource-limited settings and the development of advanced analytics for the prediction and modeling of emerging infectious diseases. Ms. Lewis also contributes her knowledge of public health and disease surveillance to the larger biosurveillance problem space. Her e-mail address is sheri.lewis@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
Disease surveillance, the foundation of public health practice, is undergoing a revolution driven by advances in information technology. The past 15 years have seen vast improvements in the collection, analysis, visualization, and reporting of public health data. Resource-limited countries have lagged behind because of challenges in information technology infrastructure and public health resources. The Suite for Automated Global Electronic bioSurveillance (SAGES) is a collection of modular, open-source software tools designed to meet the challenges of electronic disease surveillance in resource-limited settings. Individual SAGES tools may be used in concert with existing surveillance applications or en masse for an end-to-end biosurveillance capability. This flexibility allows for the development of an inexpensive, customized, and sustainable disease surveillance system. The ability to rapidly assess anomalous disease activity may lead to more efficient use of limited resources and better compliance with World Health Organization International Health Regulations.

INTRODUCTION

Disease surveillance was defined by Langmuir in 1963 as “the continued watchfulness over the distribution and trends of incidence through the systematic collection, consolidation and evaluation of morbidity and mortality reports and other relevant data,” with the “regular dissemination of the basic data and interpretations to all who have contributed and to all others who need to know.”1 Thacker expanded and refined this definition in 1988 when he wrote, “Public health surveillance is the ongoing systematic collection, analysis and interpretation of outcome-specific data for use in the planning, implementation, and evaluation of public health practice.”2 This linking of public health surveillance with the evaluation of public health practice emphasizes its primary purpose—to direct the expenditure of limited public health resources in a manner that yields the greatest return on investment.3
Public health surveillance involves clearly defining events of public health interest, counting those events, and then analyzing those events with respect to person, place, and time. For example, the U.S. Centers for Disease Control and Prevention case definition for an influenza-like illness is “fever (temperature of 100°F [37.8°C] or greater) and a cough and/or sore throat without a KNOWN cause other than influenza.”1 Patients meeting the case definition of influenza-like illness are counted by sex and age category (person) and characterized by site (place) and date of onset (time). This conceptually simple process not only characterizes the level, distribution, and spread of influenza-like illness in the community but also suggests useful information such as determinants of disease transmission, possible mitigating strategies, and future prevention strategies. Public health surveillance may be performed on all patients, so-called universal surveillance, or performed at designated sites felt to be representative of the population as a whole, so-called sentinel surveillance. Surveillance may also be described as active, when public health officials contact health care providers, or passive, when public health officials rely on reports from health care providers. A wide variety of data sources are used in public health surveillance, including vital statistics, health reports, hospital records, laboratory reports, outpatient visits, registries, and health surveys.

Disease surveillance is commonly recognized for its ability to detect disease outbreaks. Simply put, unless the baseline level of disease is well understood, it is difficult to identify disease levels significantly in excess of normal.2 This is an important function, and the early detection of anomalous disease events, particularly the intentional release of pathogens, has received much recent attention. Critics point out that disease surveillance, particularly syndromic surveillance, may not catch small outbreaks of disease that remain hidden in the background noise and also note that diseases with shocking presentations, such as hemorrhagic fevers, are generally identified by health care providers.3 Nevertheless, disease surveillance plays a critical role in the detection of disease outbreaks.1–3,6,7 Importantly, in the case of small- to medium-size outbreaks distributed over a wide geographic area—now common because of large, centralized food-processing plants—coordinated disease surveillance identifies problems that might otherwise go unnoticed in each local jurisdiction.6 Disease surveillance accomplishes several additional important functions to direct the sage practice of public health.2 Disease surveillance identifies and quantifies the diseases most burdensome to a population. How a disease spreads through the population of interest and how it affects individuals over time can both be carefully documented by disease surveillance. Importantly, disease surveillance is used to evaluate public health interventions, identifying effective and ineffective public health practices. Disease surveillance can suggest hypotheses, direct research, and detect changes in the practice of clinical (or veterinary) medicine over time. Effective disease surveillance, though not always exciting, is the foundation of successful public health practice.

For centuries, disease surveillance was a paper-based process. In the 1990s, with the emergence of inexpensive, powerful information technology tools, disease surveillance became an electronic process in wealthy countries.7 Incorporating information technology advances led to startling improvements in the timeliness of reporting and the sophistication of data analysis. Such systems have become versatile tools in health departments in the United States, and electronic disease surveillance holds promise to improve health security in resource-limited environments.8–12 Epidemiologists using electronic disease surveillance not only have the potential to detect anomalous disease activity earlier than those using traditional laboratory-based surveillance, but they also have the ability to monitor the health of their community in the face of a known threat.10–12 More than a decade ago, in collaboration with the DoD, the Johns Hopkins University Applied Physics Laboratory (APL) developed the Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE).12 ESSENCE is currently being used by the DoD, Veterans Health Administration, and numerous state and local health departments in the United States.12 ESSENCE allows for essentially real-time data collection, leading to timely anomaly detection with dynamic, sophisticated data analysis, visualization, and reporting. In addition, electronic disease surveillance systems are able to automatically ingest large amounts of preexisting electronic data streams for analysis. These data sources, such as insurance claims, pharmaceutical data, and commercial sales, differ from traditional sources, such as health data from medical treatment facilities, yet they often have content that is relevant to public health.12 The current Suite for Automated Global Electronic bioSurveillance (SAGES) initiative leverages the experience gained in the development of ESSENCE, and the analysis and visualization components of SAGES are built with the same features in mind.

Emerging and reemerging infectious diseases are among the most serious threats to global public health.13,14 The World Health Organization (WHO) has identified more than 1100 epidemic events worldwide in the last 10 years alone.15 The emergence of the novel 2009 influenza A (H1N1) virus and the SARS coronavirus in 2002 has demonstrated how rapidly pathogens can spread worldwide.13–16 This infectious disease threat, combined with a concern over man-made biological or chemical events, spurred WHO to update its International Health Regulations (IHR) in 2005.17 The new 2005 IHR, a legally binding instrument for all 194 WHO-member countries, significantly expanded
the scope of reportable conditions and are intended to help prevent and respond to global public health threats. Specifically, the IHR require strengthening disease detection and response capacities in order to report, within 24 hours of assessment, any public health event of international concern. SAGES, an electronic biosurveillance initiative described herein, aims to improve local public health surveillance and IHR compliance, with particular emphasis on resource-limited settings.

**SAGES AND ITS TOOLS**

The U.S. Armed Forces Health Surveillance Center is committed to enhancing electronic disease surveillance capacity in resource-limited settings around the world. In 2008, its Global Emerging Infections Surveillance and Response System Division (AFHSC-GEIS) entered into a robust collaboration with the APL to create SAGES. Aware of the work of others underway on individual surveillance systems components (e.g., collection of data by cell phones), we focused our efforts on the integration of inexpensive, interoperable disease surveillance software tools that facilitate regional public health collaborations.18

SAGES tools are organized into four categories: (i) data collection, (ii) analysis and visualization, (iii) communications, and (iv) modeling/simulation/evaluation (Fig. 1). Within each category, SAGES offers a variety of tools compatible with surveillance needs and different types or levels of information-technology infrastructure. In addition to offering flexibility in their selection, the tools also do not require a fixed database format. For example, rather than requiring an existing database to adapt to the tool, the SAGES database tools adapt to all Java database–compliant formats, a mainstay of enterprise information-technology development for decades. Lastly, the SAGES tools are built in a modular nature, which allows the user to select one or more tools to enhance an existing surveillance system or to use the tools en masse for an end-to-end electronic disease surveillance capability. Thus, each locality can select tools from SAGES on the basis of their needs, capabilities, and existing systems to create a customized electronic disease surveillance system.

**Data Acquisition**

Rapid data acquisition is arguably the most challenging aspect of establishing a successful electronic disease surveillance system.10–12 In resource-limited settings, it is not unusual for disease surveillance to be paper based, with data sent to upper echelons only when a courier happens to be going that way. This system of data acquisition often results in a data lag—the time from health event to analysis—of weeks to months. Because the ability to mitigate the effects of an infectious disease outbreak strongly depends on early recognition and response, minimizing data lag is of utmost importance. In resource-limited settings, it is imperative to select technologies that are both easy to incorporate into existing health services and sustainable with little or no additional financial investment. The approach should allow customizable data collection, enable multiple data streams collected in different ways, and be scalable on the basis of needs.8,9 Novel data collection tools included within SAGES are web forms, short message service (SMS) texting using simple cell phones, forms on Android smartphones transmitted by SMS, digital logbooks, and forms on tablets using Wi-Fi systems. Where appropriate, other collection methods such as e-mail and secure file transfer protocol can be applied as well.

**Analysis and Visualization**

As previously discussed, the SAGES analysis and visualization tools are built on the features and functionality of the mature ESSENCE system. The enterprise ESSENCE system requires a high-speed Internet connection, relies on
SAGES ELECTRONIC DISEASE SURVEILLANCE

Communications

SAGES tools can facilitate compliance with 2005 IHR reporting requirements and allow the sharing of actionable information across jurisdictional boundaries. Sharing of patient-level data across regional boundaries is generally not realistic and often not helpful because local public health entities are usually best suited to interpret local events. Once the data have been transformed into actionable information, however, it may be immensely valuable to share that information with other countries in the region. Dissemination of this type of information may aid in the interpretation of regional events and helps foster better, lasting public health collaborations. SAGES data visualization and reporting products are exportable into common image formats. Planned data-sharing tools will allow each SAGES user to control the type and level of detail of information shared with each recipient (“role-based access”) and also whether the information sharing is manual or automated.19 This capability does not compete with the WHO Global Outbreak Alert & Response Network (GOARN) but compliments it for organizations, such as Ministries of Defense, that wish to communicate

Figure 2. Time series (number of events by day) display in OpenESSENCE, with alerting to anomaly detection (synthetic data).
within or outside of GOARN. APL SAGES personnel and AFHSC-GEIS are currently in discussions with the WHO and WHO regional offices over the use of SAGES tools in support of WHO initiatives. Lastly, and importantly, the data collected using SAGES software remain under the sole control of the user at all times.

**Modeling/Simulation/Evaluation**

The APL SAGES program has been active in modeling infectious diseases for the U.S. military. The Pandemic Influenza Policy Model (PIPM), an early proprietary software project, allowed military public health officers and planners to vary attributes of respiratory

---

**Figure 3.** Chloropeth map with level of disease activity indicated by color and district (synthetic data).

**Figure 4.** Dynamic generation of data tables for analysis and reporting (synthetic data).
pathogens such as virulence, transmissibility, and susceptibility while also varying public health interventions.\textsuperscript{20} Once configured by the user, the PIPM then modeled the transmission of disease from individual to individual in what is termed an “agent-based” model. Using serial PIPM model runs, users could evaluate the effect of different interventions, as well as the effect of the timing of those interventions (e.g., the effect of closing schools upon the spread of influenza on a military base).

APL has sponsored several exercises to train users, test surveillance system features, and evaluate outbreak response. We have developed a number of methods for developing simulated outbreaks, both natural and man-made, which can then be “injected” into a simulated surveillance database for exercise purposes.\textsuperscript{21} Past exercises have included simulations of naturally occurring diseases such as pandemic influenza and deliberate release of biological warfare agents such as anthrax, plague, and tularemia. Participants in these exercises have ranged from local health departments to combined (international) military forces.

In addition to agent-based infectious disease modeling, the SAGES program has other techniques for predictive disease modeling. The PRedicting Infectious Disease Scalable Model (PRISM) is the most recently developed proprietary disease prediction tool.\textsuperscript{22} PRISM is not an agent-based model but rather uses novel data-mining techniques to predict the future incidence of disease. Early results have demonstrated impressive validity with dengue fever in the Republic of Peru and malaria in the Republic of Korea. It is our desire to explore possible collaborations among our disease modelers and users of electronic disease surveillance systems such as SAGES, with an ultimate goal of using disease surveillance data to improve future disease prediction efforts.

Only those diseases with the highest burdens on the population should be followed, and surveillance systems should be periodically evaluated to determine their usefulness.\textsuperscript{1,2,7} At this time, APL is working with AFHSC-GEIS to develop tools for the evaluation of SAGES surveillance systems. Several authors have described the evaluation of electronic disease surveillance systems.\textsuperscript{23–27} We intend to perform such evaluations of SAGES surveillance systems and investigate the feasibility of developing automated evaluation tools in the future.

CONCLUSION

Effective and efficient disease surveillance is critical for the economic development of nations. Globalization, particularly rapid international travel, has led to the maxim, “A public health emergency anywhere is a public health emergency everywhere.” The SAGES project is intended to enhance electronic disease surveillance capacity in resource-limited settings around the world, hopefully leading to economic development in resource-limited nations and improved global health. We have combined electronic disease surveillance tools developed at APL with other open-source, interoperable software tools to create SAGES. We believe this suite of tools will facilitate local and regional electronic disease surveillance, regional public health collaborations, and international disease reporting. SAGES tools are currently undergoing pilot testing in locations in Africa, Southeast Asia, and South America and will be offered to other interested countries around the world.

ACKNOWLEDGMENTS: We acknowledge the support of Tim Frey, Supervisor of the Bio-Threat Awareness Systems Group in APL’s Asymmetric Operations Sector. Joe Lombardo, an early and established authority in electronic disease surveillance, has provided helpful guidance to the SAGES program. Anna L. Buczk provided helpful comments on the manuscript.

REFERENCES

6Council to Improve Foodborne Outbreak Response (CIFOR), Guidelines for Foodborne Disease Outbreak Response, Council of State and Territorial Epidemiologists, Atlanta (2009).
Brian H. Feighner is the Senior Public Health Physician on the SAGES team. Timothy C. Campbell is the technical lead in information technology for the SAGES project. Aaron T. Katz is Assistant Group Supervisor of the Bio-Threat Awareness Systems Group and the SAGES Project Manager. Richard A. Wojcik is the Bio-Threat Awareness Systems Group’s Chief Engineer. Jacqueline S. Coberly is the technical lead in epidemiology for the SAGES project, and Shraddha V. Patel is the technical lead in education and training. Wayne A. Loschen leads APL’s enterprise ESSENCE technical efforts. Sheri H. Lewis is the Global Health Surveillance Program Manager in the Asymmetric Operations Sector’s Homeland Protection Program Management Office. For further information on the work reported here, contact Brian Feighner. His e-mail address is brian.feighner@jhuapl.edu.
Electroni c biosurveillance systems can improve the timeliness of public health data collection, aid in the early detection of disease outbreaks, and enhance situational awareness. As part of the Suite for Automated Global Electronic bioSurveillance (SAGES) program, the Johns Hopkins University Applied Physics Laboratory (APL) developed an open-source software tool called OpenESSENCE. OpenESSENCE provides “out-of-the-box” web-based data entry, analysis, and reporting that may significantly improve global disease surveillance, including surveillance in a wide range of resource-limited settings. Local health clinics have recognized that this new technology may help their countries comply with the World Health Organization revised International Health Regulations (IHR 2005) and prevent or mitigate disease outbreaks. This article briefly reviews OpenESSENCE and describes updates made during the last 2 years.

INTRODUCTION

Emerging public health threats can originate from almost anywhere in the world and often spread rapidly through populations, especially those lacking prior exposure to the disease in question. Early detection of these disease outbreaks is considered so important to global health that the World Health Organization issued revised International Health Regulations (IHR 2005) to enhance global cooperation and protection from emerging health threats. Recent advances in technology are helping member nations comply with IHR 2005. In 2010, Ashar et al. reviewed information and communications technologies developed for electronic health data capture and assessed their use in resource-limited settings. To complement and enhance traditional public health surveillance, new syndromic surveillance systems were developed. These systems use electronic, nontraditional prediagnostic health information as early indicators of disease incidence to detect potential disease outbreaks in populations. More than a decade ago, the Johns Hopkins University Applied Physics Laboratory (APL) began developing a syndromic surveillance system called the Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE). ESSENCE is a Java-based web application used to moni-
tor the health of populations, detect disease outbreaks early, and help prevent the spread of disease. “Enterprise” ESSENCE is the version of ESSENCE used by local and regional public health departments in the United States and by the U.S. Departments of Defense and Veterans Affairs. It allows public health professionals to collect and analyze many types of nontraditional, prediagnostic data and to use multiple anomaly detection algorithms to flag unusually high counts of disease indicators or issue alerts. ESSENCE enables users to view, parse, plot, and map results and to share selected information with other users. Enterprise ESSENCE relies on electronic health data feeds and is best suited for areas with stable Internet access.

As part of the APL Suite for Automated Global Electronic BioSurveillance (SAGES) program, the Global Emerging Infections Surveillance and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS), funded the development of a new version of ESSENCE, called OpenESSENCE (OE), for global use including use in resource-limited settings. As the name implies, OE is an open-source application that includes key features of Enterprise ESSENCE but can be used either via the Internet or as a standalone system. Whereas Enterprise ESSENCE uses proprietary software, OE uses open-source software. Doing so provides several advantages in managing health data and performing medical surveillance in a variety of global communities. OE’s open-source design makes it easier for global users to maintain and sustain their systems by providing better quality assurance, low cost for acquisition and maintenance, and extensive user input on requirements, usage, and adaptability. In 2012, Campbell et al. described the early development of OE and how it was being adapted globally to provide early warning and international awareness of disease outbreaks. This article will briefly review the basic features of OE and describe OE developments that have occurred since the Campbell et al. article was written.

OE OVERVIEW AND UPDATES

Beginning early in the development of OE, APL sought input from a variety of public health departments in resource-limited countries and found that there was consensus on the desire for a new kind of biosurveillance system: an open-source system that would place a minimum burden on data providers, allow the user to tailor the system to their needs, and provide sustainability by allowing the user to control their data and easily maintain their system.

OE is designed to be customizable, dynamic, and flexible, allowing for extension and reconfiguration, to reduce the cost of development and maintenance. To further this goal, OE is deployable using only open-source software that includes industry-standard technologies (e.g., Java EE, Apache Tomcat, Spring, MySQL, PostgreSQL, PostGIS, GeoServer, GeoExt, and OpenLayers). OE uses the Spring (http://projects.spring.io/spring-security/) framework and Groovy (http://groovy.codehaus.org) Java language extensions to create a modular, component-based application design, which allows for improved testability of components and isolation of problems. Repurposing and reusing the application is easier and more likely with a modular design that mitigates the need to rebuild the application to incorporate adjustments or enhancements. To support confidentiality and integrity of data, OE provides user-configurable access control. Because it is intended for global use, OE features extensive localization support that allows for the adaptation of internationalized software for a specific region or language. OE includes a plug-in application programming interface that allows user access to different detection algorithms. Results are provided in a format similar to those in Enterprise ESSENCE and include graphs, charts, and tabular data on case reports, as well as geographic maps of individual illness reports.

Initial User Interface and Data Input

Health indicator data are those data that reflect determinants of health and health outcomes and may come in a wide variety of formats that include numeric and textual data. OE is designed to accept input of a variety of health indicator data. OE contains a built-in data entry module so that data can be entered directly into a server running OE or via the web by multiple, geographically distributed users. The software is configurable to accept data from different databases and data sources and can accept combinations of different data sources (e.g., joins of multiple tables). At least two

Figure 1. Example of simple structured SMS message for entering data in OE.
methods of data input are provided: (i) a cellular telephone Short Message Service (SMS) interface; and (ii) a web interface for those with Internet access. Both can be configured to accept data on the basis of individual collection needs.

In most countries, even those with limited resources, cellular telephone use is pervasive. Therefore, cellular services with the ability to transmit text messages using SMS can be used as a way of collecting health data and entering those data directly into an electronic surveillance system.\textsuperscript{3,14} APL-developed SMS systems are currently used for OE data entry in multiple countries including Cambodia, Nicaragua, Cameroon, Kenya, and Uganda. Patient-level data, such as age, sex, presenting signs and symptoms, date of symptom onset, and local clinic identifier, are sent via SMS from local health clinics to a “receiver” Android smartphone. This receiver phone is connected to a computer that may be located at the central public health office, and the information is transferred to the computer using standard extract, transform, and load synchronization processes. Users can manually construct SMS messages to submit the data (Fig. 1), but this technique is tedious and error-prone. Additionally, the SMS standard\textsuperscript{15} limits messages to a maximum of 160 characters. An alternative to manual SMS messaging is the form-based data entry application, SAGES mCollect, for Android smartphones. Using mCollect, which is powered by Open Data Kit Collect (http://opendat Kit.org/use/collect), data are entered into a graphical, predefined form using a sequence of input prompts, which can include text, numbers, geo-locations, multimedia, and barcodes (Fig. 2). A multisegmenting feature developed by APL is used to overcome the SMS 160-character limitation. This feature divides the text message into 160-character segments for SMS transmission and also provides for message reassembly upon receipt by the receiver phone. A second form submission

![Figure 2. Example of the mCollect form-based data entry interface for OE.](image)

![Figure 3. User interface for entering data into OE.](image)
cesses can be used to load surveillance data from any Java Database Connectivity–compatible database. Users can configure the system specifically for the variables included in their database, thereby easing common data ingestion problems, especially the difficulties in trying to get disparate data formats to fit a specific type of data ingestion.

Analysis and Visualization

OE provides a variety of tools to analyze and visualize data and any derived data products. Furthermore, OE provides the ability for multiple users to share information derived from data without having to reveal health data that may be sensitive or private. These tools enhance the process of public health decision making by allowing visualization of disparate data types and analytic results and by facilitating the sharing of public health information across jurisdictional boundaries.

Queries are used to filter the data for analysis by date range, selection from list, multiselection, free form text/number, etc. The query form is shown in Fig. 6. The free-form text input fields support database wildcards. Combinations of logical “AND” and “OR” operators can be used to load surveillance data from any Java Database Connectivity–compatible database. Users can configure the system specifically for the variables included in their database, thereby easing common data ingestion problems, especially the difficulties in trying to get disparate data formats to fit a specific type of data ingestion.

Analysis and Visualization

OE provides a variety of tools to analyze and visualize data and any derived data products. Furthermore, OE provides the ability for multiple users to share information derived from data without having to reveal health data that may be sensitive or private. These tools enhance the process of public health decision making by allowing visualization of disparate data types and analytic results and by facilitating the sharing of public health information across jurisdictional boundaries.

Queries are used to filter the data for analysis by date range, selection from list, multiselection, free form text/number, etc. The query form is shown in Fig. 6. The free-form text input fields support database wildcards. Combinations of logical “AND” and “OR” operators can be used to load surveillance data from any Java Database Connectivity–compatible database. Users can configure the system specifically for the variables included in their database, thereby easing common data ingestion problems, especially the difficulties in trying to get disparate data formats to fit a specific type of data ingestion.
Figure 6. Screenshot of the data report query form for OE. Note that previously created queries can be saved for later use.

be used in queries. Queries are built using a graphical user interface to limit the need for detailed knowledge of structured query languages. This query builder approach focuses all database access logic into one part of the code base, thereby making it easier to manage. These queries may be saved and used to establish case definitions, allowing the user to repeat his or her analysis in a similar fashion and compare results.

Figure 7 is a screenshot showing how data can be analyzed by examining different categorizations of data relationships, such as sex, age group, district, and symptom. These features may allow the user to determine whether the disease outbreak is more prevalent within or among different data groups. Outbreak detection can be improved if the data sources contain information that can be used to include individuals potentially affected by an event and to exclude those unaffected.

Figure 8 shows a screenshot of time series and detection analysis. Time series plots can now be arranged to display different years, one above the other, so that a comparison can be made to see what was happening at the same time in previous years, or whether the seasonal illness patterns are significantly different. Clicking a point on the time series plot shows the user a detailed view of the records. Giving the user the ability to drill deeply into the data allows the user to examine the specific health information that is resulting in an algorithm-derived alert. Data details, which preserve column ordering and sorting, can be exported to CSV files for external analysis using other tools such as Microsoft Excel. Time-series visualization includes anomaly detection and image export. These features assist users in sharing information while investigating a possible outbreak.

OE uses a plug-in application programming interface for detection algorithms that allows a developer to add additional detection algorithms. Currently supported detection algorithms include open-source algorithms such as Generalized Adaptive Smoothing, Cumulative Summation, and Exponentially Weighted Moving Average, in addition to different varieties of the U.S. Centers for Disease Control and Prevention Early Aberration Reporting System algorithms (EARS C1, C2, and C3). Creating a time series without running a detection algorithm is also supported.

A new feature of OE allows the user to create a crosstab report (or N × N × N table) to display the frequency of data on the basis of multiple variables, such as district, age group, and symptom. This is similar to the concept of pivot tables in multidimensional reports. The crosstab report is dynamically generated from the data set, allowing users to select and update its dimensions for different comparisons. In addition, data filters can be applied when creating the crosstab to presample the data. The default crosstab is an n-dimensional matrix that shows the aggregated values of the selected comparison. The report also provides additional visualization features such as heat maps and table bar charts to aid in analysis of the comparison. An example of a crosstab using the heat map visualization is shown in Fig. 9. Crosstab data can also be exported to a CSV file for external analysis.

The output of different data queries and detection algorithms may be used to create maps with specific types of information. This facilitates quick identification of the geographic regions that are impacted and any notable
Figure 8. OE screenshot of time series analysis. Below the plot are details of the detection algorithm output. Detection algorithm alerts are indicated by the red peaks in the plot.

Figure 9. Example of a user-created crosstab employing the heat map visualization feature.
OE provides the user with results in a format similar to that provided by Enterprise ESSENCE, including graphs, charts, detailed data on individual illness reports, and geographic maps of locations of individual illness reports. The OE system can also be used to share actionable information via the Internet among multiple users and across different jurisdictions. OE can be used globally to improve the timeliness of health data collection, enhance the early detection of disease outbreaks, and improve situational awareness.

ACKNOWLEDGMENTS: We acknowledge the collaboration with and financial support of AFHSC-GEIS. We have received invaluable assistance from our colleagues at several U.S. DoD overseas laboratories as well as from our foreign military and civilian colleagues at the many SAGES installation sites. We thank Shradda Patel and Brian Feighner for their assistance with this article. We are deeply indebted to Wayne Loschen, the developer of the Enterprise ESSENCE system, for his sage guidance on these projects.

REFERENCES


The Authors

Timothy C. Campbell was the SAGES technical lead and developer for the OE application. Charles J. Hodanics, Zarna S. Mistry, Gabriel N. Gorelick-Feldman, and Colin J. Taylor designed and developed features for the OE application. Adjoa M. Poku was the technical lead for telephone and SMS data ingestion design and development and was a software developer for OE. Richard A. Wojcik was the Chief Engineer for all versions of ESSENCE. Steven M. Babin was the subject-matter expert in medicine and public health. Jacqueline S. Coberly was the Lead Epidemiologist on the project and obtained user input and usage data on the OE application. Aaron T. Katz was the Project Manager for OE. Sheri H. Lewis is the Program Manager for Global Health Surveillance. For further information on the work reported here, contact Timothy Campbell. His e-mail address is timothy.campbell@jhuapl.edu.
Analytic Biosurveillance Methods for Resource-Limited Settings


Public health surveillance faces many challenges in geographic regions lacking modern technology and infrastructure. This article addresses the role of analytic methods in such regions and evaluates temporal alerting algorithms using both authentic and simulated data sets. Evaluation analyses give the technical background for the statistical methods provided by the Johns Hopkins University Applied Physics Laboratory (APL) Suite for Automated Global Electronic bioSurveillance (SAGES), a collection of modular, open-source software tools to enable electronic surveillance in resource-limited settings. Included in the evaluation are only those statistical methods that are broadly applicable to multiple evolving-background time-series behaviors with limited data history. Multiple detection performance measures are defined, and a practical means of combining them is applied to recommend preferred alerting methods for common scenarios. Effective usage of these methods is discussed in the context of routine health-monitoring operations.

INTRODUCTION

Background

The 21st century has seen advances in many aspects of global disease surveillance. These advances have been driven by heightened concerns over perceived threats to public health both from natural pathogens and from bioterrorism. These concerns have led to mandated improvements at the international level, through revision of International Health Regulations of the World Health Organization, and also in the United States at the national level.

Particular concern exists for surveillance in resource-limited settings (i.e., areas with limited access to medical care; inadequate or no laboratory diagnostic capability; insufficient numbers of first responders, care providers, and public health workers; and sometimes deficiencies in fundamental hygienic needs such as clean drinking water). As a result of these issues, such regions are vulnerable to outbreaks of diseases, such as cholera and typhoid fever, that are not seen in advantaged settings.
SAGES Program at APL: Mission, History, Status

The Johns Hopkins University Applied Physics Laboratory (APL) has contributed advances in electronic disease surveillance since the late 1990s, before the surge of development stimulated by the terrorist attacks of 2001. SAGES (Suite for Automated Global Electronic bioSurveillance) is a collaboration between APL and the US Armed Forces Health Surveillance Center to extend these advances to resource-limited settings. SAGES is an open-source software tool set for data collection, analysis, visualization, and reporting. These tools were designed to maintain and extend established user-driven features of ESSENCE (Electronic Surveillance System for the Early Notification of Community-based Epidemics). The tools were designed to meet a range of institutional needs and capabilities and for convenient integration with local health-monitoring tools. The purpose of this article is to improve the SAGES evaluation/visualization tools by identifying and tuning statistical alerting methods for given contexts. For example, given the amount of historical data available, whether monitoring is done daily or weekly, and whether data are sparse or rich with cyclic patterns, which alerting methods should be used?

Alerting Methods: Principles and Current Objective

The value of statistical alerting methods in a syndromic surveillance system is for detection of statistical anomalies, not detection of actual health events whose confirmation requires definitive evidence that is not immediately available to most SAGES users. In such a system, an alert is signaled when the output of an algorithm for monitoring a data stream crosses a threshold indicating behavior that is statistically aberrant, or too far from expected values to be plausible from random variation alone. The anomaly alerts, especially in combination with other evidence, are useful to prompt investigation of true health events, but the alerts have other causes, including batched data reports, changes in data participation, and changes in diagnosis coding. This article provides the basis for the alerting methods and chosen parameters supplied through the SAGES website as of early 2014, with guidance for effective usage. This guidance does not require sophisticated or time-consuming analysis from SAGES users, who range from part-time technicians to medically trained epidemiologists with varying backgrounds and levels of availability.

METHODS

Selection of Candidate Alerting Methods

The data available to current and near-term SAGES users restricted this project and the initial SAGES open-source alerting methods to algorithms for a single time series derived by aggregation of select clinical data on individuals. For example, a typical input time series is the succession of daily or weekly counts of medical encounter records whose chief complaint field contains words related to febrile illness, such as “temperature,” “fever,” and “feverish.” Candidate alerting methods were algorithms that could flag relevant target signals in the data at manageable background alarm rates. The term background alarm is used here in place of false alarm because false positives are difficult to verify in authentic surveillance data; indeed, given the practical constraints of public health response, many alerts are not investigated at all. The section called Target Signals for Aberration Detection explains how true positives were determined in the two phases of this work.

It is assumed that SAGES users have access only to selected clinical report counts, not to exogenous clinical variables, nor to nonclinical information such as environmental data. Candidate methods may not assume more than a few months of data history, because even in situations when quality data are available for multiple successive years, the older data may not be useful for training or baseline determination because of changes in data provider participation, information systems, or diagnosis coding. Another requirement is that the methods be easily implemented and maintained without assuming future availability of statistical expertise for tuning or model refitting. Implementation and routine usage on health monitoring systems rules out methods that require excessive time to calculate baselines and produce results for input data streams that may be improvised. Transparency is also an essential requirement for SAGES alerting algorithms; the user need not understand the underlying mathematical detail, but the basic concept should be clear enough that the SAGES user can see why an alert is indicated. Many methods noted in recent survey articles do not meet all of these requirements.

For these reasons, the initial set of SAGES alerting methods was restricted to adaptive versions of the control charts long used in the statistical quality assurance community. Adaptive features, noted in the descriptions below, were considered essential for alerting that is robust relative to common data quality issues such as data dropouts and abrupt changes in the background mean for prompt recovery of sensitivity after a large authentic or artifactual spike. Future enhancements will modify the provided alerting methods according to evolving needs and capabilities of SAGES users.

Candidate Alerting Methods with Descriptions

This section briefly describes four chosen alerting methods, denoted Z-score_SAGES, EWMA_SAGES, CuSUM_SAGES, and GS_SAGES. Each is intended to alert when the excess of recent time series values above the baseline expectation is statistically significant, indicating the possibility of an outbreak. These tests ignore
anomalously low values. In each method, the modifications below, detailed in a previous issue of this journal, are implemented where applicable to account for common characteristics and challenges of the surveillance data environment:

- **Evolving data streams**: Candidate methods use a sliding, fixed-length baseline to calculate the mean and standard error, in place of values derived from phase I analysis in industrial control charts. In the latter context, in-control data behavior is typically stable, and data-generating processes can be stopped and adjusted when the data go out of control.

- **Accommodation of sparse or vanishing baselines**: Each method uses a minimum baseline standard deviation to avoid excessive statistic values and to enable the use of these methods for sparse data streams.

- **Non-Gaussian distribution of time-series data**: A correction to computed p-values is applied to account for the fact that count distributions are typically Poisson rather than Gaussian.

- **Robustness to data dropouts**: The methods test for historically implausible strings of zeros and reset to avoid prolonged, excessive alerting when data reporting resumes.

Based on the issues described above and on the practical requirement that methods must produce sensible alerting for time series formed from ad hoc queries without noticeable response delays, the following methods were chosen for comparison.

### Z-score_SAGES

This method implements a standard control chart, an X-bar chart that is a generalization of the EARS C2 algorithm, globally the most widely used alerting method for biosurveillance. Like the C2 method, it uses a sliding baseline with a fixed buffer between the test period and baseline. The test statistic at time step $t$ is then the Z-score $z_t$:

$$z_t = (x_t - \overline{X}_t) / \hat{\sigma}_t,$$  

where $x_t$ is the current time series element, $\overline{X}_t$ is the mean of the time series over the current baseline, and $\hat{\sigma}_t$ is the baseline standard deviation. Numerous global implementations of C2 use a 7-day baseline, 2-day guard band, and a fixed alerting threshold of 3 standard deviations above the mean. ESSENCE and other systems have expanded the baseline to 28 days to achieve more stable alerting behavior. In the current study, the baseline, guard band, and threshold are varied for optimal detection performance. A $p$-value threshold is derived as a lookup of the Z-score value using the Student’s $t$-distribution with degrees of freedom equal to the baseline length – 1.

### EWMA_SAGES

The exponential weighted moving average (EWMA) method evaluated for SAGES replaces the current observed value $x_t$ in Eq. 1 with the recursively weighted average,

$$E_t = \omega x_t + (1 - \omega)E_{t-1},$$  

for a fixed smoothing constant $\omega$, $0 < \omega < 1$, that expresses how the weight of tested observations is distributed backward in time. For a value of $\omega$ near 1, only the most recent values influence this average, whereas reducing the value of $\omega$ increases the influence of older values.

Statistical corrections for the weighted averaging are applied to the standard error and threshold calculations. This modification adds sensitivity to the gradual signals that form the data signature of many outbreaks of interest, but it reduces sensitivity to single spikes. For the current evaluation, combinations of the baseline value, guard band, smoothing constant, and alerting threshold were tested to seek the best detection performance.

### CuSUM_SAGES

Like the EWMA method, the CuSUM chart has also been shown to be timelier than the X-bar chart at detecting small mean shifts and gradual signals. Applications for biosurveillance have focused on aberrations above the baseline expectation. The CuSUM_SAGES implementation uses a sliding baseline as in the Z-score_SAGES method and recursively calculates an upper sum $S_{H,t}$ of scaled differences $z_t$ of $x_t$ above the baseline mean estimate $\overline{X}_t$:

$$S_{H,t} = \max(0, z_t - k + S_{H,t-1}).$$  

In this expression, only differences of the observation $x_t$ above ($\overline{X}_t$ plus $k$ standard deviations) are added to the running test sum, while smaller differences are ignored. Equation 3 assures that the upper sum is nonnegative, and the method alerts if $S_{H,t}$ exceeds a computed threshold. The initial value $S_{H,0}$ of this statistic is set at half the alerting threshold to enable prompt alerting as in the Fast-Impulse-Response CuSUM, and it is reset to this value after an alert to avoid persistent, unwanted alerts because of an extremely high (possibly erroneous) value, while still maintaining sensitivity.

Although many authors have found the detection performance of the CuSUM similar to that of the EWMA, the methods are substantially different. A CuSUM chart does not use the strict time-based weighting of past observations but is influenced only by those observations with scaled baseline exceedance above a fixed level. For
this reason, the CuSUM threshold is usually determined empirically, and the CuSUM_SAGES version derives p-values from lookup tables calculated by running the algorithm on simulated time series of length 100,000.

**GS_SAGES**

This method was adapted for SAGES user groups wishing to monitor daily count data, such as counts of selected clinic reports. In many SAGES settings, only count data are available, with no catchment or population-at-risk data to allow estimation of incidence rates. The main utility of this method is for monitoring daily time series with systematic day-of-week effects, including any regular weekly pattern, which the method can infer and progressively modify. For monitoring with controlled bias, statistical alerting algorithms must adjust to such known systematic data behaviors. The GS_SAGES algorithm is robust to common situations such as clinic closings on weekends or only on a particular day of the week and also on known calendar holidays. The algorithm requires only a couple of months of representative startup data. It is useful for series of counts that are not too sparse in the sense that an overall median count of at least three reports per day is preferred. For median counts closer to zero (i.e., no reports on the majority of days), a simpler adaptive method based on the EWMA control chart is recommended (see the section Results Using Simulated Syndromic Data).

The GS_SAGES algorithm applies generalized exponential smoothing for rapid adjustment to short-term trends and weekly patterns. It has been chosen in place of regression modeling as used in other systems because conventional least-squares regression has been shown vulnerable to large errors when short-term trends affect the data. GS_SAGES uses a prediction based on recursive smoothing equations given below, not on regression or any other global model. These equations can adapt to local changes in the mean value of the observed counts, as in a conventional EWMA chart, but the smoothing is generalized so that the prediction can also adapt to changes in the trend and in the weekly pattern.

For the smoothing equations, let $\alpha$, $\beta$, $\gamma$ denote smoothing coefficients for updating terms corresponding to the level, trend, and seasonality, respectively, and let $s$ be the length of the season in the data. If $y_t$ is the observed count on day $t$, then terms for level $m_t$, trend $b_t$, and seasonality $c_t$ are updated as with the following equations:

- **Level**: $m_t = \alpha y_t + \gamma c_{t-s} + (1 - \alpha)(m_{t-1} + b_{t-1}), 0 < \alpha < 1, (4)$

- **Trend**: $b_t = \beta (m_t - m_{t-1}) + (1 - \beta)b_{t-1}, 0 < \beta < 1, (5)$

- **Seasonality**: $c_t = \gamma y_t + \gamma c_{t-s} + (1 - \gamma)c_{t-1}, 0 < \gamma < 1, (6)$

with $s = 7$ for weekly pattern adjustment. The $k$-step ahead forecast is then:

$$\hat{y}_{n+k|n} = (m_n + kb_n)(c_{n-s+k}). \quad (7)$$

This last quantity may be used for either smoothing or estimation; i.e., it may be used to replace either the test quantity $x_t$, as in EWMA_SAGES, or the sliding baseline mean in Eq. 1. From trying both options on SAGES data, GS_SAGES replaces $x_t$ with the Eq. 4 forecast. The standard error in Eq. 1 is stratified by day of week when a weekly pattern is present. Implementation of these equations requires a careful choice of smoothing constants $\alpha$, $\beta$, $\gamma$ and of initial values $c_0, c_1, \ldots, c_s, m_0, b_0$.

**Data Sets for Alerting Algorithm Evaluation**

The alerting algorithms were evaluated and compared in two phases. The first phase used authentic reports of dengue/dengue-like illness from the National Epidemiology Center of the Republic of the Philippines (RP). The second phase used simulated, stochastic time-series counts to test algorithm performance on less disease-specific time series with systematic background behavior such as day-of-week effects and temporal correlation.

**Authentic Dengue-Related Report Time Series**

Collaboration with the RP, the source of the authentic data described in this article, has broadly informed SAGES development. Dengue is a viral infection for which there is no approved vaccine. It remains a severe health threat in the RP and is included in that country’s list of officially reportable diseases. Globally, dengue surveillance is important because there are up to 100 million annual infections in tropical climates and because prompt treatment can prevent severe illness.

The patient report data were provided by the National Epidemiology Center of the RP Department of Health. Before 2008, dengue-related illness was reported through the National Epidemic Sentinel Surveillance System (NESSS), which gathered data from multiple hospital surveillance sites. After the 2002 emergence of severe acute respiratory syndrome, surveillance methods changed, and NESSS was replaced by the Philippines Integrated Disease Surveillance and Response system (PIDSIR), which became operational in 2008. The number of reportable diseases was increased, and disease-reporting units were added. Some of the data collection, processing tools, and methods were also changed under this new system, enabling access to more complete information.

The time series used for the algorithm evaluation and comparison were regional daily counts of dengue/dengue-like illness reports from the NESSS or PIDSIR system covering nearly 19 years, from the beginning of 1993 to the end of 2011. Regions used for aggrega-
Alerting timeliness on authentic daily time series. Alerts during the 96 chosen event intervals were considered true positives, and alerts outside these intervals were considered false positives.

Simulated Syndromic Background Data

The second evaluation phase used time series that were richer, in scale and in background behavior, and more syndromic (i.e., less disease specific). This phase was motivated by the syndromic report-count data increasingly available on a daily basis in resource-limited settings. For example, the SAGES system also collects daily illness data using short message service (SMS) cellular telephone technology, but the SMS data available for the current analysis covered only one full season and were not used in the method evaluation. Lacking sufficient SAGES data history for more syndromic data, we created background data and target signals with simulation. The simulated time series were modified random vectors drawn from Poisson distributions, often representative of count data. Based on exploratory analysis of the available syndromic SAGES data, we applied three types of modification to these random vectors.

1. Scale: Time series were computed for daily Poisson mean (equivalently, variance) levels 0.5, 3, 10, and 50. This range of values gave report count scales from the small municipality level to the province level.

2. Autocorrelation coefficient: For each scale level, time series were generated with lag-1 autocorrelation coefficient 0 (for independent daily counts), 0.3, and 0.6 (for extreme next-day dependence). The transformations adding these temporal dependencies preserved the Poisson property.

3. Day-of-week pattern: Three day-of-week patterns were applied for each combination of scale and lag-1 autocorrelation. The first pattern was uniform, with no day-of-week weighting, similar to the PIDSR data series. The second pattern was drawn from past syndromic data, with relative weighting of (0.15, 2.15, 1.55, 0.95, 0.95, 0.95, 0.30) for Sunday through Saturday, and the third pattern was drawn from past syndromic data, with relative weighting of (0.15, 2.15, 1.55, 0.95, 0.95, 0.95, 0.30) for Sunday through Saturday.

Figure 1. Weekly counts of dengue-related clinic hospital reports in Cebu Province from November 1992 through November 2011.

Target Signals for Authentic Dengue Report Count Series

Evaluating the alerting performance of statistical methods requires a sufficient number of target signals in the data. Detailed specification of outbreak dates was unavailable for the 10 PIDSR time series, although dengue epidemics were evident in each time series. These events are associated with the rainy season in RP between June and February, when the mosquito vector for the dengue virus is most prolific.

A total of 96 outbreak intervals were selected from the 10 time series, with at least five intervals from each municipality. The selections were based on experience recognizing aberrant signals in time series from multiple surveillance settings. Although these signal selections are subject to judgment bias, the lack of precise truth data, an obstacle characteristic of biosurveillance research and practice, has led to similar target specification procedures in other method evaluations. Moreover, the procedure in the current effort had nearly 19 years of usable authentic dengue-like case reports from multiple municipalities, clearly visible target events, and a relatively quiet background outside the event intervals.

To avoid the subjective choice of exact beginning and ending dates for an epidemic in the noisier daily data, event dates were chosen from weekly plots and used to evaluate alerting on both daily and weekly data. This decision acknowledged the imprecision of measuring alerting timeliness on authentic daily time series. Alerts during the 96 chosen event intervals were considered true positives, and alerts outside these intervals were considered false positives.
The third pattern, with weighting vector (0, 2.15, 1.55, 0.95, 0.95, 0.95, 0), assumed no counts on weekends, representing report counts from clinics open only weekdays, a scenario of practical interest in many resource-limited areas.

The modifications described above give 36 combinations of scale, lag-1 autocorrelation, and day-of-week effect. For each such combination, 18 stochastic time series of report counts of length 730 days, exactly 2 years, were generated as benchmark data for alerting algorithm evaluation.

Figure 2 gives examples of the simulated daily counts used for algorithm evaluation. Sample simulated series with daily means of 3 and 10 report counts are plotted. The upper half of the figure shows simulated series with no day-of-week effect, whereas series in the lower half are modified with day-of-week proportions extrapolated from authentic report data. As discussed above, 18 such series were generated for each combination of daily mean, autocorrelation coefficient, and day-of-week effect.

**Target Signals for Aberration Detection**

**Target Signals for Simulated Syndromic Count Series**

The strategy for injecting target signals into the simulated background data was to select a realistic signal shape, sample from that shape on selected event dates to obtain daily case count injects attributable to an outbreak, and add those injects to the simulated background counts. With this procedure, the signal start, peak, and end dates are known precisely. The strength of the signal was chosen to give a detectable target that would be a challenge to the candidate alerting algorithms. One such injected signal was thus added to each simulated time series. In testing the algorithms, alerts during inject intervals were considered true positives, and alerts on other days false positives. For false alarm calculation, alerts on non-inject days were considered false positives. The advantage of evaluation using simulated data is that no effects of unknown outbreaks are hidden in the data, so that alerts outside of known signal intervals may be accurately called false alerts rather than background alerts.

**Figure 2.** Sample plots of simulated time series without temporal correlation and with daily means of 3 and 10 report counts. Series in the upper plot assume no day-of-week effect, whereas the lower plot shows series with a day-of-week pattern extrapolated from historical data.
For the signal shape, we chose the lognormal distribution proposed by Sartwell in 195021 and widely used since then as representative of the epidemic curve distribution of many infectious disease.19 The rationale is that the distribution of care-seeking dates plausibly reflects the distribution of symptom-onset dates (i.e., the epidemic curve).

Evaluation Measures

Multiple criteria have been recommended for evaluating surveillance algorithms.22 The results described below apply a combination of these measures including sensitivity, specificity, timeliness, and positive predictive value (PPV). After reviewing published attempts to combine these measures,23 we sought a straightforward approach emphasizing priorities of the resource-limited setting. We first list operational definitions of the performance measures applied to the authentic dengue data with known events and to the simulated syndromic data with injected events:

- **Event Sensitivity**, the ratio of alerted target events to all target events: Of the total number of events known or injected, for how many does the algorithm alert before the peak date of the event?

- **Specificity**, or (1 – the background alert rate): Of the dates that are not during known or injected events, for what percentage does the algorithm correctly fail to alert?

- **Timeliness**: What is the delay in days between the start of a target event signal and the first algorithm alert?

- **Temporal Sensitivity (Coherence)**: On what proportion of the days during target events does the algorithm alert?

- **PPV**: What proportion of all algorithm alerts occur during known or injected events?

Event sensitivity is the value adopted by many authors for surveillance alerting methods and is often weighed against the specificity measure, computed on the basis of event days, not events. The temporal sensitivity or coherence measure is rarely published because during the course of an event lasting more than a week, the data may be anomalous only on a few days around the peak, so values of coherence are typically low, especially in nonspecific data. In practice, however, health monitors often cope with the lack of time and resources by investigating only after several alerts are seen, so the coherence measure was included in this evaluation. The PPV as defined above tells how many alerts are likely to be investigated before an event of interest is found. In practice, this measure is more relevant than specificity, a function of disease prevalence, for evaluating algorithm utility.

From these considerations, we defined an algorithm as an alerting method with a fixed set of parameters and threshold. Recommended algorithms were chosen as follows: Consider only algorithms whose sensitivity, specificity, and coherence meet strict minimum criteria. To accommodate the imprecision in alerting timeliness, drop algorithms whose average alerting timeliness is more than a full day longer than the shortest delay among those remaining. Among the remaining algorithms within 0.005 of the highest PPV value, the one with the shortest average alerting delay was recommended.

RESULTS

Methods tested on the dengue data were Z-score_SAGES, CuSUM_SAGES, and EWMA_SAGES. For the noisier simulated data, typical of less specific syndromic series, the GS_SAGES method was also tested.

Results Using Weekly Dengue-Related Report Data

The testing procedure described in Methods was applied to the weekly aggregated counts of the 10 report time series described above to examine alerting performance on the 96 target events. Each series consisted of 992 weeks of counts, with the first 42 weeks reserved as warm-up intervals for the longest algorithm baselines, so that alerts considered false positives could occur in 950 weeks minus the event intervals.

Tested parameter/threshold sets included 120 parameter/threshold combinations for the Z-score_SAGES method, 720 combinations for CuSUM_SAGES to test values of \( k \), and 840 combinations for EWMA_SAGES, including additional combinations for the latter two methods to vary the \( k \) and \( \omega \) constants. These 1680 algorithm combinations were applied to test detection performance on the 96 target signals. From the resulting algorithm output, candidate methods were restricted to those with specificity > 95%, specificity > 95%, coherence > 66%, and alerting delays less than 1.5 weeks. The remaining combinations are tabulated in Table 1 with sorted PPV values above 0.70. No Z-score combination met the timeliness criterion with high PPV. Detection performance for the tabulated combinations is favorable because the weekly data aggregation yields good coherence and high PPV at the required specificity/specificity levels for the seasonal target events. Many other CuSUM and EWMA combinations yielded slightly lower PPV values. Analysis of the full table indicated the use of CuSUM_SAGES, and the parameter combination in the second row with an alerting threshold of \( p = 0.01 \) can be considered the best PPV/timeliness result.

Results Using Daily Dengue-Related Report Data

The same methods were applied with a similar number of parametric combinations to the 10 daily dengue-report time series without aggregation. Each series contained 6939 days of counts, with the first 300 days...
reserved for algorithms with long baselines, so that false positives could occur in each series on 6639 days minus the event intervals. Because of the increased volatility of daily count data, few algorithm combinations yielded a coherence measure above 50%. From the resulting algorithm output, candidate methods were restricted to those with sensitivity > 95%, specificity > 95%, coherence > 40%, and alerting delays less than 4.5 days. The remaining CuSUM_SAGES and EWMA_SAGES combinations are tabulated in Table 2. Again, no Z-score_SAGES combination met the timeliness criterion with acceptable PPV.

From Table 2, 16 algorithm combinations met the event sensitivity/specificity requirement with PPV values above 0.7.

Coherence values are lower and alerting thresholds higher than in Table 1. The EWMA_SAGES algorithm in the top row achieved a PPV of 0.77 with a mean delay

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Baseline Length (weeks)</th>
<th>Guard Band Length (weeks)</th>
<th>Parameter (CuSUM k, EWMA (\alpha))</th>
<th>Alerting Threshold</th>
<th>Specificity</th>
<th>Event Sensitivity</th>
<th>Coherence</th>
<th>PPV</th>
<th>Delay (weeks)</th>
<th>Delay (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.3</td>
<td>0.01</td>
<td>0.96</td>
<td>0.99</td>
<td>0.73</td>
<td>0.75</td>
<td>1.48</td>
<td>10.39</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.2</td>
<td>0.01</td>
<td>0.96</td>
<td>0.99</td>
<td>0.75</td>
<td>0.75</td>
<td>1.39</td>
<td>9.74</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.4</td>
<td>0.02</td>
<td>0.96</td>
<td>1.00</td>
<td>0.78</td>
<td>0.75</td>
<td>1.42</td>
<td>9.96</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.5</td>
<td>0.03</td>
<td>0.95</td>
<td>1.00</td>
<td>0.79</td>
<td>0.74</td>
<td>1.45</td>
<td>10.18</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.1</td>
<td>0.01</td>
<td>0.95</td>
<td>1.00</td>
<td>0.77</td>
<td>0.73</td>
<td>1.35</td>
<td>9.45</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>1</td>
<td>0.4</td>
<td>0.02</td>
<td>0.96</td>
<td>1.00</td>
<td>0.74</td>
<td>0.73</td>
<td>1.46</td>
<td>10.25</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.07</td>
<td>0.01</td>
<td>0.95</td>
<td>1.00</td>
<td>0.77</td>
<td>0.73</td>
<td>1.37</td>
<td>9.60</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>12</td>
<td>1</td>
<td>0.5</td>
<td>0.03</td>
<td>0.95</td>
<td>1.00</td>
<td>0.76</td>
<td>0.72</td>
<td>1.48</td>
<td>10.39</td>
</tr>
<tr>
<td>EWMA_SAGES</td>
<td>12</td>
<td>2</td>
<td>0.7</td>
<td>0.03</td>
<td>0.96</td>
<td>1.00</td>
<td>0.71</td>
<td>0.71</td>
<td>1.49</td>
<td>10.46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Baseline Length (days)</th>
<th>Guard Band Length (days)</th>
<th>Parameter (CuSUM k, EWMA (\alpha))</th>
<th>Alerting Threshold</th>
<th>Specificity</th>
<th>Event Sensitivity</th>
<th>Coherence</th>
<th>PPV</th>
<th>Delay (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWMA_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.5</td>
<td>0.05</td>
<td>0.97</td>
<td>0.98</td>
<td>0.45</td>
<td>0.77</td>
<td>3.95</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.3</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.49</td>
<td>0.77</td>
<td>4.05</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.2</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.56</td>
<td>0.76</td>
<td>4.11</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.4</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.46</td>
<td>0.76</td>
<td>3.69</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>2</td>
<td>0.3</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.46</td>
<td>0.76</td>
<td>3.96</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>2</td>
<td>0.2</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.53</td>
<td>0.76</td>
<td>3.95</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.5</td>
<td>0.05</td>
<td>0.97</td>
<td>1.00</td>
<td>0.42</td>
<td>0.73</td>
<td>3.72</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>2</td>
<td>0.4</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.43</td>
<td>0.73</td>
<td>3.70</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.1</td>
<td>0.05</td>
<td>0.96</td>
<td>0.97</td>
<td>0.66</td>
<td>0.73</td>
<td>4.27</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>112</td>
<td>2</td>
<td>0.1</td>
<td>0.05</td>
<td>0.96</td>
<td>0.97</td>
<td>0.63</td>
<td>0.73</td>
<td>4.30</td>
</tr>
<tr>
<td>EWMA_SAGES</td>
<td>84</td>
<td>7</td>
<td>0.5</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.41</td>
<td>0.73</td>
<td>4.21</td>
</tr>
<tr>
<td>EWMA_SAGES</td>
<td>112</td>
<td>7</td>
<td>0.3</td>
<td>0.05</td>
<td>0.96</td>
<td>1.00</td>
<td>0.52</td>
<td>0.72</td>
<td>3.38</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>84</td>
<td>7</td>
<td>0.3</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.44</td>
<td>0.71</td>
<td>4.10</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>84</td>
<td>7</td>
<td>0.4</td>
<td>0.05</td>
<td>0.97</td>
<td>0.99</td>
<td>0.41</td>
<td>0.70</td>
<td>4.01</td>
</tr>
<tr>
<td>EWMA_SAGES</td>
<td>112</td>
<td>2</td>
<td>0.3</td>
<td>0.05</td>
<td>0.96</td>
<td>1.00</td>
<td>0.48</td>
<td>0.70</td>
<td>3.53</td>
</tr>
<tr>
<td>CuSUM_SAGES</td>
<td>84</td>
<td>7</td>
<td>0.2</td>
<td>0.05</td>
<td>0.96</td>
<td>0.99</td>
<td>0.51</td>
<td>0.70</td>
<td>4.18</td>
</tr>
</tbody>
</table>
Results Using Simulated Syndromic Data

Relative Method Performance

Compared with alerting performance on the PIDSR dengue report data, the algorithms were less effective on the simulated syndromic time series. Coherence and positive predictive value dropped consistently because of the higher background noise level of Fig. 1, day-of-week effects for some of the series, and the transient, stochastic nature of the target signals.

For each combination of simulation parameters, the alerting method/parameter combination was the one with the highest PPV given specificity and event sensitivity $\geq 95\%$, coherence $> 20\%$, and alerting timeliness within 1 day of other qualifying combinations. The top methods by these criteria are listed in Table 3 with the corresponding coherence, PPV, and alerting delay. The GS_SAGES method was the choice for series with a day-of-week effect whenever the count mean was at least 3, likely because of the GS_SAGES adaptation to cyclic patterns. For the sparse time series with mean value 0.5, a CuSUM or EWMA was the best choice. As in the results from the less noisy dengue data with clearer seasonal target events, none of the Z-score_SAGES algorithm combinations met the combined criteria.

Overall Alerting Quality

The effects of the data scale and of autocorrelation on PPV are summarized in the bar charts in Fig. 3 for series with and without day-of-week effects. The improvement of PPV with the scale of the data is consistent. For sparse data series, PPV is near or below 0.1, indicating that only one of 10 alerts results from an injected event. For means of 10 or 50, the PPV exceeds 0.3 unless autocorrelation is excessive.

Algorithm Recommendations

This section summarizes method recommendations depending on the amount of historical data available, the data scale represented by the series mean, and the day-of-week effect. The chart in Table 4 suggests the method of choice given the available data. The baseline columns represent 2-, 4-, 8-, and 16-week baseline lengths. If 112 days of representative data are available, the methods in the rightmost column are recommended.
For PPV averaged over all algorithm/parameter combinations, the overall dependence of PPV on the baseline length and time series scale is summarized in Fig. 4. Except for sparse data series, a baseline of at least 28 days is recommended in view of the jump in PPV from 14 to 28 days. Additional PPV increases may be realized when longer data history of up to 56 days is reliably available. Previous experience and other studies\textsuperscript{10} have suggested diminishing returns for baselines longer than 8 weeks except for regression methods developed for specific time series with mean values above 10 encounters per day.

CONCLUSIONS

The above-described algorithm evaluation analyses on authentic and simulated data support the use of alerting methods on surveillance data from resource-limited settings. The analyses give the background and usage guidance for the open-source methods provided with SAGES.\textsuperscript{7} In view of the wide range of backgrounds and experience of SAGES users, only basic knowledge of the local data characteristics—e.g., outcomes of interest and the scale, seasonal/cyclic behavior, and quality of indicator time series—is required to use this guidance.

From the analysis on the dengue report data, if input series counts are aggregated from carefully chosen patient records, such as the Cebu dengue-related patient reports, then adaptive statistical methods can yield timely alerts with high sensitivity and specificity and practical PPV. The weekly analysis gave better overall PPV and coherence statistics at the cost of alerting timeliness. From the best timeliness results using the weekly data, alerts are not expected until the second week of an event. By contrast, analyses using daily data consistently showed that alerts could be expected after 4–5 days. However, this timeliness advantage has relevance only for PPV averaged over all algorithm/parameter combinations.

### Table 4. Chart of recommended alerting methods for daily report count data as a function of baseline length, daily mean count, and presence of day-of-week effect

<table>
<thead>
<tr>
<th>Recommended Alerting Methods</th>
<th>Daily Mean Visits</th>
<th>Baseline length (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>No Day-of-Week Effect</td>
<td>\begin{tabular}{c} 0.5 \text{ CuSUM, EWMA} \ 3 \text{ CuSUM, EWMA} \ 10 \text{ CuSUM} \ 50 \text{ CuSUM} \end{tabular}</td>
<td>\begin{tabular}{c} \text{ EWMA} \ \text{ CuSUM, EWMA} \ \text{ CuSUM, EWMA} \ \text{ CuSUM, EWMA} \end{tabular}</td>
</tr>
<tr>
<td>Customary Day-of-Week Effect</td>
<td>\begin{tabular}{c} 0.5 \text{ CuSUM} \ 3 \text{ CuSUM, EWMA} \ 10 \text{ CuSUM, EWMA} \ 50 \text{ GS_SAGES} \end{tabular}</td>
<td>\begin{tabular}{c} \text{ EWMA} \ \text{ CuSUM, EWMA} \ \text{ GS_SAGES} \ \text{ GS_SAGES} \end{tabular}</td>
</tr>
</tbody>
</table>
if the public health system can initiate response measures soon enough to exploit it. There will always be the question of whether an early alert is a true signal, and the reduced coherence of algorithms applied to the more volatile daily data will require confidence in the analytic methods and probably the need for additional corroboration. Thus, operational considerations and investigation protocols should be considered along with statistical alerting capability in designing a surveillance system. The analyses on the simulated syndromic data show that alerting performance can vary widely with the input data, emphasizing the importance of careful selection of data indicators or syndrome groups and of monitoring only as many as the investigation and response resources can manage, as noted by other authors. For many nonspecific syndrome groups, the PPV of even a well-chosen method may be much lower than 0.2–0.3, and correspondingly many alerts will need to be investigated before an event of interest is detected. Especially in resource-limited surveillance, this requirement must be understood, or the alerts will be ignored.

Regarding the methods themselves, the analyses show that effective monitoring requires methods appropriate for and adaptive to the input data. Results with the Z-score_SAGES method show that the standard Shewhart-type control chart is not suitable for surveillance data streams, which typically violate the underlying data assumptions. Adaptive versions of this method would be more competitive for detection of single spikes, whereas the other methods are better suited to detection of signals spread over multiple days. Depending on the record filtering criteria, input data streams may display distinct day-of-week patterns. Only the GS_SAGES method yielded effective performance measures while controlling for these patterns. In the absence of these patterns, certain parameter combinations of both EWMA and CuSUM methods were optimal. None of the methods gave high PPV for alerting of moderate-sized events in sparse data streams.

There are limitations to evaluations using both authentic and the simulated data streams. Authentic historical background data pose the problem that false alerts cannot be verified, so calculated PPV may be underestimated. Authentic target signals, the footprints of health events in data streams, are rarely available, and the beginning and ending dates of these signals must be estimated, as in the dengue report data described above. This uncertainty compromises the alerting timeliness and coherence measures. The Cebu dengue report was chosen to lessen these problems. Simulated data streams and target signals avoid these problems, but evaluations using simulated data have the burden of proving that the results are applicable to authentic data. The simulated time series described above were designed to capture statistical properties of observed patient record counts. The combination of authentic and simulated data testing was intended to supply evidence from both perspectives.

Future efforts will seek to further customize statistical alerting methods for surveillance data streams, but the direction of such improvements must keep pace with developing needs in response to global epidemiological needs and with advancing data technology.

**Acknowledgments:** The authors acknowledge the support and advice of Erhan Guven and rest of the APL_SAGES technical team under Program Manager Sheri Lewis. The opinions or assertions in this article are the private views of the authors and do not necessarily reflect the official policy or position of the U.S. Department of the Army, the U.S. Department of Defense, or the U.S. government.

**References**

The Authors

Howard S. Burkom is an APL Principal Professional Staff member who leads and designs algorithm development initiatives for the ESSENCE and SAGES systems. Yevgeniy A. Elbert is an APL statistician who contributes to all phases of data analysis and method implementation and evaluation. Jacqueline S. Coberly is an APL epidemiologist who applies her academic and field experience to ensure relevance and practicality of technical development, to participate in evaluation, and to advise/assist with documentation. Three medical epidemiologists from the Republic of Philippines (RP) were essential to this study, not only for arranging access to critical data sets but also for providing perspective on the challenges of doing surveillance in the RP and for description of national surveillance programs. They are John Mark Velasco of the Armed Forces Research Institute of Medical Sciences and Enrique A. Tayag and Vito G. Roque Jr. of the National Epidemiology Center at the Department of Health. For further information on the work reported here, contact Howard Burkom. His e-mail address is howard.burkom@jhuapl.edu.
Development of Mobile Health Capabilities for Remote Data Collection in Resource-Limited Settings

Adjoa M. Poku and Aaron T. Katz

Since 2007, experts with the Johns Hopkins University Applied Physics Laboratory (APL) and the SAGES (Suite for Automated Global Electronic bioSurveillance) program have been developing mobile technologies focused on improving electronic disease surveillance. Key findings from a 2007 assessment of a short message service (SMS)–based data collection system in the Philippines helped shape development efforts for future SAGES mobile systems. Subsequent industry and trade studies helped the team to identify suitable communications channels and essential requirements for a data collection system targeting resource-limited settings. The resulting open-source framework provides form-based data entry on Android devices, encrypted SMS data submissions, and automated data processing at a remote reporting site. The SAGES development team at APL continues to improve components of mobile systems in response to changes in the mobile technology ecosystem and the needs of the SAGES implementing partners.

INTRODUCTION

Disease surveillance systems are important tools for characterizing patterns of disease occurrence and detecting anomalous disease activity, aiding in disease prevention and control efforts.¹ The effectiveness of a disease surveillance system is limited by many factors, including the timeliness of reporting and the quality of the data provided.¹ Technological advances in the area of electronic disease surveillance have been shown to have a positive impact on overall reporting timeliness and data quality relative to paper-based reporting,² and it is in this context that recent advances in mobile technologies have the potential to advance the field further.

Since 2007, experts with the Johns Hopkins University Applied Physics Laboratory (APL) and the SAGES (Suite for Automated Global Electronic bioSurveillance) program have been engaged in the development of mobile technologies focused on improving electronic disease surveillance. This article provides a brief history of these efforts and an overview of the current state of the mobile technology efforts for SAGES.
SAGES OVERVIEW

SAGES was developed by APL in collaboration with the Global Emerging Infections Surveillance and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS), to provide an end-to-end electronic disease surveillance capability in resource-limited settings. SAGES tools can be configured to customize a surveillance system that is designed around the needs and constraints of a particular installation and partnering organization. SAGES is composed of three primary categories of tools: data collection; data warehousing; and data analysis and visualization (Fig. 1). The overall objective of the mobile components of SAGES is to provide options for data collection in surveillance systems that are well matched to available communication and computing infrastructure at the data collection sites.

THE mHEALTH LANDSCAPE

The growing field of mobile health systems, termed mHealth, is recognized as a distinct subfield of electronic health, or eHealth. No universally accepted definition of mHealth exists, but for the purposes of this article we will use the definition provided by the World Health Organization Global Observatory for eHealth: mHealth is “medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices.”

The communication infrastructure needed to support mHealth systems is growing significantly. The International Telecommunications Union estimated that by the end of 2013, the global mobile-cellular penetration rate, based on individuals, was 96%, with an 89% penetration rate in the subset of developing countries. By the end of 2012 the global mobile broadband penetration rate stood at 22.1%, with a prediction of 30% by the end of 2013.

The adoption of capable mobile platforms is growing in concert with this increasing mobile penetration. In 2013, sales of smartphones exceeded sales of feature phones, accounting for 53.6% of sales globally. In August 2013 the bestselling smartphone internationally, the Samsung Galaxy S4 International Version, sported a GPS sensor, a 1.9-GHz quad-core processor, and 2 GB of RAM. Although it is not possible to make a direct comparison, these specifications are roughly comparable to those provided by the 2007 2.0-GHz MacBook.

Given the convergence of increasingly capable mobile platforms for health applications and increasing communication channel availability, the stage has been set for increased mHealth system realization. To this end, mHealth systems have seen a significant adoption globally, with the World Health Organization reporting that as of 2011, 87% of high-income countries and 77% of low-income countries responding to their global mHealth survey are participating in at least one mHealth initiative.

It is in this context of growing capacity for mHealth systems and the promise demonstrated by eHealth to support disease surveillance that the SAGES program began developing mHealth solutions focused on disease surveillance.

Figure 1. SAGES high-level architecture.
SAGES mHEALTH TIME LINE

The SAGES program’s involvement in mHealth dates back to 2007 with a collaborative effort between APL and Philippines-AFRIMS Virology Research Unit (PAVRU). PAVRU and the Republic of the Philippines Health Office, Cebu City (CHO) staff jointly developed a short message service (SMS)–based data reporting system to augment the existing paper-based reporting capabilities at clinics in Cebu City. APL was asked to perform a technical and public health assessment of the PAVRU system. Several key findings emerged from this effort that would help shape future SAGES mHealth development efforts, most notably:

- The sustainability of a mHealth system is constrained by the ability to develop and maintain the system using locally sourced materials and networks.
- The utility of a mHealth system is directly proportional to the validity of the data produced, and as such, robust data validation capabilities should be integrated and supported at as many levels as possible.
- The viability of a system is based not only on technical factors but also on the degree to which public health professionals, decision makers, funding organizations, and health workers embrace the concept.

The joint PAVRU–CHO system ultimately demonstrated the utility of integrating a close to real-time SMS channel of individual clinic data into a disease surveillance system to realize improvements in the timeliness of data reporting relative to paper-based systems.11

In 2009, APL undertook a study of global electronic information communication technologies, characterizing these technologies with respect to accessibility, cost, hardware requirements, and global availability. The results revealed a viable solution for mHealth in resource-limited settings.12 The finding from this study, which is still supported by recent data,5 is that although broadband communication methods are desirable for their speed and bandwidth, they do not have sufficient penetration in the developing world to serve as an underpinning for a mHealth system. In contrast, mobile-cellular communications have broad penetration (approximately 90%) in the developing world;5 thus, leveraging the SMS protocol for data gathering in a mHealth system in the developing world appears to be the best path forward.

Mobile hardware evolves in tandem with the continual improvements and expansions of communications networks and protocols. Additionally, the availability of devices in different global markets often varies in terms of features and costs. In 2007, when SAGES initially entered the mHealth space, feature phones dominated the global markets. These handsets supported the Java 2 Micro Edition framework and offered a limited set of interactivity and a relatively small screen. By 2009, the technological landscape for mHealth system hardware had fundamentally shifted. With the release of the first generation of Apple iPhones in 2007 and the further proliferation of small-form-factor devices supporting Java 2 Micro Edition, there was a clear shift in the industry toward more interactive capabilities directly on the devices. Unfortunately, although these systems were highly capable, the proprietary nature of their application programming interfaces as well as the relatively high cost of the devices limited the utility of these systems in resource-limited settings.

With Google’s open-source release of the Android Software Development Kit and the reference HTC Corporation’s G1 hardware in 2008, the burgeoning smartphone space began to see the introduction of lower-cost but highly capable platforms with a myriad of sensors and touch screen interfaces. It is in this context that SAGES forged ahead to develop a more fully featured mHealth data collection capability that leveraged Android-equipped devices. Leveraging previous results from the APL–PAVRU effort, the system was architected with an aim to provide data validation at the point of collection, to provide data transmission via SMS, and to target lower-cost hardware sourced in the markets of foreign partners who were working with AFHSC-GEIS to develop an electronic disease surveillance capacity.

With this overarching mission in place, SAGES developers conducted a trade study to compare the suitability of developing a framework internally or leveraging existing frameworks. It was quickly determined that numerous suitable customizable frameworks existed in the marketplace, and leveraging an existing framework would be the most economical and expedient path forward.

Eight data entry frameworks were evaluated for the SAGES mobile platform (Table 1). The data entry frameworks were evaluated against seven separate criteria:

- Support for form-based data entry via a graphical user interface
- Support for transmission of data via SMS
- Support for Android-based devices
- Support for creation of forms with multiple data entry field types
- Open-source licensing for the framework
- Evidence that the community around the project is actively providing support and that the project is accepted as being beyond its beta release (i.e., a mature product)
- Free of licensing, maintenance, and service costs
Nine data receiver frameworks were evaluated against seven criteria (Table 2):

- Support for receipt of data via SMS
- Local storage of transmitted data
- Compatible with the Microsoft Windows operating system
- Open-source licensing for the framework
- Setup process supported by an installer
- Evidence that the community around the project is actively providing support and that the project is accepted as being beyond its beta release (i.e., a mature product)
- Free of licensing, maintenance, or service costs

Table 2. Selection criteria for SAGES's server-side tool

<table>
<thead>
<tr>
<th>Server-side tool</th>
<th>SMS receipt layer</th>
<th>Data stored locally</th>
<th>Windows compatible</th>
<th>Open source</th>
<th>Installer-based setup</th>
<th>Mature product</th>
<th>Limited potential for operational costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODK Aggregate</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FrontlineSMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>EpiSurveyor</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GATHERdata</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RapidSMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RapidAndroid</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mesh4X*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eMOCHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blue shading denotes selected candidate tool; taupe shading denotes candidate tool.

* Could not gain access to adequate information to validate any claims.

Table 1. Selection criteria for SAGES’s client-side tool

<table>
<thead>
<tr>
<th>Client-side tool</th>
<th>Form-based graphical user interface</th>
<th>SMS submission layer</th>
<th>Android support</th>
<th>Multiple field types</th>
<th>Open source</th>
<th>Mature product</th>
<th>Limited potential for operational costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODK Collect</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FrontlineSMS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>JavaROSA</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>EpiSurveyor</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GATHERdata</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mesh4X*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eMOCHA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moca</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blue shading denotes selected candidate tool; taupe shading denotes candidate tool.

* Could not gain access to adequate information to validate any claims.
Based on these criteria, three receiver frameworks were selected as candidates: RapidSMS, RapidAndroid, and FrontlineSMS. All three platforms were highly capable, but RapidSMS’s lack of support for the Windows operating system and FrontlineSMS’s proprietary licensing eliminated these frameworks. RapidAndroid met all seven criteria and was selected as the development base for the data receipt capability for SAGES.

SAGES mHEALTH ARCHITECTURE

The SAGES mHealth architecture has been designed around a model of supporting full end-to-end cellular communications for data collection. It comprises three major components: a mobile data collection platform; a mobile data receiving platform; and an extract, transform, and load (ETL) process (Fig. 2).

The data collection platform, SAGES mCollect, is an Android application that has been developed on top of the open-source ODK Collect framework. It provides customized form-based data collection and thus can support the collection of both individual patient data and aggregate condition counts (e.g., the number of influenza cases seen at a clinic during a given week). The ODK brand offers tools that can be used to develop custom forms tailored to the needs of different reporting sites, and these forms can be shared among all of the users of SAGES mCollect as part of a SAGES installation. SAGES mCollect includes several mechanisms for data validation and quality assurance, which satisfies the requirement of performing data validation at the point of data entry.

Because the ODK Collect framework does not inherently support all of the features necessary for a secure cellular (minimum 2G)-only data infrastructure, several additional capabilities were developed in realizing SAGES mCollect. Extensive work was done to create a multipart SMS protocol, which allows for sending data in quantities that would otherwise be too large to package into a single SMS message (typically limited to 140 characters). Because of privacy concerns related to open transmittal of sensitive health data, a capability for sending and receiving encrypted and multipart SMS messages was provided. The system currently supports Advanced Encryption Standard 128-bit symmetric key encryption (the 128-bit key size meets the National Institute of Standards and Technology’s minimum approved value for providing security strength using Advanced Encryption Standard cryptographic keys). All of these additions are transparent to the end user of the SAGES mCollect app.

The data receiver application, SAGES mReceive, has been designed to support installations where a minimum 2G cellular data connection is available. For this use case, it has been implemented as an app on top of the open-source RapidAndroid framework. mReceive runs on an Android phone that is tethered via universal serial bus to the OpenESSENCE workstation. (OpenESSENCE is the flagship SAGES web system that provides data warehousing, analysis, and visualization capabilities.) This framework was extended in a fashion similar to that used for the mCollect application, adding in the multipart SMS and data encryption capabilities. In addition, an automation suite was built into the app, enabling mReceive to repackaging and export data received in a form suitable for consumption by the ETL process.

The ETL process has been implemented as a set of lightweight scripts, a Java application, and database-stored procedures that execute on a scheduled basis on the OpenESSENCE workstation. The mReceive application writes its data to a directory on the receiver phone, which is monitored by the ETL process. If data are available in this directory, the ETL process pulls the data from the phone, transforms them to the match the OpenESSENCE database schema, and pushes them into the OpenESSENCE database. A set of stored procedures in the OpenESSENCE data-
base is triggered upon this load, serving as a final validation layer before the data enter the database.

CONCLUSION

The SAGES mHealth architecture has been an integral component in most of the SAGES deployments to date. By supporting structured electronic data collection in remote point-of-care settings, it has the potential to improve data timeliness and validity, enabling more effective disease surveillance. It has been architected as a fully open-source system, on top of existing open-source frameworks, decreasing its cost burden and increasing its potential for modification and reuse. At the time of publication of this article, APL staff has partnered with public health professionals in sub-Saharan Africa to jointly tailor the architecture to support their unique data collection needs, with ownership and maintenance migrating fully to the partners in country.

The mHealth footprint for SAGES is expected to increase and become further refined in the future. SAGES mHealth components will continue to be suitable for support in resource-limited settings, but the leaders of the project plan to take advantage of recent advancements in mobile technologies to more broadly support better-resourced environments.

REFERENCES


The Authors

Adjoa M. Poku is a software engineer in the Bio-Threat Awareness Systems Group in the Asymmetric Operations Sector (AOS). She is currently working on disease surveillance and modeling systems, including SAGES. Aaron T. Katz is the Assistant Group Supervisor and a project manager in the Bio-Threat Awareness Systems Group in AOS. For more information on the work reported here, contact Adjoa Poku. Her e-mail address is adjoa.poku@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
A Tailored Approach to Implementing Open-Source Electronic Disease Surveillance Tools

Shraddha V. Patel

The World Health Organization issued revised International Health Regulations (IHR) in 2005, calling for significantly increased reporting and responding capabilities for all signatory nations and potentially improving the capacity of all countries to detect, assess, communicate, and respond to public health threats. Electronic disease surveillance is an important component of a comprehensive global public health disease prevention and control strategy and can contribute significantly to IHR 2005 compliance. The Johns Hopkins University Applied Physics Laboratory (APL) has created SAGES (Suite for Automated Global Electronic bioSurveillance), a suite of open-source electronic disease surveillance tools for resource-limited environments. Engineers and epidemiologists at APL use a tailored approach when implementing these tools, working with local stakeholders to identify specific needs, constraints, and expectations to increase overall adoption and sustainment of the system. The resulting system has the potential to facilitate local and regional electronic disease surveillance, regional public health collaborations, and international disease reporting. SAGES tools have been implemented in several countries around the world.

INTRODUCTION

Emerging public health threats often originate in countries that have limited public health resources and infrastructure. In recognition of this fact, the World Health Organization issued revised International Health Regulations (IHR) in 2005, calling for significantly increased reporting and responding capabilities for all signatory nations. If implemented fully, the IHR will improve the capacity of countries to detect, assess, communicate, and respond to public health threats. The mission of the Global Emerging Infections Surveillance and Response System, a division of the Armed Forces Health Surveillance Center (AFHSC-GEIS), is to support global surveillance, training, research, and response with regard to emerging infectious disease threats, in support of health protection among U.S. forces, the Military Health System, and the global public health
community. Electronic disease surveillance is an important component of a comprehensive global public health disease prevention and control strategy and contributes significantly to support for IHR 2005 compliance in partner nations. AFHSC-GEIS is committed to enhancing electronic disease surveillance capacity in resource-limited settings around the world. Use of electronic methods for data collection and analysis has the potential to improve the accuracy and timeliness of outbreak detection as well as to provide situational awareness during or in the aftermath of an outbreak or pandemic.

To support AFHSC-GEIS in its goals, the Johns Hopkins University Applied Physics Laboratory (APL) is leveraging its extensive experience in the design and implementation of ESSENCE (Electronic Surveillance System for the Early Notification of Community-based Epidemics) in the DoD, the U.S. Veterans Health Administration, and multiple state health departments to create and implement SAGES (Suite for Automated Global Electronic bioSurveillance). SAGES is an open-source electronic disease surveillance system for use in resource-limited settings around the world. The tools are customized for the organization (for example, a Ministry of Health or Ministry of Defense) that will be using them. In most instances, the introduction of SAGES tools represents a paradigm shift from a paper-based disease surveillance system to an electronic system. To ensure success in this environment, APL epidemiologists and software engineers follow a tailored implementation process to implement SAGES tools in partner nations.

**PLANNING PHASE**

The implementation process begins when AFHSC-GEIS identifies a host nation that is interested in implementing SAGES within a specific organization in their country (typically, Ministry of Defense or Ministry of Health). The organization is the public health authority that will own, use, and maintain the SAGES system. AFHSC-GEIS also identifies an implementing partner who will work closely with the organization and APL during the implementation process. Typically, the implementing partner is U.S. personnel of other AFHSC-GEIS-funded collaborators located in the host nation or region. Historically, many of these collaborators have been part of the AFHSC-GEIS network of overseas laboratories that support AFHSC-GEIS public health initiatives.

The implementing partner plays the critical role of ensuring that all stakeholders are on board and the implementation goes as smoothly as possible. The partner identifies one or more points of contact (POCs) within the host nation with whom to coordinate system design and approval of the final system design. The implementing partner engages with the host organization to ensure that there are no conflicting priorities, because there may be competition for limited public health resources. In this phase, the implementing partner and the host nation POC will establish initial buy-in for the SAGES system within the host government and commit to meet with the APL team in-country.

**DESIGN PHASE**

After obtaining the initial approvals, the APL team begins designing the SAGES system for the partner nation. The APL team consists of one or more software engineers and an epidemiologist. The implementing partner and partner nation POC assist in scheduling a site visit for the APL team, which includes meetings with decision makers, public health officers responsible for disease surveillance, and local information technology (IT) support. The team assesses country-level capabilities and needs for the surveillance system in question. The assessment must consider the requirements and objectives of the partner nation while keeping in mind infrastructure, finance, and personnel constraints within the host organization. The eventual design reflects the best-fitting, low-cost technology for the locale, enabling the system to be locally supported and not sponsor driven.

The answers to these questions inform the design of the proposed SAGES system for the host organization. During the assessment visit, the APL team completes a draft version of the design to share with the implementing partner and the POC to gain agreement on the way forward.

**DESIGN APPROVAL PHASE**

After returning to the United States, the APL team further reviews the findings from the in-country assess-

---

**Table 1. Design phase questions**

<table>
<thead>
<tr>
<th>Focus Area</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public health</td>
<td>What is the purpose or objective of the system?</td>
</tr>
<tr>
<td></td>
<td>Who are the users of the system?</td>
</tr>
<tr>
<td></td>
<td>What data will be collected?</td>
</tr>
<tr>
<td></td>
<td>Where will the data collection happen?</td>
</tr>
<tr>
<td></td>
<td>How will the data be collected?</td>
</tr>
<tr>
<td></td>
<td>What diseases are of most importance?</td>
</tr>
<tr>
<td></td>
<td>How frequently will data be analyzed?</td>
</tr>
<tr>
<td></td>
<td>What types of analyses are needed?</td>
</tr>
<tr>
<td>Technical</td>
<td>What kind of hardware is available?</td>
</tr>
<tr>
<td></td>
<td>What is the nature of the available network infrastructure (Internet, Wi-Fi, cellular)?</td>
</tr>
<tr>
<td></td>
<td>Who will pay for the necessary hardware and services (SMS, Internet, Wi-Fi)?</td>
</tr>
<tr>
<td></td>
<td>Will routine training and support be available?</td>
</tr>
</tbody>
</table>

SMS, Short message service.
Using the information gathered from the assessment phase, APL software engineers customize features in OpenESSENCE such as labels, fields, database structure, data entry screens, customized reports and queries, data types, and the theme (colors and logo). OpenESSENCE supports internationalization and can easily accommodate any language. Field labels and other user interface items are translated into the native language of the partner nation, with help from the implementing partner or POC (Fig. 2 shows an OpenESSENCE system customized with a Spanish user interface). Shape files for GIS mapping are acquired and added to the system. The system is implemented on a server that is accessible by the partner nation so that stakeholders can review the customized system and offer feedback before final implementation in-country. This process is iterative; stakeholders are encouraged to offer feedback during this phase before the APL team arrives in-country for the installation.

The implementing partner and POC are responsible for acquiring all hardware and telecommunication services as described in the system design. They also make sure that appropriate space is available to house the

![Diagram](image_url)

**Figure 1.** An example of a SAGES system design.

![OpenESSENCE](image_url)

**Figure 2.** OpenESSENCE with Spanish user interface.

The team provides the final design to the implementing partner and POC for review and feedback and then modifies the design as needed based on any feedback. The APL team then finalizes the design and distributes it to the implementing partner and POC. In most cases, the initial design reflects a pilot system with a small number of users and data collection sites. The pilot system model allows problems to be identified and resolved quickly without jeopardizing the overall system adoption process.

The POC seeks review and approval of the design from the appropriate authority in the partner nation. Implementation will not move forward until approval has been received. Figure 1 shows an example of a SAGES system design.

**DEVELOPMENT PHASE**

Once approval for the architecture has been received, the development of the customized SAGES system begins. The system includes OpenESSENCE, which is an open-source multiuser network-accessible data entry, analysis, and visualization tool that enables an epidemiologist to monitor population health data from any computer connected to that network. OpenESSENCE supports dynamic query capability, GIS (geographic information system) mapping, and graphical analysis via time series charts, bar charts, and pivot tables. It also supports anomalous event detection for outbreak investigation.
Just as important as the content of the training are the people who attend. The time spent in-country is valuable but short, so it is important to gather as many end users as possible for the training opportunities. Although this is not always feasible, the APL team plans the visit well in advance to maximize participants’ availability. While in-country, the APL team may need to travel to multiple sites to meet with various groups of end users. The team also works closely with an individual identified as the “superuser” for the SAGES system. This person takes on the role of trainer when new end users need training and is the main POC for APL team members after they depart the country. This train-the-trainer approach is effective because it enables the transfer of knowledge without the need for the APL team to be continuously involved. When personnel roles and responsibilities change, the SAGES adoption process continues with the presence of a strong champion within the organization.

The in-country training affords a valuable opportunity to build important collaborative relationships with stakeholders, improving the adoption process and potentially improving outcomes. The APL team listens to the questions and concerns of the end users and adjusts the software or the process as needed. This type of collaboration builds a level of trust that is very important when working with a foreign partner.

**OPERATIONAL PHASE**

The APL team provides post-installation support and troubleshooting as needed to ensure the system is operating as expected. This phase typically lasts throughout the pilot process. Communication occurs via e-mail or scheduled conference calls. The post-installation phase

**Figure 3.** APL team members and members of the Royal Thai Army after training in Thailand.
Implementing open-source electronic disease surveillance tools in resource-limited settings, but using a tailored approach to implementation in which the design team collaborates with local stakeholders to identify specific needs, constraints, and expectations increases the overall adoption and sustainment of the system. The resulting system has great potential to facilitate local and regional electronic disease surveillance, regional public health collaborations, and international disease reporting.

Acknowledgments: The author is grateful to the following SAGES team members for their contributions: Sheri Lewis, Brian Feighner, Jacqueline Coberly, Aaron Katz, Timothy Campbell, Charles Hodanics, Adjoa Poku, Richard Wojcik, and Lieutenant Commander Christopher Perdue (AFHSC).

References

Conclusion
SAGES tools are currently deployed in several countries throughout the world, including Cambodia (Fig. 4), Cameroon, Uganda, and Peru. Many challenges exist when implementing electronic disease surveillance tools in resource-limited settings, but using a tailored approach to implementation in which the design team collaborates with local stakeholders to identify specific needs, constraints, and expectations increases the overall adoption and sustainment of the system. The resulting system has great potential to facilitate local and regional electronic disease surveillance, regional public health collaborations, and international disease reporting.

The Author
Shraddha V. Patel is a Senior Professional Staff member in APL’s Asymmetric Operations Sector. She is the Assistant Project Manager for the SAGES task. She is responsible for training and documentation for the SAGES tools and has participated in SAGES implementations. Her e-mail address is shraddha.patel@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
The use of open-source software (OSS) has dramatically increased in the past several years, particularly in the public health domain. The Johns Hopkins University Applied Physics Laboratory’s (APL) work on developing and licensing OSS identified a need within the public health community to better understand the definition and connotations of the words open source and the various open-source licenses. The use of OSS in the public health domain can dramatically improve the implementation of mobile and electronic health initiatives in resource-limited settings because OSS provides an affordable alternative to costly proprietary software.

INTRODUCTION

The term open source generally refers to software that is made readily available by an individual or group for others to use, modify, or redistribute under a licensing agreement with very few restrictions. Anyone can use the software without having to pay royalties or negotiate a license agreement. Open-source software (OSS) is not a new creation, but it has been used with increasing popularity in large-scale commercial software projects in recent years. It has been called “the software that runs the Internet,” referring to its significant use in the Internet’s infrastructure, including the Apache Web server, the Mozilla browser, and Linux operating system. There are currently at least 50 different open-source licenses, and they represent a unique approach to licensing when compared with licenses normally used in a commercial environment. (See Table 1 for a listing of licenses by name.)

In the past several years, the use of OSS in the public health field has grown dramatically. In particular, the field of mobile health, or mHealth, has seen a substantial increase in the use of OSS due to the ubiquitous nature of cellular telephones. By definition, mHealth refers to “the practice of medicine and public health, supported by mobile devices.” In general, it involves the use of “mobile communication devices, such as mobile phones and PDAs, for health services and information.” Many of these platforms and associated tools, whose primary users are those in the field of public health or clinical care, are purportedly open source. However, in a community that is generally not as savvy in information technology, the term open source is confusing and leaves many unanswered questions. The Johns Hopkins University Applied Physics Laboratory (APL), in conjunction with the Global Emerging Infections Surveillance and Response System,
a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS), has been working to develop and deploy open-source disease surveillance capabilities in resource-limited settings. In the course of this work, APL and AFHSC-GEIS identified a need to better understand the definition, nuances, and connotations of the term open source within the public health community.

The purpose of this article is to provide an introduction to open-source licensing and the main elements to consider when determining whether to use OSS or when selecting an open-source license. In particular, it provides examples of the use of open-source licensing in the development tools used in the public health domain. This article also provides background on copyright and the distinction between a copyright and a license, a discussion of the history of open source and “free software” (terms that are often used interchangeably and also frequently consolidated and referred to as FOSS, or Free and Open-Source Software), and an overview of commonly used licenses with strong user communities. It also discusses several myths related to the benefits and hazards of using open-source licenses and OSS and provides examples of how government agencies are confronting these myths and successfully using OSS to their advantage.

## BACKGROUND ON COPYRIGHT

Copyright is a form of intellectual property law and protects original works of authorship. All software is subject to copyright law, and as soon as source code is created, anyone (other than the author) who wants to use it must obtain explicit permission from the author. As soon as a work is created and “fixed in a tangible form that is perceptible either directly or with the aid of a machine or device,” it is protected by copyright. Copyright is how an author retains control over his or her work. Software copyright is “the exclusive legal right to control the rules for copying, modifying, and distributing a work of software.” The person or organization that has the right to control the work is called the copyright holder. When copyright holders permit others to use, modify, or distribute their software, they have granted a license. The license is the permission to use the software in some way—it can be an unconditional grant of permission that mirrors the rights of the copyright holder or a conditional grant of permission that allows individuals to copy or use the software according to certain provisions. Open-source licenses fall into both categories.

A general commercial copyright license usually protects the copyright holder’s interests by placing restrictions on how the software can be used. For example, a commercial software license usually prohibits the copying or modification of the software, mainly by distributing only the machine-readable binary or object code. OSS licenses give users more rights than a general commercial license because the user gets access to the source code and has the right to change the source code. The term copyleft was generated by the free software community and is the term for a license condition that ensures that all modified versions of the software can be copied, modified, and/or distributed in the same way as the original. By ensuring that downstream users receive source code and permission to modify it, a copyleft license is said to keep code “forever free.” Not all open-source licenses have copyleft provisions, but many do. Despite what the name may imply, copyleft is still a license and is enforced by copyright law. Instead of withholding permission to copy or modify a work (as in the traditional sense of a copyright), copyleft uses copyright law to actually require that those permissions be granted.

It is important to note that using an open-source license is not the same as placing the software in the public domain. The terms of the open source license must be met, and the copyright holder retains rights to the work. If the terms of the open-source license are not met (e.g., the same licensing provisions are not applied to derivative works), the copyright has been infringed and the copyright holder has certain legal remedies available to him or her. No matter how permissive the open-source license, the copyright holder’s interests are protected. If the software is released into the public domain, the author surrenders the copyright. Put another way, as soon as software code is created and saved, a copyright attaches to the work. Placing the software in the public domain relinquishes all rights associated with the work. It is not equivalent to granting a license because the author is not limiting or placing restrictions on the use of the software in any way. In fact, software placed in the public domain can be used, modified, and removed from the public domain by another user asserting copyright ownership.

## FREE VERSUS OPEN SOFTWARE

The terms free software and open software are often used interchangeably, and they are also frequently consolidated and referred to as FOSS. However, although a majority of software is both open and free, there are distinctions to be made between the two categories, both in philosophy and in the licenses that fall within each. The free software movement is, at its roots, about the users’ freedoms, whereas open source focuses on making software better from a practical perspective by allowing access to the code so that others can improve it. These two views may lead to the same outcome in how the software is treated from a copyright perspective, but the goals for getting to that outcome are slightly different. (For a comprehensive overview of free software and OSS, see Ref. 8)

The concept of free software began in 1984 with Richard Stallman, a Massachusetts Institute of Technology (MIT) researcher. Stallman was concerned that computing would be dominated by a few powerful people
if software were all proprietary. He considered software scientific knowledge that should be shared and distributed to further innovation in computer science. In 1984, he left MIT and began the GNU Project and the Free Software Foundation (GNU is a recursive acronym for "GNU’s Not Unix"). One goal of the GNU Project was the development of a freely available operating system that could run GNU software.8

The most important characteristic of free software is the underlying philosophy for why software should be free and what free means. The philosophy of free software is one that respects users' freedoms while benefiting society by promoting sharing and cooperation. The term free software is about freedom, not price. The distinction Stallman makes is that free software is “free as in speech, not as in beer.”8 A program is free if you can run the program for any purpose; you have the freedom to modify the program (requiring access to the source code); you have the freedom to redistribute copies with or without a fee; and you have the freedom to distribute modified versions of the program so the community of software developers can benefit from improvements.8,9 Note that the freedom to sell copies of software is permissible because the term free here does not refer to price, so there is nothing prohibiting someone from generating revenue from free software—the founders of the free software movement believed such revenue could ideally be used to generate new free software projects. However, to thwart businesses from co-opting free software for their exclusive commercial use, Stallman created the GNU General Public License (GPL). The GPL license is discussed in detail later in this article.

Many software companies rejected the concept of free software in part because it seemed to fundamentally conflict with having and furthering a commercial interest in a product. So, in 1997, a group of individuals came together to promote the concept of free software and created the term open source.8 Using the term open source was a way to market the idea of free software by removing the economic context of “free” to make it more palatable to private companies. However, there are practical differences between free software and OSS. While open source captures much of the spirit of GNU, it allows for provisions free software does not, such as the ability to mix proprietary software and OSS. The Open Source Initiative (OSI), an organization that provides oversight of the open-source mission, refers to open source as “a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.”10

The OSI created the Open Source Definition, which has several distribution terms with which OSS must comply.10 According to OSI, software is open source if it meets the following criteria:11

1. **Free Redistribution.** The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. **Source Code.** The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. **Derived Works.** The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. **Integrity of the Author’s Source Code.** The license may restrict source code from being distributed in modified form only if the license allows the distribution of “patch files” with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. **No Discrimination Against Persons or Groups.** The license must not discriminate against any person or group of persons.

6. **No Discrimination Against Fields of Endeavor.** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. **Distribution of License.** The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. **License Must Not Be Specific to a Product.** The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.
9. **License Must Not Restrict Other Software.** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open source software.

10. **License Must Be Technology Neutral.** No provision of the license may be predicated on any individual technology or style of interface.”

**OVERVIEW OF COMMONLY USED LICENCES**

In general, open-source licenses can be broadly categorized into those that apply no restrictions on the distribution of derivative works and those that do apply restrictions to ensure that the code will always remain open/free. (Note that the term *derivative works* is defined by the U.S. Copyright Act and generally refers to a work based on one or more preexisting works. However, the Act does not specifically address derivative works in software, so the law as it applies to OSS is not well established.) The former is also called an “academic license,” and its purpose is to promote a public commons with unlimited use but it contains no requirement to contribute back to the community.

The latter type of license is also referred to as a reciprocal or “share alike” license because it requires that any derivative work retain the original license. Although there are licenses that exist outside of these categories, the licenses discussed below are grouped into one of these two categories, with the exception of the Mozilla license, which is characterized as a hybridization of both. Table 1 summarizes the most commonly used open-source licenses, although additional licenses are included in the discussion below.

**Academic or Nonprotective Licenses**

**The Berkeley Systems Distribution License**

The Berkeley Systems Distribution (BSD) license is one of the least restrictive and most recognized open-source licenses. The license was developed by the University of California at Berkeley and allows free use of the OSS, including the ability to modify the software. The BSD license allows for redistribution and use of source code whether modified or not, as long as the source code retains the copyright notice and other notices regarding disclaimers of warranty and limitations on liability found in the license. The BSD license allows the software to be combined with proprietary software or modified and turned into proprietary software. The BSD allows derivative works to be released under a license other than the BSD; hence there is no copyleft provision. The original BSD had a clause mandating attribution of contributors in advertising of the software. This clause has since been removed, although users of code under the old version of the BSD license must be careful to comply with the advertising clause. A clause still exists prohibiting use of the copyright holder’s name in any promotion of software.

**The MIT License**

The MIT license is very similar to the BSD license and is often referred to as being part of the BSD family of licenses. Like the BSD license, it permits reuse of open-source code within proprietary software as long the MIT licensing terms are included in the proprietary software. The main differences between the MIT license and the BSD license are that the MIT license does not contain a clause prohibiting the use of the copyright holder’s name in promotion of the software and it places more emphasis on the user by emphasizing the right to “use, copy, modify, merge, publish, distribute, sublicense and/or sell copies of the Software.”

**Apache**

The Apache license was created by the Apache Software Foundation. Like the BSD and MIT licenses, the Apache license allows software to be used without any obligation to redistribute the source code of any of the derivative works. The main difference is that the Apache license provides a clause about patent licensing and termination. In addition to providing a patent clause, the Apache license requires that any modifications to the source code distributed under the license carry prominent notices that the files were changed.

**Artistic**

The Artistic license falls into the same category as the BSD, MIT, and Apache licenses in that it allows modified versions of Artistic software to be licensed independently, with some conditions. Version 1.0 was criticized by the Free Software Foundation for being too vague, but version 2.0 is accepted by both the Free Software Foundation and OSI. The Artistic license was the first open-source license deemed enforceable under copyright law as opposed to contract law.

**Mozilla**

The Mozilla Public License (MPL) was originally created by Netscape, and version 1.1 is used by the Mozilla Application Suite, Mozilla Firefox, and other Mozilla software and has been adapted for use by other companies. It combines aspects of the BSD and GPL licenses. It allows for commercial licensing of derivative works, and changes to source code covered under the license must be made freely available. Additions to source code that are not modifications and contribute to a larger work can be licensed under something other than MPL.
The GPL does not allow software licensed under a GPL to be combined with a proprietary program, because a proprietary program will not give a user as many rights as the GPL (fundamental to the notion of copyleft).21 Any software using a GPL must, when distributed, publish the copyright notice and disclaimer of warranty on each copy and provide recipients of the program with a copy of the license. By modifying or distributing GPL software, a user is deemed to have accepted the terms of the GPL.

Lesser GPL

The copyleft nature of the GPL and the concern that any code written in GPL incorporated into another program will require the second program to be licensed under GPL, no matter how small a portion of the code is originally GPL, led to the development of another license.15 The Lesser GPL (LGPL) was created to allow proprietary software to be used with GPL software through the use of programming libraries (which is why it is sometimes referred to as the Library GPL). The LGPL allows the proprietary software incorporated with GPL-licensed software through a library to be licensed independently from the GPL. It is a compromise between the strong copyleft nature of the GPL and more permissive licenses such as the BSD. The LGPL allows GPL software to be linked to a non-GPL program regardless of whether it is free. In the case of programming libraries, the GPL software can be used by the library (and hence linked to other programs).22 Despite the fact that the Free Software Foundation created the LGPL, it does not encourage its use, mainly because with the exception of

<table>
<thead>
<tr>
<th>Open-Source Initiative</th>
<th>Description</th>
<th>License Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontline SMS</td>
<td>Designed for grassroots nongovernmental organizations in developing countries, helps organizations overcome communication barriers by allowing users to send, receive, and manage short message service (SMS) over a mobile network.</td>
<td>LGPL</td>
</tr>
<tr>
<td>Java Rosa</td>
<td>Open-source platform for data collection on mobile devices</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>Rapid SMS</td>
<td>Open-source framework for dynamic data collection, logistics coordination, and communication leveraging basic SMS mobile phone technology</td>
<td>BSD</td>
</tr>
<tr>
<td>SAGES (Suite for Automated Global Electronic bioSurveillance)</td>
<td>Collection of modular, flexible, OSS tools for electronic disease surveillance</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>RapidAndroid</td>
<td>A fully featured implementation of Rapid SMS that uses the mobile device to act as a standalone appliance for SMS management</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>Ushahidi</td>
<td>Initiative that creates OSS for information collection, visualization, and interactive mapping</td>
<td>LGPL</td>
</tr>
<tr>
<td>ODK</td>
<td>Free and open-source set of tools that help organizations author, field, and manage mobile data collection solutions</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td>OpenXData</td>
<td>Open-source data-collection platform that supports low-cost mobile phones</td>
<td>Apache 2.0</td>
</tr>
</tbody>
</table>
limited circumstances, it does not further the interests of free software developers.  

Eclipse Public License

The Eclipse Public License (EPL) replaced a license called the Common Public License. It has weaker copyleft provisions than the Common Public License had, and it also has a patent clause. Additions to source code originally published under an EPL license can be licensed in another way as long as the additions do not constitute derivative works of the EPL-covered source code but act as “separate modules” of software. Derivative works under EPL must also be licensed as EPL, which makes it a limited copyleft, a characteristic of the GPL license. However, the EPL requires that anyone distributing the work grant all recipients rights to any patents that may cover modifications. This patent clause is a restriction that is not compatible with GPL, so EPL and GPL works cannot be combined and legally distributed, but combined works using other licenses are permissible.

COMMON MISCONCEPTIONS ABOUT OPEN-SOURCE LICENSES

Open Source Means Free

This particular misunderstanding is perhaps the most common and is linked to the “free as in beer” way of thinking about free. OSS is provided to users at no cost, but this does not mean implementing OSS is free of cost. Although the software costs no money to download, which makes it accessible to a broader community of users, the assumption that there is no cost of ownership is faulty. For example, installation and integration of the software often requires technical expertise, and this cost can strain development budgets if the integration is complex. Maintenance of the code is another cost and requires the time of either in-house or external developers. Moreover, the defining characteristic of open source is really the ability to access the source code and not as much the fact that the source code is made available at no cost.

Open Source Has No Copyright Restrictions

As discussed previously, but worth emphasizing again, providing software as open source does not mean the developer has relinquished copyright protection. In fact, how a user is able to exploit the source code and restrictions on that use varies by license, each of which protects the rights and intent of the original author. Open-source licenses are grounded in copyright law, and the copyright holder gets to choose which rights are granted to other users.

OSS Is Unreliable

There are two components to this myth. The first part is asserted on the basis that the software is unreliable because it is either produced by amateurs or circulated without being tracked for bugs or quality. The second part of the myth is that the software is unreliable in the sense that it is uniquely insecure because vulnerabilities in the code can be easily detected. As for the first assertion, and as discussed in the introduction of this article, large, commercial software projects use OSS, and there is a high level of demand for the use of OSS in many domains. Although the software is not necessarily tracked for quality, and the licenses may not assert warranties for fitness, some software projects do have managers tracking code. Moreover, providing accessibility to a broad group of users is one of the reasons software is made open source—so others can improve the existing code. This logic speaks to the security issue as well. The transparency of OSS and the ability to improve the software are reasons many consider it more secure.

ENFORCEABILITY

The legal enforceability of open-source licenses is a nuanced and developing area. The cases vary depending on the license at issue and the facts around which enforcement is sought. Given the philosophical underpinnings of OSS, it was not immediately clear whether certain contract elements, namely the exchange of consideration, could be met given the free nature of the license. There is a growing body of case law, but for purposes of this article, it is primarily important to note that open-source licenses are enforceable under both contract and copyright law.

The case of Jacobsen v. Katzer highlights the enforceability through both legal mechanisms. In Jacobsen, the court found that if a licensee breaches a condition placed on the license grant, the licensor’s copyright has been infringed. The court also found that injunctive relief can be granted for open-source licenses. Injunctive relief is granted by a court against an act or condition, as opposed to a grant of money damages. An example in this context may be an order to stop distribution of the software by those not complying with the license terms. This type of relief is particularly important in the open-source community because monetary damages, which are typically sought in contract cases, may not be an available option as the software may have been distributed without profit. Jacobsen also confirmed that open-source licenses do not lack consideration and can therefore be enforced under contract law.

BENEFITS AND LIMITATIONS OF OSS

The open-source model has been very successful and provides developers with many benefits. First, access to
source code enables developers to improve the code, create programs that are more interoperable, and perfect their own programs that they are using OSS to develop. Access allows others to build on software in ways not envisioned by the original creators. This ability to access source code is, in part, due to the strong communities around many types of open-source licenses, and these communities provide a large pool of code from which to work. Open-source licensing provides developers who may not otherwise be able to pay for a program access to the source code, as most programs distributed under open-source licenses are free. This is particularly important in the case of resource-limited countries, which need access to similar software but do not have the means to pay for or sustain a proprietary software license. Most importantly, the broad rights that are granted to users through an open-source license provide a significant benefit because they allow users to modify, use, or distribute the software, whereas commercial licenses are usually distributed only in a form that cannot be modified.

Despite the many benefits of OSS, users of this software and those selecting an open-source license must be aware of the distinctions between the types of licenses, no matter how seemingly trivial the distinctions can be. A major consideration is whether the user wants derivative works to be proprietary, in which case a copyleft license would not be appropriate. Although there are various specialized licenses to address unique circumstances, if the developer wants to make the program open source to tap into the development community, he or she will want to pick a license that is easy for other developers to work under—probably a standard and widely used license.

Although the accessibility of the software is a fundamental characteristic of open source, most licenses contain disclaimers concerning warranties and fitness for a particular purpose. Although there is no definitive evidence suggesting that OSS is of lesser quality than commercial software (as indicated above, some in fact argue the opposite), the licensee may have to accept risks that the software has major errors. Some initiatives are large enough to provide code monitoring and bug tracking, but this is not always the case. Also, the fact that numerous people are contributing to the code increases the likelihood that code infringing on intellectual property rights (here, perhaps certain copyright terms) is introduced. Most licenses disclaim all warranties, and it may be difficult to audit the code to determine which contributor or contributors may have violated the terms of the license. An open-source project that has many authors, each of whom has a license on his or her work, makes determining who can enforce the copyright difficult (i.e., determining whether one owner can bring an action on behalf of all copyright owners or whether each must be found and joined in an action). However, the idea that OSS is more prone to claims of intellectual property infringements is generally not supported by fact, even though there have indeed been such claims against open-source development projects and there will likely continue to be such claims. The existence of these claims alone does not support the conclusion that OSS is especially vulnerable. It does, however, emphasize the enforceability of open-source licenses as legitimate intellectual property claims that can be brought before a court.

GOVERNMENT USE OF OSS

Until recently, government use of OSS has been limited, but with the expansion of mobile and cloud computing, more agencies are adopting policies for using OSS on government-funded projects. Some of the government’s reluctance stems from the sensitivity of certain data and concerns about information assurance. However, because OSS goes through continuous peer review, some argue that it is more secure than proprietary software. In particular, the DoD and NASA have embraced the use of OSS, with the latter agency being referred to as “the summa cum laude when it comes to open source” since using OSS to develop cloud computing networks. The Department of Homeland Security created the Homeland Open Security Technology (HOST) program to leverage the use of OSS in the development of technologies to support cybersecurity objectives.

In a DoD-circulated memorandum, the DoD confronted many of the previously discussed misconceptions about OSS and stated that “there are many OSS programs in operational use by the Department today, in both classified and unclassified environments.” The memorandum specifically advises that as part of the market research federal agencies must conduct to procure property or services, OSS should be included in the research when it meets mission needs. Moreover, it points out that many open-source licenses allow the user to modify the OSS for use with no obligation to redistribute, therefore quelling the misunderstanding that the DoD or any government agency would have to distribute the source code to the public, which would be prohibited on classified projects. (Note that the memorandum does outline conditions under which the code should be distributed to the public and essentially states that doing so must be in the government’s interest; the government must be authorized to release the code; and public release cannot be otherwise restricted by law.) The memorandum underscores other benefits of using OSS, including the following:

- The ability to “respond more rapidly to changing situations, missions, and future threats” because of the unrestricted ability to modify source code
- The identification and elimination of defects through the “continuous and broad peer-review enabled by publicly available source code”
- The availability of the code for maintenance and repair by the government and its contractors (rebuiting the notion that OSS comes with a limited or no warranty and therefore should not be used)
- The ability to reduce reliance on a particular vendor due to the use of OSS, which can be maintained by a variety of vendors
- The cost advantage provided by OSS, as it typically does not have a per-seat licensing cost
- The ability to widely disseminate the software, which allows the agency to contribute to a collaborative software development environment, particularly one run by the Defense Information Systems Agency (www.forge.mil/Community.html)

The work of the DoD, NASA, and the Department of Homeland Security will undoubtedly help set the trend for broader use of OSS by the government. However, the development of mobile technologies for the government is also stimulating increased use of OSS. Government-deployed mobile applications are an area of growth, and many agencies are interested in using mobile operating systems like Google's Linux-based Android for development.23 mHealth initiatives and the need for electronic processes to support healthcare (eHealth) provide particularly good examples of government use of OSS.

USE OF OSS FOR PUBLIC HEALTH INITIATIVES

The growth of global and national mHealth and eHealth needs has spurred innovation in software development. As medical practitioners and health institutions are encouraged by the federal government to digitize patient information to reap efficiency and productivity benefits of digital information, a need for sophisticated tools has arisen.28 In addition, in areas that are resource limited but where cellular technology is prevalent, mHealth solutions can dramatically improve the ability of local and nonprofit public health organizations to harness the power and potential usefulness of large amounts of health data. The monetary costs of licensing and maintaining proprietary software systems have been common challenges to these end users. Fortunately, the OSS paradigm has gained strong worldwide acceptance, and grassroots entities, researchers, and nonprofit institutions are on the frontier of developing innovative open-source tools to fulfill user needs.

The Suite for Automated Global Electronic bioSurveillance (SAGES) program at APL has been involved in the mHealth and eHealth open-source space since 2007 and has developed three open-source tools: ESSENCE Desktop Edition (EDE), OpenESSENCE, and a short message service (SMS) data collector. All three of these tools leverage OSS—this was a key design factor to ensure affordability and sustainability.

In addition to SAGES, other open-source tools have been developed by several groups to fit various needs for mobile-based data collection. Some tools and frameworks leverage others, whereas some were new software architectures entirely.

The wide selection of OSS licenses gives developers options to license their work depending on their preferences. Consequently, it is important for end users to evaluate licensing of a third-party tool before integrating it with their own projects to ensure that the license terms are not violated.

Besides mHealth and eHealth applications, the use of OSS has played a significant role in postdisaster areas such as Haiti and, most recently, the Philippines. OpenStreetMap (OSM) is a crowd-sourced mapping application that provided a detailed map of the areas hit by typhoon Haiyan within 3 days of landfall.29 The Red Cross used OSM to coordinate the volunteer effort in the Philippines, and the organization now has a policy of using OSS in all of its projects. A key reason the Red Cross cited for the adoption of this policy is the reduction or elimination of sustainment costs of software after the organization leaves an area.

In the aftermath of the 2010 earthquake in Haiti, several crowdsourcing applications such as OSM were used to map the damage. Ushahidi (http://www.ushahidi.com/about-us), a company that develops OSS for information collection and interactive mapping, led one of the main volunteer efforts to produce a crisis map.30 In an independent evaluation of the use of the Ushahidi Platform—Ushahidi's collection, visualization and interactive mapping tool—in Haiti, a report produced by a team of independent consultants noted that Ushahidi's mapping effort provided critical situational awareness that influenced operational and tactical decisions and saved lives.31 The report urged that stronger support from the nongovernmental organization community would be useful in making the application more widely used in the response community, something OSM was able to do through its relationship with the Red Cross in the Philippines. It is worth mentioning that Ushahidi staff worked collaboratively with OSM and other applications as sources of information. This partnership was critical to the company's effectiveness. The ability to share information and provide near real-time updates was facilitated by the fact that all the technology was using OSS and relying on volunteer input to improve the information's accuracy.

CONCLUSION

The variety of open-source licenses and the user-community support that accompany many of them offer tangible benefits to governmental and nongovernmental entities wishing to use and develop OSS. With the increased use of OSS by the government, many of the
misconceptions, particularly regarding risks to the user, have been dispelled. Using OSS or licensing software with an open-source license requires a clear understanding of the distinctions between the licenses and the obligations that each require for use and sharing of such software.

The public health domain provides excellent examples of how using OSS can spawn collaboration and technological advances. OSS in the public health field is a rapidly evolving space that drives innovation and brings needed tools into the hands of those often disenfranchised because of a lack of financial assets. Although skeptics have cited decreased quality and instability of OSS, there are numerous examples of reliable OSS applications that have had a significant impact in resource-limited areas.

ACKNOWLEDGMENTS: I thank my APL colleague Adjoa Poku for sharing with me her knowledge of open-source tools for mobile-based data collection and for her feedback on the development of an earlier version of this article.

REFERENCES


The Author

Erin N. Hahn is a Senior Professional Staff member in the National Security Analysis Department at APL. She conducts analysis on projects involving policy issues related to irregular warfare, international law, and information privacy. She also contributes to studies on insurgencies and unconventional warfare and supports the Predicting Infectious Disease Scalable Method (PRISM) project in APL’s Asymmetric Operations Sector. She previously worked as an attorney in private practice. Her e-mail address is erin.hahn@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
Releasing Tools for International Disease Surveillance as Open-Source Software: A Case Study

Raj J. Ashar

Since the development of tools for the Suite for Automated Global Electronic bioSurveillance (SAGES) began in 2008, the SAGES team and sponsor have envisioned the eventual release of these tools as open-source software to the global public health and technology communities. Open-source software allows members of the public to study, customize, and operate their own local copies of the software and source code, often without monetary fees. As such, releasing SAGES as open-source software assures prospective users that they retain complete control over the health data collected by SAGES-based systems, and aligns well with the model of self-sustainability intended for the operation of SAGES systems in resource-limited settings. Preparing two SAGES tools, OpenESSENCE and SAGES Mobile, for release as open-source software projects entailed a multifaceted, months-long effort that spanned policy, technical, and community considerations. This article describes the issues, trade-offs, and decisions that were addressed leading up to the successful open-source release of OpenESSENCE and SAGES Mobile in June 2013. The aim of this case study is to inform future Johns Hopkins University Applied Physics Laboratory (APL) and external efforts to release open-source software.

INTRODUCTION

Open-source software follows a design methodology that makes software source code available to members of the public for usage, modification, reproduction, and study. This methodology stands in contrast to that for commercial software, whose source code may be considered a trade secret and is often unavailable to the public. The transparency afforded by open-source software can empower end users in several ways. It can build the user’s trust that the software is operating as advertised, without surreptitiously performing any unintended operations.
on the user’s data. Additionally, open-source software can strengthen an end user’s sense of system ownership by allowing the user to customize and operate the code in a manner that satisfies the user’s requirements. It should be emphasized that releasing open-source software entails much more than providing the source code for a software application to an individual sponsor or partner organization.

The source code for an open-source software tool is generally made available in accordance with a license agreement, which is a framework of terms specifying rights and obligations for every party to the software. In many cases, open-source software developers make their tools publicly available without charging users fees for procuring or operating the software. This means that a plethora of open-source software tools for numerous purposes are freely available to anyone who has sufficient hardware and network resources to run the software. Consequently, free open-source software can dramatically lower a system’s total cost of ownership and make such systems particularly suitable for resource-limited settings.1

Since the development of the first tools for the Suite for Automated Global Electronic bioSurveillance (SAGES) in 2008, the SAGES team at the Johns Hopkins University Applied Physics Laboratory (APL) has envisioned the eventual release of these tools as open-source software to the global public health and technology communities, with the long-term goal of transitioning SAGES development and evolution to those diverse communities.2 This vision has been shared by our sponsor, the Global Emerging Infections Surveillance and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS, http://www.afhsc.mil/geis). AFHSC-GEIS recognized that developing freely available open-source SAGES tools could invest its international partner organizations more deeply in the success of deploying public health surveillance systems based on SAGES software. The open nature of the software would assure international partners that they retain complete control over the SAGES systems and the data being collected, and the free cost of software would allow partners to dedicate scarce resources toward other public health priorities. It was hoped that these advantages together would allow SAGES-based systems to be deployed and self-maintained at minimal cost to all parties involved, and to become self-initiating and self-sustaining efforts that provide international partners with greater insight into the health of their populations for years to come.

International deployments of the SAGES tools began in 2009 and have taken place across Africa, Asia, and South America.2,3 These deployments demonstrated that SAGES enhances public health situational awareness in resource-limited countries.3,4 The introduction of SAGES in these settings has succeeded in large part because countries can independently collect, analyze, and control the dissemination of population health data within their own national borders.4 As the initial goals of developing sustainable and customizable electronic disease surveillance systems were realized, interest in SAGES and requests for technical assistance started to grow.4

In addition to professional interest in SAGES among the global public health and technology communities, some teams working at prospective overseas sites expressed interest in obtaining the SAGES source code so they could verify that the software stores population health data entirely within their own national borders. Additionally, individual SAGES developers who had supported deployments at different sites began fielding e-mail requests for assistance, which complicated the team’s efforts to provide a timely, coordinated, and centralized response to common inquiries. To fulfill the sponsor’s stated direction of furthering “the development of SAGES as an open-source tool that can be installed and configured without direct U.S. support,” the impetus for releasing the SAGES tools as open-source software became clear by the end of 2012.5 Hence, the SAGES open-source release was scheduled to take place at the end of June 2013.

Successful open-source projects, such as the Mozilla Firefox web browser, are sustained over time by community interest in, and dependence on, the software.6 As such, the responsibilities of providing major enhancements, maintenance patches, publicity, technical support, and documentation eventually come to rest with a group of contributors who personally use the software—a group that is much larger than the original development team.7 Although this community interest parallels the forces of market demand that sustain other types of commercial software, the open nature of these projects enables a comparatively more nimble pace of improvement and innovation.8 Thus, the open-source model aligns well with the long-term vision for SAGES.

Although any new open-source project could potentially reuse a significant amount of existing open-source licenses, tools, and infrastructure, it will still contend with an assortment of choices and challenges.9 To distribute software to a wide audience, a project needs to select and integrate components that are compatible from both a technical standpoint as well as a licensing perspective.6 Once a project “goes live,” motivating volunteers to remain involved becomes a managerial task.7 Open-source projects that originate from private industry or government organizations may need to further satisfy institutional policy and quality requirements before release. APL project teams that intend to release their work as open-source software face distinct challenges because of APL’s roles as a university-affiliated research center and a trusted agent for the United States government (USG), which are codified in numerous legal, contractual, and procedural requirements.
Notwithstanding the challenges posed by the task of making software open source, the advantages of contributing software to the community have been recognized by the USG. Because there are circumstances under which releasing open-source software serves the government’s interests, the USG has already made numerous contributions to the open-source community. One software project funded through the DoD, the Ozone Widget Framework (OWF), was even directed by congressional mandate “to publish and maintain on the public Internet the [OWF] application programming interface specifications, a developer’s toolkit, source code,” and other resources, and to “establish a process by which private individuals and companies may voluntarily contribute” improvements to source code, documentation, and the underlying application programming interface.

At APL, there is precedent for making significant contributions to open-source software that benefit sponsors, end-user communities, and the public at large. Under the former Global Engagement Department, independent research and development funds allowed APL to implement and release key components of the .NET version of the NASA World Wind geospatial visualization tool (http://worldwind.arc.nasa.gov/index.html). Sponsor funding has enabled the Asymmetric Operations Sector (AOS) to develop and release major new features for the Xen hypervisor (http://www.xenproject.org/developers/teams/hypervisor.html) and the OpenStack cloud computing platform (http://www.openstack.org/). The release of the FEAT Editor (http://sourceforge.net/p/feateditor/wiki/FEAT%20Editor%20Home/), a tool developed with sponsor funding by the National Security Analysis Department to edit federation agreements for simulations, as a stand-alone open-source project was expedited through contractual language that specified explicitly that open-source software should be developed while also maintaining compliance with USG acquisition and export control policies.

This article describes the multifaceted effort to release two SAGES tools, OpenESSENCE and SAGES Mobile, as open-source software that can be independently obtained, customized, and operated. The release of these tools marked the first APL open-source software contribution to the global public health and technology communities. In addition to the basic decisions that must be made for every new open-source software project, the nature of this open-source release posed two fundamental challenges. One challenge was to satisfy various institution-specific requirements for ongoing public releases of technical work, while the other challenge was to make the SAGES tools readily accessible and usable for a technically diverse international audience. It is hoped that this article serves as a guide for future APL and external project teams that seek to make the fruits of their labor publicly available as open-source software.

Methods

Approach

Early in 2013, the SAGES software development team began the effort to release SAGES as open-source software. At that time, versions of both OpenESSENCE and SAGES Mobile had already enjoyed several years of operational deployment. SAGES software team members collectively share decades of software engineering, open-source, and international deployment experience. Based on the team’s expertise, three types of end users were identified as the target audience for the SAGES open-source release:

1. Information technology (IT) liaisons: Overseas IT staff who maintain and troubleshoot deployed SAGES systems

2. Public health end users: Overseas public health staff and epidemiologists who rely on SAGES systems for regular public health surveillance

3. Independent users and developers: SAGES users and software developers who are not directly affiliated with APL or the sponsor but who obtain SAGES via public Internet download and potentially contribute code to SAGES

One SAGES software engineer was tasked with leading the open-source release effort, in close collaboration with other software team members and SAGES project management. APL resources outside the project were consulted as needed for approvals and project support. These APL resources included AOS leadership, the Office of Counsel, the Business and Communications Services Department, the Information Technology Services Department, and the APL OpenStack software development team. In particular, consulting with the OpenStack team helped provide direction for the SAGES open-source release because their project already makes contributions to the open-source community and thus had worked through the various policy requirements.

Open-Source Release Considerations, Risks, and Opportunity

Releasing open-source software encompasses three broad categories of considerations: policy, technical, and community. Policy considerations pertain to the statutory, contractual, software licensing, and institution-specific requirements that an open-source software project must satisfy for the purpose of minimizing risk to all stakeholders. Technical considerations pertain to all aspects of engineering and distributing the software openly, from the stages of design, implementation, and testing, to release, packaging, and operational deployment. Community considerations pertain to building and engaging an end-user community in order to ensure
that the software remains as accessible and useful as possible. Although some aspects of these considerations may not be apparent to end users, holistically addressing all three categories is vital to the success and longevity of an open-source project. A further explanation of the tasks and decision points involved with each consideration is provided in the results below.

Risk is inherent to these considerations and to the overall endeavor of releasing open-source software. Although there are numerous websites where source code can be posted under the terms of an open-source license, there is no “cookie-cutter” template of processes and tools that will guarantee a project’s success. In fact, one study of more than 170,000 projects hosted on SourceForge found that the majority of open-source software projects fail before reaching at least three software releases. There are numerous ways a project can place itself at greater risk of failure by not effectively addressing aspects of one or more considerations. If policy issues are not resolved, this may force the open-source project to abruptly change or cease its activities, which affects existing end users who rely on the project for patch upgrades. Policy issues could further subject the project’s sponsors, development team, and independent developers to reputational harm, civil liability, or even criminal liability. Intractable technical issues adversely impact the quality and functionality of released software, which will in turn drive away prospective end users and developers. In the absence of a community that depends on the project, technical support for the software will be scarce, and there will be few enthusiastic end users to market the software by word of mouth.

That said, effectively addressing all three categories of considerations does not by itself ensure a project’s success. Rather, long-term end-user adoption and adaptation of the software is key, but that is accompanied by its own risks. If the community takes interest in the software, it is likely that the software’s functionality, underlying source code, and supporting artifacts will then be subjected to a greater degree of scrutiny and critique than previously experienced, which may make the original development team feel as though it is under siege from criticism. Regardless of the causes, the failure of an existing open-source project will strand end users, who have often invested significant time and resources integrating the software into their workflows, because it leaves those users without future security updates, technical support, and new features to improve the existing software. The consequences of a project failure may also indirectly affect stakeholders who rely on the software but do not use it directly, such as decision makers who depend on the software for critical information.

Despite the risks inherent to releasing open-source software, the strengths that SAGES already has, including the distinct niche it fills, and the potential benefits stemming from the widespread adoption of SAGES present a compelling opportunity for an open-source success that enhances global public health. SAGES leverages IT advances to improve public health surveillance capabilities in resource-limited countries, which can aid in the early detection of, and response to, disease outbreaks of international concern. The SAGES sponsor has recognized the critical importance of SAGES software for maintaining international health security and has thus continued to support this work over several years. Members of the SAGES team welcome the release of SAGES to the open-source community and are focused on constantly improving its quality and usefulness to end users. SAGES has built an existing user base across parts of the world over the last few years. Interest in SAGES continues to mount, with teams in some prospective host countries expressing the desire to verify that SAGES does indeed store their health data within their national borders. Therefore, the release of SAGES as open-source software will allow the global public health and technology communities to deploy public health surveillance capabilities in new settings, either independently or with sponsor support. Because lessons and improvements will be identified through these new deployments, the release also brings the project closer to accepting external technical contributions from public health liaisons and independent developers.

RESULTS

Overview

An overview diagram of the steps involved with the SAGES open-source software release process is provided in Fig. 1. Although technical activities are at the heart of the SAGES open-source release process, there are interdependencies between the policy and technical considerations and between the technical and community considerations. Fortunately, some activities could be executed in parallel so that the release process proceeded more efficiently.

Policy Considerations

Policy considerations need to be addressed carefully in order to minimize legal and reputational risks to the software developers, their organization and sponsors (if applicable), and software end users. In brief, the concerns underlying this category of consideration boil down to two root questions:

1. Can the software be released as open source?

2. If the software can be released as open source, what are the terms under which release would be permissible?

An open-source project in the United States must comply with numerous obligations that are imposed by
Figure 1. Illustration of the SAGES open-source software release process at APL. The three columns of this process map outline the steps involved with each of the policy, technical, and community considerations. The vertical axis of this map shows the sequence of steps that must take place during the release process. Steps marked with asterisks (**) normally take place only when a software tool is first made open source.
federal statute, funding sources and home institution (if applicable), and the third-party software on which it depends. (For the purposes of this discussion, written obligations imposed by third-party software fall under the category of software licensing obligations. Additionally, obligations that are not formally codified in a contract document fall under the category of institutional obligations.)

During this release effort, it became clear that statutory and institutional concerns were intertwined. Similarly, contractual and software licensing concerns were also found to be intertwined. The two sets of related concerns are discussed below.

**Statutory and Institutional Obligations**

From a statutory perspective, export regulations governing the release of software to non-U.S. entities vary depending on whether the software has military applications. If the software does not have military applications, its export is still prohibited to any embargoed country, to any end user who is deemed to be of concern, or in support of any prohibited usage activity. Noncompliance with export regulations may subject stakeholders of an open-source project to criminal or administrative penalties. Based on the statutory considerations, the nature of SAGES needed to be characterized early in the process in order to determine whether the open-source release should proceed.

For the SAGES open-source release, additional procedural concerns came into play. These concerns centered on embracing the open-source paradigm while minimizing risk to APL and to the sponsor. At APL, risk mitigation often involves consulting with, and obtaining approvals from, internal public release, export control, and legal experts. APL project management and the sponsor then provide oversight on an ongoing basis. In turn, these requirements dictate a need to establish formal release processes and guidelines. Throughout the release process, the SAGES team worked closely with APL experts and the sponsor to address public release and export control concerns.

Several policy decisions fundamentally shaped the SAGES open-source release. First, addressing the question of whether the release could even go ahead was crucial. Sponsor agreement for releasing the SAGES tools as open-source software had been in place from the project’s inception and was reconfirmed before the release. Early in the open-source release process, SAGES was granted internal export authorization to proceed under the condition that its functional purpose remains as a “health data collection, analysis, and reporting platform.” This approval established the basis for the release effort to proceed.

Agreement also had to be reached regarding the actual public release process that the SAGES team would follow. SAGES project management received sponsor concurrence for a proposal that met USG needs for public release review while also accommodating the dynamism inherent to the open-source community. That proposal made a distinction between major and minor software releases of functionality. Before a major release, a release notes document describing the capabilities of the release will be provided to the sponsor for review, and APL will execute structured code reviews and integration testing of those capabilities. A minor release, which may fix bugs or augment existing capabilities, will not require prior sponsor review because it falls under the scope of the documentation provided for the associated major release. Internally, SAGES was granted an exemption to improve the tools on an ongoing basis without continuously triggering the APL public release process.

With this exemption, the SAGES software development team shifted to using GitHub (the chosen hosting site, as described below) as the sole host site for APL and external development of core SAGES functionality.

These policy decisions were formally codified for future reference and may serve as a template for future APL efforts to release open-source software. Before the open-source release, an internal policy memorandum describing the release process, which incorporated documentation of key decisions, was distributed to APL stakeholders. After the release, guidance for making appropriate public commits to GitHub, including a code review checklist, was issued to the SAGES software development team.

Policy considerations directly influenced technical discussions and activities pertaining to third-party dependencies. Export concerns drove the close scrutiny of two libraries that feature encryption capabilities. One library, Spongy Castle (http://rtyley.github.io/spongycastle/), was proactively removed from the SAGES Mobile software stack because of its support for sophisticated elliptic curve cryptography. Another library, Spring Security (https://github.com/spring-projects/spring-security), was kept in the OpenESSENCE software stack after confirmation that projects already in the public domain fall outside the scope of export control.

**Software Licensing and Contractual Obligations**

Leaders of an open-source project must select the license terms under which the project asserts copyright. In turn, these terms must fully respect all contractual obligations imposed by the project’s source(s) of funding. This consideration is particularly relevant in the context of open-source software whose development has been funded by the USG, such as SAGES. Government acquisition contracts typically invoke statutes that require that the USG retain “unlimited rights” to use, reproduce, and modify software developed at taxpayer expense. Noncompliance with contrac-
tual obligations may subject an open-source project to civil penalties.

The selected license places obligations on the way end users can modify and distribute the software. One significant decision that an open-source project team must make is whether the project should be licensed under the terms of “copyleft” provisions. Essentially, this means that any work that is derived from a copyleft-licensed work must also be licensed under the same copyleft-license terms if that derivative work is itself distributed. As such, copyleft licenses are also pejoratively referred to as “viral” licenses because the original terms of redistribution are automatically passed on to any software that incorporates copyleft-licensed code. Versions 2 and 3 of the GNU General Public License (GPL) are among the most widely recognized examples of copyleft licenses, and at a minimum they require anyone who distributes modified versions of GPL-licensed software to also make available the modified source code to the recipients.

A project team’s decision regarding whether to adopt a copyleft license extends upstream to any software components that the project incorporates. Open-source software projects like SAGES generally reuse components, or libraries, from third-party open-source software. Third-party libraries provide capabilities that have already been developed, and hopefully continue to be maintained, by the community. Open-source software projects built on these libraries are obligated to fully comply, or be compatible, with the terms of the libraries’ licenses. If a library is licensed under terms that are not compatible with the license selected for the software project, then that library will need to be removed from the software and replaced before the software can be distributed. Given a copyleft-licensed library, a restrictive interpretation of the license terms may argue that a software application that merely depends on the library also becomes subject to the same copyleft terms. Consequently, an open-source project may elect to avoid distributing libraries licensed under copyleft terms, such as the GPL.

Thus, the license terms of an open-source project must be compatible with third-party library licenses, as well as the statutory and contractual obligations outlined above. Numerous available resources describe the considerations that must be weighed when an open-source project evaluates different types of licenses for potential adoption. An analysis performed more than a year before the start of this open-source release effort had recommended the Apache License, Version 2.0 (hereafter referred to as “Apache 2.0”) for OpenESSENCE.

As a license that is commonly adopted by open-source software projects, Apache 2.0 was selected in order to comply with third-party library licenses that OpenESSENCE linked to at that time, and to avoid imposing copyleft requirements on independent developers.

Unfortunately, finalizing the selection of the open-source license for all SAGES tools became complicated by the prior release of a SAGES tool under different terms. The selection of Apache 2.0 as the license for OpenESSENCE prompted questions regarding why the license agreement developed for an older version of a different SAGES tool, known as ESSENCE Desktop Edition (EDE), had not been adopted instead. (EDE is a single-user analysis and visualization tool for disease surveillance in resource-limited and disaster settings that has also been developed under the SAGES umbrella of tools. It is meant to be installed on a single computer and does not require an Internet connection. Because EDE is a mature tool that has been deployed successfully in the Republic of the Philippines, further development of EDE is not planned at this time.)

Applying the Apache 2.0 license to SAGES subsequently required striking a balance between honoring contractual requirements and fully preserving end-user rights. An appendix to the Apache 2.0 license provides a standard license notice that should be attached to works licensed under Apache 2.0; typically, this notice is copied to the top of all source code files as a header. Given that SAGES had been supported by USG funding, APL was contractually required to insert language stating USG rights in the software. In general, the boilerplate language generally used by APL to state USG rights in header notices is as follows: “This material may only be used, modified, or reproduced by or for the U.S. Government pursuant to the license rights granted under FAR clause 52.227-14 or DFARS clauses 252.227-7013/7014.” However, the SAGES team felt that including the APL boilerplate notice “as is” with the Apache header could be perceived as precluding non-USG usage of SAGES, which would contradict the intent of the open-source release. The APL Office of Counsel resolved this dilemma by updating the boilerplate USG rights language to read “This material may be used, modified, or reproduced by or for the U.S. Government pursuant to the rights granted under the clauses at DFARS 252.227-7013/7014 or FAR 52.227-14.”

A prerelease inventory was taken to confirm that SAGES relies on third-party dependencies whose licenses are compatible with Apache 2.0. That review first cataloged the libraries and corresponding licenses to which open-source distributions of OpenESSENCE and SAGES Mobile link, excluding any libraries that are used internally for testing purposes. (Because a binary-only version of EDE would be made available on the SAGES publicity website, a list of the librar-
ies and licenses to which EDE links was also compiled in the interest of due diligence. The list of licenses in use (see supplemental Table 1 at http://www.jhuapl.edu/techdigest/TD/td3204/32_04-Ashar-Tables.pdf) was compared to a list of licenses that the Apache Software Foundation has deemed “similar in terms” to Apache 2.0. OpenESSENCE was found to rely on a handful of graphing libraries that were licensed under the terms of the GNU Lesser GPL (LGPL), which is not compatible with Apache 2.0. Nevertheless, those libraries could still be redistributed because OpenESSENCE only statically links to their binary versions and does not modify any underlying source code. Additionally, three libraries used by OpenESSENCE were found to be dual-licensed under both the terms of GPL and exemptions to GPL for software licensed under the terms of Apache 2.0; in those cases, SAGES qualified for the exemptions. In summary, the third-party libraries referenced and redistributed by SAGES are licensed in a manner compatible with Apache 2.0.

Technical Considerations

The technical considerations that arise when releasing open-source software center on preparing the software and project to gracefully handle greater usage, scrutiny, and external contributions from IT liaisons as well as independent users and developers. As such, there are two types of activities that form the technical basis for an open-source release:

1. Selecting the open-source hosting site where the software will reside
2. Undertaking activities to improve the quality and maintainability of the software, so that end users can download a complete “package” of functioning software for test and modification purposes

These activities are described in the following sections.

Hosting Site Selection

A key technical decision for the project is selecting the hosting site for source code and supporting materials. First and foremost, the target hosting site will serve as the project’s technical repository, where developers across the world will go to download the source code and potentially contribute improvements back into the project. The heart of this repository is the version control system (VCS), which maintains an audit trail of all submitted changes to project files. Through this audit trail, anyone can look back at all prior commits and versions of the code base.

There are numerous VCSs available, with Subversion, Git, and Mercurial being among the most prevalent. For an open-source project, one challenge is to select a VCS that will remain popular with software developers through the foreseeable future. The hosting site may also feature bug-tracking capabilities, which can prove particularly useful when the project begins to accept bug reports and code contributions from independent users and developers. Given the importance of the hosting site to an open-source project, the project team may seek a site that shares similar objectives to their own project and that makes the project highly accessible and visible to the software’s target audience.

To begin the search for a hosting site, the team began by understanding the existing technical composition of SAGES and identifying features desired for the upcoming open-source release. OpenESSENCE and SAGES Mobile are implemented primarily in Java and JavaScript on top of open-source software technology stacks. Source code for both tools had been available internally to APL developers through a Subversion VCS, and there was discussion on the SAGES software team about transitioning SAGES to the newer Git VCS. The desire for robust workflow and collaboration capabilities, as well as the software team’s knowledge about the target user base, informed a survey to gather data for the selection of the hosting site.

The survey resulted in a thorough review of widely used open-source hosting sites in existence as of January 2013. This survey consisted of a number of criteria identified by the SAGES development team to ensure that the hosting site’s features aligned with the workflow, visibility, and sustainability goals, as well as overall purpose, of the open-source SAGES project. Alphabetic lists of the survey criteria, including their definitions, are provided in the Appendix, available at http://www.jhuapl.edu/techdigest/TD/td3204/32_04-Ashar-Appendix.pdf. The following types of criteria span the categories of interest for the open-source release:

- **Technical:** Pertinent technical details about the hosting site
- **Workflow:** Support for collaboration, communication, and receiving code contributions from IT liaisons and independent developers. (Note that the communication aspect of this type of criteria overlaps with the community considerations described below. At the outset of this effort, it had been hoped that the hosting site could also host the message forums and project web pages.)
- **Visibility:** Availability and prominence of the software project to current and potential end users
- **Focus:** Whether the site’s technical objectives aligned with those of SAGES. (For example, some hosting sites focused on technology stacks not used by SAGES, such as Microsoft .NET, or did not even focus on open-source software.)
- **Cost:** Cost for publicly hosting project software
- **Informational:** General information about the hosting site that did not enter into the selection process
The SAGES development team evaluated the candidate hosting sites against the survey criteria primarily on the basis of features advertised by their public web pages, including “Terms of Service” agreements and web pages for individual open-source projects hosted by the sites. In some cases, responses for the survey were obtained by creating a free account on a site and logging in to view its features. When responses to survey criteria could not be obtained through these means, the sites’ sales or technical support teams were contacted for clarification; responses were received to most requests for information. (Note that the actual quality of features on each hosting site was not formally assessed by this survey.) The criteria themselves were divided into two sets: preliminary and main. The preliminary criteria were used to quickly eliminate candidate sites whose features clearly did not align with the requirements for hosting SAGES. If a site did not meet all of the preliminary criteria, then the site was eliminated from further consideration. The main criteria served to collect data about potentially viable hosting sites for the purpose of informing the selection process.

A list of 30 candidate open-source software hosting sites was compiled from several sources.32–35 (Two hosting sites, GitHub and Savannah, were each evaluated twice because of the terms and conditions available to different types of site users.) Of the 30 candidates, 23 sites did not meet the preliminary criteria (see supplemental Table 2 at http://www.jhuapl.edu/techdigest/TD/td3204/32_04-Ashar-Tables.pdf). Main criteria data were collected for the remaining seven sites, including two sets of data captured for GitHub (see supplemental Table 3 at http://www.jhuapl.edu/techdigest/TD/td3204/32_04-Ashar-Tables.pdf). Based on the main criteria data, the SAGES software team narrowed the selection to GitHub (public repository) and Google Code. (For the purpose of discussion, GitHub [public repository] will hereafter be referred to simply as GitHub.)

Although these two sites lack desired features for communicating with a wider audience, such as public/private message forums, they satisfy core technical requirements and are popular with software developers. Both sites offer free hosting to open-source projects without displaying any advertising to end users. After considering the features of both sites, the SAGES software team elected to host SAGES on GitHub because of the enormous number of projects hosted by that site, as well as the site’s perceived visibility in the developer community. At the same time, the team also decided to transition from a Subversion VCS to a Git VCS because of Git’s growing popularity among developers as well as its robust support for distributed software development.

These decisions affected the technical direction of SAGES on several levels. Git allows developers to submit pull requests, which are essentially requests to review code changes before they are included in the trunk, or main code base. It also features the ability to easily make branches, or copies of an existing code base where developers can test adding their own enhancements for potential integration back into the trunk.36 Thus, the selection of the Git VCS (http://git-scm.com/) anticipates eventually incorporating enhancements contributed by independent developers. GitHub envelops a community experience around Git because it allows developers to follow colleagues, monitor projects, discuss code, and submit issue reports.37 Because open-sourcing software allows other developers to create entirely new tools from existing code, GitHub also makes forking a repository, or creating an entirely separate repository from an existing code base, very easy and traceable to the original code base.38 Immediately after these decisions were made, developers on the SAGES software team quickly learned how to use Git; they have now been using it on GitHub without difficulties for more than a year.

Thus far, GitHub has provided a good home for OpenESSENCE and SAGES Mobile (Fig. 2). As of mid-February 2014, the SAGES GitHub page links to eight repositories of code for the various components that make up SAGES.39 The SAGES software team relies on GitHub as the sole VCS for core SAGES functionality, as mentioned above. To date, the core OpenESSENCE repository is among the most active; it has seen more than 100 commits from eight contributors.40 At the start of 2014, GitHub introduced a feature for monitoring traffic volumes to the different repositories, which we will use to gauge technical interest in SAGES.41

Figure 2. GitHub home page for SAGES software projects.39
Quality and Maintainability

The prospect of releasing the software openly to the world offers an opportunity to enhance its quality and maintainability. In addition to the quality assurance activities that normally occur in the course of software development, a period of rigorous testing and resolving critical bugs should take place before the release. Approximately 2 weeks before the SAGES release, test cases for both OpenESSENCE and SAGES Mobile were developed collaboratively by SAGES public health experts and software developers on Google Drive documents. Approximately a dozen test cases were formally documented for SAGES Mobile, and several dozen test cases were developed for OpenESSENCE. These SAGES team members then executed the OpenESSENCE test cases using different versions of the Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome web browsers. SAGES Mobile test cases were executed on a variety of smartphones running different versions of the Android operating system. During the testing period, 46 bug reports were submitted and addressed. Bugs found during testing were reported in a local instance of the JIRA software issue tracker (https://www.atlassian.com/software/jira).

The impending release also spurred the reevaluation of both code and tools that support the development process. Support for building multiple modules was enhanced in OpenESSENCE by migrating the project’s build automation system from Maven (http://maven.apache.org/) to Gradle (http://www.gradle.org/). Additionally, manual scans of the code were performed to ensure that inappropriate text, such as the names of specific countries or developers, would not be included in the release. Although a small handful of instances were found and fixed, the preponderance of the code did not require any changes because writing code for eventual public release had been an objective throughout SAGES development.

Resources for Independent Use

Training resources that end users can reference at any time to help them use the software are essential to the success of open-source projects that seek longevity. As a rule, written documentation is an essential part of making any publicly released software useful. Existing English-language manuals for administering and using OpenESSENCE have been updated since the open-release to reflect enhancements that have been added to the SAGES tools (http://www.jhuapl.edu/sages/support.html). Similar manuals for SAGES Mobile were provided after the inaugural release. Given that public health end users generally lack system administration skills, an OpenESSENCE demonstration capability had already been available over the web so that prospective end users could easily try interacting with a SAGES tool without first installing it. That public OpenESSENCE demonstration capability (https://openessence.jhuapl.edu/openessence; Fig. 3) has been periodically refreshed since the release, allowing anyone to practice using its analysis and visualization features.

Additionally, the intrinsically international nature of the SAGES user base required that the software itself feature user-interface support for non-English languages from the inaugural open-source release. To support the international user base, French and Spanish translations were built into OpenESSENCE (Fig. 4) and added to the software packages that constitute SAGES Mobile (Fig. 5). These “out-of-the-box” localizations will allow users in Africa, Latin America, and other parts of the world to immediately begin deploying SAGES.

Community Considerations

Cultivating a community of end users around open-source software involves first understanding the needs of the user base. For SAGES, the resources made available to the community have to accommodate a diverse audience. The technical skill sets of SAGES users primarily span the fields of public health and IT, while the cultural diversity of the user base spans multiple languages. In addition to building support for the multilingual user base directly into the software, the SAGES team stood up a slate of resources that allow end users to both independently obtain technical support and seek support from the developers and other users. Further details on both types of resources follow below.

Figure 3. Log-in page of OpenESSENCE demonstration site.
Communication Resources

Multiple channels of communication need to be set up so that end users can learn about, and obtain technical support for, the software. These channels must be able to scale as interest in the software grows. Merely providing an e-mail address that allows users to contact development team members about issues can become burdensome quickly to developers and also impede end users from sharing best practices with each other. For SAGES, the primary channels of communication that needed to be put into place were message forums and a publicity website.

Public message forums provide a means for end users to seek technical support, suggest new features and enhancements, and relate their experiences about using the software. Because the audience of the message forums includes the development team and end users, this medium offers a means for end users with common interests across the world to connect with each other. Moreover, this medium makes support requests visible to the entire development team, which allows questions to be answered in a coordinated fashion. Contemporary message forums typically feature archiving of messages, so a searchable knowledge base of questions and discussions about the software builds up over time. Because each in-country SAGES deployment has some degree of customization to meet local requirements, the ability to also create private message forums for addressing deployment-specific concerns was desired.

Although it had originally been hoped that the hosting site would offer message forum capabilities, the survey found that the message forums offered by GitHub and Google Code did not feature all of the capabilities for mailing lists, private forums, and public forums desired for communicating with the different segments of the SAGES user base. An informal research effort in January 2013 looked at using the commercial vBulletin software (http://web.archive.org/web/20130118224503/http://www.vbulletin.com/) for message forums but found that it had the disadvantage of requiring that the SAGES project self-host and maintain it. Instead, Google Groups was adopted as the message forum solution for SAGES.

Google Groups freely hosts public and private message forums without showing advertising to forum subscribers. Google Groups offers a mailing-list feature, so subscribers are able to monitor and interact with forum discussions via e-mail, in addition to logging in with a web browser. One minor disadvantage of choosing Google Groups is that message-forum subscribers are required to possess Google accounts. However, that requirement was judged to be a minor trade-off in exchange for its overall flexibility and feature set. Three groups were initially set up on Google Groups. Two of those groups are intended for discussion among end users and developers, with the first group targeted to public health end users and the second group intended for technical discussion about SAGES. A third group has been created primarily as a mailing list for announcements about SAGES, and it is targeted to a broad audience. As of February 2014, 53 subscribers around the world who had learned about
SAGES through an APL press release or word of mouth had registered for the third group.

To market SAGES across the world, a publicity website had been developed well before this open-source release effort began. The audience for this website includes both SAGES end users and others who are interested in SAGES but may not use it directly. (This other segment of the audience includes policy makers, the media, and members of the general public.) The team felt strongly that maintaining a publicity website was essential to the continued adoption and success of SAGES. Because the open-source release offered an opportunity to rethink the branding of SAGES itself, this meant refreshing both the content and the look and feel of the website (http://www.jhuapl.edu/sages; Fig. 6) in concert with the open-source release.

The refreshed publicity website presents a curated view of information about SAGES and electronic disease surveillance, including lists of the third-party open-source libraries used by the SAGES tools. It also serves as a project hub: it links end users to the SAGES GitHub repositories and Google Groups and allows users to sign up for e-mail announcements about SAGES. The website itself makes OpenESSENCE, SAGES Mobile, and EDE executables available for download and deployment. Through the website’s integration with three popular social networks (Fig. 7), end users can spread a word-of-mouth message about SAGES to friends, colleagues, and the world. Shortly after the inaugural open-source release, a Spanish-language version of the website was established. The site was also adapted for optimal viewing on both mobile devices and PCs by applying a responsive web design approach. To draw public attention toward SAGES and the milestone of this inaugural open-source release, a joint press release was issued by APL and the sponsor.

The publicity website has been instrumental to the success of the SAGES open-source release. It won both approval and acclaim from the sponsor before the release. Since the release, web traffic to the publicity site has provided one means to gauge interest in SAGES across the globe. (The site’s web traffic is being tracked by IBM Unica NetInsight v8.6.0.0 web analytics software, http://www-03.ibm.com/software/products/en/on-premise-web-analytics/.) Between the June 2013 public release of the site and mid-February 2014, there were more than 5,000 visits to the site with approximately 24,000 page views. Within those figures, hits to the Spanish-translated portions represented 79 visits and 222 page views. Visits to the website (based on Internet Protocol addresses) have originated from countries throughout all of the continents except Antarctica (Fig. 8). Approximately one-half of the publicity site’s visits originated outside the United States, with traffic from China and India collectively representing more than 10% of the site’s visits.

DISCUSSION

Two SAGES tools, OpenESSENCE and SAGES Mobile, were publicly released
on schedule as open-source software. The open release of source code and binary executable files was complemented by making ample training and informational resources for SAGES available to all current and potential end users. Following in the vein of self-sustainability required for disease surveillance systems to be successful, the open-source hosting and communications infrastructure of SAGES is based on low-cost, sustainable components. In short, this release effort fulfilled the sponsor’s direction to “further the development of SAGES as an open-source tool that can be installed and configured without direct U.S. support.”

It serves the interests of the USG and the public to foster a vibrant open-source ecosystem by publicly releasing code and related software artifacts, when appropriate, under the terms of an open-source license. Substantial software projects for the public good, such as SAGES, would not be possible if the numerous libraries on which they are built had not been previously released by public and private entities throughout the world. Third-party enhancements to open-source software benefit both the original developers and a wider community of end users, who may even adopt the software as an industry standard and thereby reinforce this virtuous cycle. Sponsorship of the development of open-source software may even ensure that any organization, and not only the original developers, can be tasked to make future enhancements and modifications to mission-critical software.

There are multiple ways in which SAGES can build on this successful open-source release. Meeting the needs of existing users and potential new users is vital for the continued growth of SAGES, with community considerations being core to several of these needs. In the near term, the SAGES tools, publicity website, and documentation could be translated into additional languages. The collection of SAGES training resources could be expanded to include “how-to” videos posted on YouTube for all segments of the user base, as well as more developer-focused content on the wiki of the SAGES repository of technical support information and lessons learned about the SAGES tools. Active end-user involvement will also signal to prospective end users that the software is well supported and widely adopted. Nurturing a base of independent end users and developers will be essential for the longevity and long-term usefulness of SAGES. Eventually, ownership and stewardship of SAGES will transition to the community, perhaps in the form of a nonprofit organization such as the Apache Software Foundation (http://www.apache.org/).

Technical opportunities abound to make the SAGES tools even more useful and accessible to end users. Since the inaugural open-source release, the APL development team has made numerous commits to GitHub that provide incremental improvements to the SAGES tools. As the developer base grows, processes for validating and accepting external code contributions from independent developers will need to be formalized. SAGES itself will benefit by continuing to leverage, and possibly engage with, developments in the wider open-source community. Depending on future demand for EDE, its source code could also be released publicly. Most intriguingly, innovative new tools could be released under the SAGES umbrella. These tools could further extend SAGES’ capabilities in public health communications, modeling, simulation, and evaluation of automated surveillance systems. Novel SAGES tools could also take advantage of improvements in underlying technologies to rapidly acquire, process, and report on greater volumes of more-detailed clinical observations and data.

From a policy perspective, the most challenging aspect of this effort was satisfying the multiple policy considerations in a manner consistent with the overall technical, community, and programmatic objectives for the release. We recommend that future open-source release efforts create and update a wiki site that provides a concise summary of policy decisions and opinions related to the release, while also maintaining an audit trail to track all changes made on the wiki.

![Figure 8. Geographic distribution of visits to the SAGES publicity website between 28 June 2013 and 21 February 2014.](image-url)
it is also suggested that the open-source release effort occur within a short time period after the decision to go open source, so that the same core group of key contributors is available to provide the context necessary for shepherding the software all the way through release.

This effort showcases effective cross-enterprise collaboration in support of a critical sponsor mission and in service of global public health, an objective shared more widely by other divisions of the Johns Hopkins University (see, for example, http://www.hopkinsglobal-health.org and http://www.jhumhealth.org/). Its success provides a guide for existing project teams that may seek to release their work as open-source software. Project teams involved in future efforts for which open-source software is intended to be released should agree on language endorsing that intention when contracts are negotiated.13 Organizations within the defense establishment have even begun to network among each other to discuss the advantages of using and contributing to the open-source community (see http://www.mil-oss.org/).

CONCLUSION

We have described the successful effort to release existing SAGES tools for international disease surveillance as open-source software. Releasing open-source software is inherently a multifaceted task that involves addressing numerous policy, technical, and community considerations. This effort succeeded through the collaboration of a technically diverse team that benefited from the lessons learned through prior open-source releases of other software. The team was also motivated by the underlying purpose of SAGES to better global public health in a sustainable manner through the innovative application of modern information technologies.

ACKNOWLEDGMENTS: We are indebted to the entire SAGES project team and interns for their critical contributions in making this open-source release successful. We are especially grateful to Shraddha Patel for her close collaboration on the open-source release as well as her review of the initial manuscript. We also thank Aisha Ahmad, Laura Glendenning, Shaku Harshavardhana, and Tom Rossberg for their support leading up to the release, as well as Carol Brueggemeier, William Riggs, and Nigel Tseng for assisting in the research of this manuscript. Finally, we are grateful to the peer reviewers of this manuscript for offering feedback that helped improve the clarity of the discussion. The views expressed are those of the author and do not reflect the official policy or position of the DoD or the USG.

REFERENCES


18Ashar, R. J., Developer Guidelines for Releasing SAGES Code to GitHub (1.0.0), Technical Memorandum QA1-13-0014, JHU/APL, Laurel, MD (30 July 2013).


The Author

Raj J. Ashar is a software engineer in the Bio-Threat Awareness Systems Group in the Asymmetric Operations Sector and a member of the Senior Professional Staff. He led the effort to release SAGES as open-source software on GitHub and has also contributed to other public health informatics, homeland protection, and defense programs at APL since 2003. His e-mail address is raj.ashar@jhuapl.edu.
The purpose of the pilot study described in this article was to investigate whether Twitter could be a viable data source for monitoring dengue-like illness in the Philippines. The results suggest that a relatively small but sufficient number of tweets mentioning dengue-like illness in a person can be isolated from data collected from the Twitter public application programming interface. More importantly, the temporal distribution of these dengue-like tweets was similar enough to the distribution of counts of new cases of dengue-like illness in the same region to suggest that the tweets could provide a valid data source for monitoring the temporal trend of dengue-like illness. Although it is not within the scope of the project described in this article, it would be relatively easy to operationalize the use of tweets collected from the Twitter public application programming interface as a timely, valid data source for an electronic disease surveillance system.

INTRODUCTION

Dengue, also known as “breakbone fever” for the severe myalgia and joint pain experienced by patients, is a major cause of morbidity and mortality around the world. It is caused by a Flavivirus that is transmitted to humans when they are bitten by infected mosquitoes.\(^1\) There are four distinct serotypes of the dengue virus (DEN 1–4), all of which cause disease in humans that ranges from asymptomatic infection to severe, fatal hemorrhagic illness.\(^2\) Recovery from infection provides serotype-specific immunity but does not protect from infection with other serotypes of the dengue virus. Rather, repeat infection with a different serotype may be associated with increased risk of severe hemorrhagic dengue.\(^3\) The incidence of dengue has increased 30-fold since the first severe outbreaks of hemorrhagic dengue were recognized in the Philippines and Thailand in the 1950s.\(^4,6\) This rate of increase classifies dengue as an emerging infection with the potential to cause a global epidemic or pandemic.\(^7\) Indeed, dengue is now endemic in more than 100 countries around the world, putting nearly 50% of the global population at risk of infection.\(^8\) Reports of new cases to the World Health Organization...
suggest that 50–100 million people around the world contract dengue each year. Many cases are subclinical, however, so a better estimate may be closer to 390 million infections per year, of which 96 million present clinically. The recent and increasing occurrence of clusters of dengue in the southern United States and across Europe also suggests that the geographic distribution of the virus is spreading as global warming increases temperatures, creating potential mosquito habitats in formerly temperate zones.

With no effective drug therapy or vaccine, control of the mosquito vector and surveillance for clinical infections are the primary public health tools available to fight dengue. Early identification and location of outbreaks can help target intervention campaigns to reduce existing mosquito populations and breeding areas in high-risk locations. The goal of such campaigns is to minimize the spread and impact of an outbreak, but to be effective, intervention needs to start as soon as possible.

Public health authorities in many resource-rich countries use electronic disease surveillance systems to improve the timeliness of disease detection. Electronic systems allow authorities to monitor the spread of disease in a population in near real-time fashion. These systems use computerized health data from multiple sources to generate displays of the frequency of new cases of disease temporally and geographically. They can improve early identification of potential outbreaks but only if the computerized data are available quickly—ideally, the day they are collected.

Electronic disease surveillance can be especially valuable in resource-limited areas where new infectious agents frequently arise, and electronic systems targeted for these environments are available. Unfortunately, resource-limited countries, which include most dengue-endemic countries in the world, often lack the infrastructure and resources needed to rapidly digitize the health data needed for electronic disease surveillance. Medical data are often not computerized in these areas or are not computerized quickly enough to be useful in near-real-time surveillance. Although many resource-limited countries are moving toward electronic disease surveillance, implementation of data collection and data transmission protocols will take time and funding and is easily a decade or more from completion. In the meantime, other data sources are being sought that could be used now by an electronic disease surveillance system to monitor disease trends.

A number of electronic systems, such as BioCaster, HealthMap, Global Public Health Intelligence Network, and EpiSPIDER, mine publicly available electronic news media for reports of specific diseases. Some of these systems have been in use for more than a decade and have provided useful information on disease trends. Unfortunately, news reports tend to lag behind an outbreak, so mining news reports may not provide information quickly enough for public health professionals to intervene and slow the spread of an outbreak. Social media, such as the microblogging platform Twitter, provides digitized data continuously 24 hours per day. Posts on Twitter, or tweets, are limited to 140 characters or less, and users tweet to update friends on their activities and thoughts. The content of tweets, therefore, varies wildly—from social commentary to what the user is having for dinner. Most tweets are publicly available through the Twitter application programming interface (API). A pseudo-random sample of tweets meeting user-specified criteria can be obtained relatively easily and free of cost from the Twitter API. Twitter is also heavily used in many resource-limited areas where other sources for electronic disease surveillance are limited. The Philippines, for example, is among the top 20 producers of tweets in the world.

Multiple investigators have mined tweets for information about the behaviors, moods, and habits of Twitter users, and some have also looked for information to inform disease surveillance. Investigators used Twitter to monitor influenza activity in the United States during the H1N1 pandemic in 2009–2010 and noted good correlation with the number of new influenza cases as collected by public health authorities. Similarly, Collier et al. found a moderately strong association between World Health Organization/National Respiratory and Enteric Virus Surveillance System laboratory incidence data for influenza and the incidence of tweets mentioning influenza during the 2009–2010 influenza season in the United States. Outside of the United States, Chunara et al. compared the volume of cholera reports for Haiti collected from HealthMap and Twitter posts with the number of new cholera cases collected via standard surveillance methods by the Haitian Ministry of Public Health. They found a statistically significant positive correlation between the combined HealthMap/Twitter data and the incidence of cholera as collected by the Haitian Ministry of Public Health data (Pearson correlation coefficients ranging from 0.76 to 0.86). Another study by Chan et al. found significant positive correlations (Pearson correlation coefficients from 0.82 to 0.99) between the number of tweets mentioning “dengue” or similar phrases and dengue incidence as measured by public health authorities in Bolivia, Brazil, India, Indonesia, and Singapore.

If a subset of tweets that mimics the true incidence (i.e., the count of new cases) of a disease in a population could be reliably identified, it would be relatively simple to set up a continuous feed of tweets from the Twitter API, process the raw tweets to extract the appropriate tweet subset, and feed those tweets directly into an electronic disease surveillance application. This would provide an inexpensive, yet timely, surrogate disease surveillance data source.
METHODS

Study Design

The study described in this article has two objectives: to determine whether tweets mentioning dengue-like illness in an individual can be identified in the Twitter sample collected; and if so, to determine whether the temporal distribution of these “dengue-like” tweets is similar enough to the temporal distribution of new counts of dengue-like illness, as collected by Philippines public health authorities, to be used as a data source to monitor dengue-like illness in the Philippines.

Under optimal conditions, a diagnosis of dengue fever is confirmed by laboratory tests that identify the presence of the dengue virus or antibodies to the virus in the blood of a patient. These blood tests are not always available, however, and in their absence dengue is diagnosed clinically, based on the presentation of a specific set of symptoms in a patient. The clinical diagnosis of dengue used in the Philippines in 2011 was a patient presenting with fever and one or more of the following symptoms: headache, eye pain, muscle or joint pain, rash, nausea, or vomiting. The cases discussed in this article include both those confirmed to be dengue by a laboratory test and those diagnosed clinically by a physician; therefore the term dengue-like illness is used instead of dengue.

Tweet Collection

Tweets were collected using Version 1.0 of the free Twitter public API, which allows an individual to request a feed of public tweets matching specific search criteria. Each request, or query, returns a 1% pseudo-random sample of all tweets meeting those criteria, although the precise tweet selection process used by the API has not been disclosed by Twitter. Two separate search criteria were used to collect tweets for this study. The first API query asked for tweets from two areas of the Philippines for specified time periods: 18 June 2011 through 9 September 2011 for Cebu City, Philippines, and 24 July 2011 through 16 September 2011 for the National Capital Region (NCR), which includes Manila and surrounding suburbs. The second API query requested all tweets from the Twitter users whose tweets were returned by the first geographic query. Tweets from both API queries were combined for this analysis.

For tweet collection, Cebu City and the NCR were defined geographically by the latitude and longitude of a point at the center of the region and a radius in miles extending out from the central point (Table 1). The location of a tweet was recorded as the latitude and longitude of the tweeting device if geotagging was enabled on the device. For geotagged tweets, the latitude and longitude were extracted from the tweet metadata, and the closest populated place, as based on a lookup against an online gazetteer (GeoNames, www.geonames.org), was taken as the user’s location. If geotagging was not enabled, the user’s location was inferred by matching the location given in his or her Twitter profile against the gazetteer. Only tweets that mapped to a location within the specified geographic coordinates in Table 1 were retained.

Identification of Tweet Subsets

The Twitter convention @username was used to identify and remove usernames to anonymize the tweets. Duplicate tweets and retweets (tweets posted by one user and then forwarded by another user) were removed from the data set before analysis. During preliminary examination of tweets, several words commonly included in tweets not containing mention of dengue-like illness were identified and the corresponding tweets were removed before analysis.

Simple Keyword Searches for Dengue-Like Tweet Subsets

Although only a fraction of all the tweets mentioned the keyword fever, it is the only required symptom in the clinical case definition used in the Philippines, and public health authorities in Cebu City, Philippines, monitor new reports of undifferentiated fever as a surrogate measure of dengue-like illness in seasonal surveillance activities. For those reasons, this term was chosen as the focus of the simple keyword analysis. The keywords fever and feverish were examined. Dengue-like (DL) tweet subsets were created by searching tweets for those keywords in English and/or Tagalog, the native language of the Philippines. For the Fever subset, tweets containing

<table>
<thead>
<tr>
<th>Table 1. Descriptive statistics of tweet data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Central (latitude, longitude)</td>
</tr>
<tr>
<td>Radius (miles) of tweet locations</td>
</tr>
<tr>
<td>Dates of tweet collection</td>
</tr>
<tr>
<td>No. of people who tweeted</td>
</tr>
<tr>
<td>Total no. of tweets collected</td>
</tr>
</tbody>
</table>
fev or lagnat were selected, and tweets containing the words feverish or nilalagnat or may sinat or sinisinat were used to identify the Feverish Tweet subset. Each subset is labeled with the keyword it represents: Fever DL Tweet and Feverish DL Tweet.

**Human-Tagged Tweet Subset**

The Fever DL Tweets were reviewed manually to identify those actually mentioning a person who was sick with a fever, with or without other symptoms. Tweets that used the word fever in a nonclinical way were excluded. These tweets were tagged and form the human-tagged tweet subset (HT Tweet).

**Using Query Expansion to Retrieve DL Tweets**

All tweets were indexed using the Lucene text indexing and search API (http://lucene.apache.org/). High-frequency words such as articles, pronouns, and prepositions, in both English and Tagalog, were removed. URLs were converted to the tag _url_. The high-precision query used was: [(fever lagnat) AND (headache rash pain bleed* blood) NOT cold* NOT cough* NOT nose NOT _url_ NOT bieb*]. In the query, * is used to indicate that any word starting with the preceding string should be matched. This query asks for tweets that contain the word fever and one or more of the words headache, rash, pain, bleed*, or blood. It also adds the restriction that the tweet must not contain any of the words cold*, cough*, or nose; must not contain a URL; and must not contain references to Justin Bieber (i.e., _NOT bieb*). These restrictions were used to eliminate respiratory complaints and tweets containing links to health-related sites. Next, words that were most closely associated with the results of this query as compared to all tweets in the index were identified by calculating the normalized pointwise mutual information (PMI) for each word returned in the query results but excluding the original query terms where N equals all tweets:

\[
\text{pmi}(x; y) = \frac{p(x, y)}{p(x)p(y)}
\]

\[
\text{npmi}(x; y) = \frac{\text{pmi}(x; y)}{-\log[p(x, y)]}
\]

\[
p(x, y) = \frac{|\text{query results containing word } y|}{N}
\]

\[
p(x) = \frac{|\text{query results }|}{N}
\]

\[
p(y) = \frac{|\text{all tweets containing word } y|}{N}
\]

PMI is an information theoretic measure of association between two random variables. The normalized form of PMI maps the values to the range (−1 to 1), where −1 means no association, 0 reflects total independence, and 1 represents complete association. The random variables in this case are the number of tweets matching the high-precision query, x, and the number of tweets, y, in the complete index that contain a word that was contained in three or more of the returned tweets. These words form the set of words that co-occur with the query terms. Words that are more likely than not to co-occur with query terms will have a high PMI. The top 32 words, as scored by their calculated PMI, are shown in Fig. 1. Words with a strike-through were not used as expansion terms. The expansion terms were then added as a disjunction “AND-ed” to a query for fever or lagnat and run against the index again to retrieve an expanded set of tweets.

**Dengue Incidence Data**

Two sources of daily counts of new dengue-like cases of illness were used in this study: counts of dengue and dengue-like cases reported to the Philippines Integrated Disease Surveillance and Response System (PIDSR), and daily counts of the number of people presenting with fever at government-funded clinics in Cebu City, Philippines. PIDSR is an integrated disease surveillance system used throughout the Philippines to collect information about nationally reportable diseases. Incidence data for selected diseases are collected and summarized at the local level and sent forward through municipal and provincial public health authorities to the National Epidemiological Center (NEC) where surveillance data are compiled for the whole country. Use of anonymized PIDSR data for individual patients from the NCR and Cebu City in this study was approved by the NEC. As in other notifiable disease systems around the world, illnesses reported in PIDSR are thoroughly investigated so receipt of the information at NEC is often delayed. The date of disease onset and the date the case is reported are both included in each report, and the onset date was used to plot cases temporally for this analysis. PIDSR data were available for all of 2011, but only data from 8 June 2011 through 26 September 2011 were used. This period corresponds to the dates when tweets were collected plus an additional 10 days at the beginning and end of the period, allowing examination of the effect of shifting tweets forward and backward in time on the
correlation with the incidence data. To address the disproportionate sampling of cases from Cebu City and the NCR, Pearson correlation coefficients were computed to compare the temporal distribution of cases between the two locations at different points in time.

The second source of incidence data is unique to Cebu City, Philippines. The Cebu City Health Department (CHD), the public health authority for the city, has traditionally used the number of new cases of undifferentiated fever reported by government health clinics each day as a simple way to track dengue-like illness during the peak dengue season in May through December (personal communication, D. Macasoco, CHD).

In 2009, the CHD replaced its paper-based fever system with an electronic system that collects data via short message service (SMS) cellular phone messages. Each day, personnel at government clinics throughout the city send a single SMS to a dedicated phone line at the CHD for each person presenting at the clinic with fever. The SMS messages are received by a mobile phone attached to the dedicated line and are automatically transferred to a desktop computer. A custom application on the computer receives and parses the SMS for validity and stores valid messages in a database. Fever time series compiled from these data are reviewed to monitor fever incidence in Cebu City. The date of onset of fever is not included in the fever SMS data, so the date of the clinic visit is used to plot these cases temporally. The valid fever SMS messages (Fever SMS) from this system for June to November 2011 were provided by the CHD for this analysis, and as with the PIDSR data, only the data from 8 June 2011 through 26 September 2011 were used. Because the government clinics in Cebu City do not generally see patients on weekends, the Fever SMS data show a strong day-of-week effect that is not seen in the PIDSR or Twitter data because they are reported or collected daily. To facilitate comparison of the Fever SMS to the PIDSR and Twitter data, a 7-day moving average of the Fever SMS counts was computed, and the resulting daily averages, rounded to the nearest whole number, were used when comparing the Fever SMS data to other data for temporal correlation (Fig. 2).

Comparison of HT and Other Tweet Subsets with Dengue Incidence Data

The temporal distribution of the HT Tweet, Fever DL Tweet, Feverish DL Tweet, and PMI Tweet subsets were compared to the temporal distribution of the Fever SMS average counts and the PIDSR counts of new cases of dengue-like illnesses. The Fever, Feverish, and PMI Tweet subsets were also compared temporally to the HT Tweet subset. Pearson correlation coefficients were computed as a measure of agreement for the different comparisons.
To examine whether the tweet subsets might provide dengue-like illness trend information more or less quickly than Fever SMS or PIDSR counts, the tweet subsets were shifted forward and backward in time, day by day, for up to 10 days each way, and the correlation of the shifted data to the unshifted Fever SMS and PIDSR counts was recomputed for each of the daily shifts.

RESULTS

Description of PIDSR and SMS Data

The incidence of PIDSR data from Cebu City and the NCR was compared to see whether the temporal patterns of reported cases of dengue-like illness were similar in the two cities in 2011 (Fig. 3). The correlation between the cities was positive and statistically significant, albeit moderate (Pearson correlation coefficient, 0.598, p < 0.001). Comparisons were also made for the time period when tweets were collected solely in Cebu City (18 June 2014 through 23 July 2014) and for the period after the start of collection of tweets from the NCR, 24 July 2014 through 16 September 2014). Correlation in both periods was somewhat lower than in the combined period but still positive (Table 2). Given the general similarity in distribution of dengue-like case reports from the two cities, the data were combined during comparisons to the tweet data sets.

The correlation of the SMS (collected only in Cebu City) and Cebu City PIDSR data was moderate but positive and statistically significant (0.533, p < 0.001), validating the use of the Fever SMS data as a surrogate for dengue-like illness in public health disease surveillance activities (Table 2). The correlation of the SMS data to the combined Cebu City plus NCR PIDSR data was also positive (0.775, p < 0.001) and increased when the SMS were shifted to the left by six days (0.826, p < 0.001) (Fig. 4). This lag is expected because the PIDSR data are measured from date of onset and the SMS data from clinic visit, which logically follows the date of onset. Because the PIDSR data have an average 14.8-day lag from onset to data entry, however, the SMS data likely provide timelier trend data.

Description of Tweets and Creation of Tweet Subsets

A total of 15,750,771 tweets were collected prospectively from 18 June 2011 through 16 September 2011

Table 2. Pearson correlation coefficients by location and time for Fever SMS, PIDSR, and all Fever Tweets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cebu City PIDSR vs. NCR PIDSR</td>
<td>0.474</td>
<td>0.303</td>
<td>0.598</td>
</tr>
<tr>
<td>SMS vs. Cebu City PIDSR</td>
<td>0.439</td>
<td>0.198</td>
<td>0.533</td>
</tr>
<tr>
<td>SMS vs. Cebu City and NCR PIDSR</td>
<td>0.464</td>
<td>0.504</td>
<td>0.775</td>
</tr>
<tr>
<td>Cebu City Fever DL Tweets vs. SMS</td>
<td>0.625</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>NCR Fever DL Tweets vs. SMS</td>
<td>n/a</td>
<td>0.164</td>
<td>n/a</td>
</tr>
<tr>
<td>Cebu City Fever DL Tweets vs. Cebu PIDSR</td>
<td>0.393</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>NCR Fever DL Tweets vs. NCR PIDSR</td>
<td>n/a</td>
<td>0.107</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 4. Temporal distribution of PIDSR reports vs. SMS reports and SMS reports shifted –5 days.
Initially, tweets were collected only from Cebu City, but returns were relatively low. To augment the small number of tweets being collected from Cebu City, the Twitter API request was changed in late July 2011 to add tweets from the NCR. This process increased the total number of tweets collected. It also decreased the number of tweets collected from Cebu City, however, because the 1% sample of tweets returned by the Twitter API was then split between Cebu City and the NCR. Because the NCR has a much larger population, the Cebu City tweets were grossly reduced after 23 July 2011 when tweet collection began in the NCR (Fig. 5). By 1 August 2011, Cebu City contributed 1% or less of the tweets collected each day. To compensate for the decline in the Cebu City Twitter feed, the tweets from the two locations were combined during most analyses.

The content of the tweets varied wildly. Nearly a quarter (3,849,264) of the tweets were exact duplicates or retweets (Fig. 6). Review of tweets containing the term fever showed that fever had multiple meanings in the tweets. Some tweets did mention fever as a symptom of an illness in a person, but it was used most often to describe obsessive activity or strong emotions. For example, 127,958 (0.8%) of all tweets proclaimed [Justin] Bieber fever, [Harry] Potter fever, [David] Azkal fever, or a fever for some other person or place. In addition, tweets containing the term ha or ha ha (4,399,242 tweets, or 27.9%) generally meant that the word fever was being used in a joking fashion rather than as a description of illness. Removal of the I have a fever for... and the joking tweets left a total of 7,424,308 tweets. Of those, 6,235 contained the word fever (Fig. 6), and these tweets make up the Fever DL Tweet subset.

The Fever DL Tweet subset was reviewed manually to identify tweets that, in fact, used the term fever to describe a person with a dengue-like illness. A total of 4,099 tweets met that definition and are included in the HT Tweet subset. A similar query of the refined tweet set (N = 7,424,308) for tweets containing the English and Tagalog words for feverish produced 620 tweets that make up the Feverish DL Tweet subset. The more complex keyword query that used the PMI calculations was applied to the initial data set to create the PMI Tweet subset containing 940 tweets.

**Correlation of DL Tweet Subsets with Fever SMS and PIDSR Counts**

Because the HT Tweets are a subset of the Fever DL Tweets, a positive correlation was expected and observed between the HT Tweets and the Fever and Feverish Tweet subsets (Table 3). Shifting the Fever and Feverish Tweet subsets in time did not increase their correlation with the HT Tweets.
The HT Tweets were also positively correlated with both the Fever SMS and the combined PIDSR incidence counts (Pearson correlation coefficients, 0.658 and 0.712, respectively, \( p < 0.001 \)) (Table 3). Correlation with both sets of incidence counts increases when the HT Tweets are shifted forward in time, increasing the correlation to 0.745 (+6 days) for Fever SMS and 0.819 (+9 days) for the PIDSR data (Table 4 and Fig. 7). This suggests the HT Tweets could provide information on changes in the trend of dengue-like illness nearly a week earlier than the two traditional sources of dengue incidence data. If the 14.8-day average lag-time between disease onset and data entry (i.e., when the PIDSR data are ready for analysis) is included for PIDSR, the HT Tweets could lead the PIDSR data by as much as 3 weeks.

The Fever and Feverish DL Tweet subsets also showed statistically significant positive correlations with the Fever SMS and combined PIDSR incidence data counts.

### Table 3. Pearson correlation coefficients for pairs of DL subsets vs. incidence counts

<table>
<thead>
<tr>
<th>Subset</th>
<th>Fever DL Tweets</th>
<th>Feverish DL Tweets</th>
<th>PMI Tweets</th>
<th>HT Tweets</th>
<th>Fever SMS</th>
<th>PIDSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fever DL Tweet</td>
<td>1.0</td>
<td>0.849</td>
<td>0.761</td>
<td>0.920</td>
<td>0.611</td>
<td>0.601</td>
</tr>
<tr>
<td>Feverish DL Tweet</td>
<td>1.0</td>
<td>0.701</td>
<td>0.849</td>
<td>0.541</td>
<td>0.552</td>
<td></td>
</tr>
<tr>
<td>PMI Tweet</td>
<td>1.0</td>
<td>0.858</td>
<td>0.721</td>
<td>0.746</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HT Tweet</td>
<td>1.0</td>
<td>0.658</td>
<td>0.712</td>
<td></td>
<td>0.745</td>
<td></td>
</tr>
<tr>
<td>Fever SMS</td>
<td>1.0</td>
<td>0.786</td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>PIDSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4. Maximum Pearson correlation coefficients for time-shifted DL subsets vs. incidence counts

<table>
<thead>
<tr>
<th>Subset</th>
<th>SMS [Time Shift, Days]</th>
<th>PIDSR [Time Shift, Days]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fever Tweet</td>
<td>0.679 [+3]</td>
<td>0.764 [+9]</td>
</tr>
<tr>
<td>Feverish Tweet</td>
<td>0.613 [+4]</td>
<td>0.735 [+7]</td>
</tr>
<tr>
<td>PMI Tweet</td>
<td>0.721 [+0]</td>
<td>0.752 [+5]</td>
</tr>
<tr>
<td>HT Tweet</td>
<td>0.745 [+6]</td>
<td>0.819 [+9]</td>
</tr>
</tbody>
</table>

### Figure 7. Unshifted and shifted HT Tweets vs. Fever SMS 7-day moving average counts and PIDSR counts.

- (a) HT Tweets vs. SMS
- (b) shifted HT Tweets vs. SMS
- (c) HT Tweets vs. PIDSR
- (d) shifted HT Tweets vs. PIDSR
The correlations of the Fever and Feverish DL Tweets with the Fever SMS and PIDSR counts were weaker than those observed for the HT Tweet subset. As with the HT Tweets, the correlations of the Fever and Feverish DL Tweets with the Fever SMS and PIDSR counts were strengthened by moving the tweet subsets forward in time (Table 4), suggesting that the Fever and Feverish DL Tweet subsets could also provide earlier warning of changes in trends of dengue-like illness than the Fever SMS and the PIDSR incidence counts (Fig. 8). The correlation of the Fever DL Tweets was also examined by location. The correlations of the Fever DL Tweets with the SMS and PIDSR incidence counts remained positive for both Cebu City and the NCR locations, although the strength of the correlations decreased (Table 2).

The unshifted PMI Tweet subset also showed a positive correlation with the Fever SMS and PIDSR incidence data (0.7207 and 0.7464, respectively, $p < 0.0001$) (Table 3). The correlation of the PMI Tweets with PIDSR data increased when the tweets were shifted forward in time, but shifting the tweets in time did not improve correlation with the SMS data (Fig. 9). The correlation of the unshifted PMI Tweets with the SMS and PIDSR data is stronger than similar correlations observed for the Fever and Feverish DL Tweet subsets, but this advantage was reduced when the DL Tweet subsets were time-shifted (Table 4).

**DISCUSSION**

This study identified several keyword-based methods used to isolate tweets from users in two locations in the Philippines who mention dengue-like illness in a person. The study showed that the temporal distribution of those tweet subsets is similar to the temporal distribution of counts of new dengue-like illness as recorded by Philippines public health authorities. Although the results are encouraging, this study was an exploratory pilot study with several limitations. The study addressed only a single disease in one country. To use Twitter as a source for electronic disease surveillance, the same preliminary work needs to be repeated for each illness monitored from Twitter data. Although using keyword permutations of the term fever was successful in the Philippines, the keywords will vary by location, if only because of language differences. The keyword distribution may also change over time, so ongoing evaluation
Figure 9. Unshifted and shifted PMI DL Tweets vs. Fever SMS 7-day moving average counts and PIDSR counts. (a) PMI DL Tweets vs. SMS; (b) PMI DL Tweets vs. PIDSR; (c) shifted PMI DL Tweets vs. PIDSR.
and update of the keyword set(s) would be needed if the
tweets were used for disease surveillance long term.

There are also questions about the repeatability of
data collected from the Twitter public API. The data
from the API is, presumably, a pseudo-random 1% sample of tweets identified by the API queries used,
but Twitter has not disclosed the exact methods used to
create the 1% sample. It is, therefore, possible that
the outcome of sampling will vary by location, within
a given location, over time, or by all these factors. This
problem needs further evaluation before the API is used
routinely in disease surveillance, because the cost of
obtaining a larger ongoing Twitter feed is prohibitive in
resource-limited areas.

The most serious limitation of this project was the
decline in Cebu City tweets due to procedural changes
in tweet collection. Separate analysis of the tweets by
location show similar but weaker correlations to the
Fever SMS and PIDSR data, suggesting that combina-
tion of tweets from Cebu City and the NCR did not bias
the study results.

Adding a Twitter data feed to an electronic disease
surveillance system would be relatively easy. Customized
code would need to be written to capture the continuous
feed of data from the free Twitter API for the appropri-
ate geographic areas and then to write it to a database.
Tweets stored in the database would need to be processed
automatically to isolate those mentioning the specific
illness being monitored. Hand-tagging the tweets con-
taining information on fever would be too cumbersome
for this process, but automated processes like those used
to create the Fever DL and PMI DL Tweet subsets could
be easily adapted for this purpose. Last, the tweets men-
tioning the specific illness would need to be visualized
in an electronic disease surveillance system.

The primary advantages of using the Twitter data
in disease surveillance are speed and cost; the Twitter
public API is freely available and provides near-real-time
computerized data. This type of surrogate data would
augment, not replace, traditional public health disease
surveillance. Its purpose is to help public health per-
sonnel identify and intervene in disease events rapidly,
hopefully limiting the impact of the event. Traditional
disease reporting is still needed to provide detailed,
specific information on the incidence and movement
of disease through a population, and the public health
community must continue to move toward fully auto-
mated disease surveillance. Although further evaluation
is clearly needed, this study suggests that the Twitter
public API could provide a free source of disease inci-
dence data for use in electronic disease surveillance.

ACKNOWLEDGMENTS: We thank Howard Burkom for his
assistance in analysis, as well as the personnel in the
Philippines AFRIMS Virology Research Unit. The opinions
or assertions in this article are the private views of the
authors and do not necessarily reflect the official policy
or position of the U.S. Department of the Army, the U.S.
Department of Defense, or the U.S. government.

REFERENCES
1Public Health Agency of Canada, “Dengue Fever Virus (DEN 1,
DEN 2, DEN 3, DEN 4) – Material Safety Data Sheets (MSDS),”
(last modified 18 Feb 2011).
2New York State Department of Health, “Dengue Fever (Breakbone
Fever, Dengue Hemorrhagic Fever),” http://www.health.ny.gov/dis-
3Simmons, C. P., Farrar, J. J., van Vinh Chau, N., and Wills, B.,
4Gubler, D. J., “Epidemic Dengue/Dengue Hemorrhagic Fever as a
5Guebler, D. J., Clark, G. G., “Dengue/Dengue Hemorrhagic Fever:
The Emergence of a Global Health Problem,” Emerg. Infect. Dis. 1(2),
6Kyle, J. L., Harris, E., “Global Spread and Persistence of Dengue,”
7World Health Organization Dengue Control,” http://www.who.int/
8Centers for Disease Control and Prevention, “Dengue Epidemiology,”
http://www.cdc.gov/dengue/epidemiology/index.html (last updated
9Bhatt, S., Gerthing, P. W., Brady, O. J., Messina, J. P., Farlow, A. W.,
Number, and Impact of Climatological Events for Multi-wave Dengue
13Khasnis, A. A., Nettleman, M. D., “Global Warming and Infectious
2013, 690285 (2013).
15May, L., Chretien, J.-P., and Pavlin, J. A., “Beyond Traditional Surveil-
ance: Applying Syndromic Surveillance to Developing Settings—
16Patterson-Lomba, O., Van Noort, S., Cowling, B. J., Wallinga, J.,
Gomes, M. G., et al., “Utilizing Syndromic Surveillance Data for Esti-
mating Levels of Influenza Circulation,” Am. J. Epidemiol. 179(13),
17Samoff, E., Fangman, M., Hakenerwether, A., Ising, A., and Waller, A.,
“User of Syndromic Surveillance at Local Health Departments:
18Borchert, J. N., Tappero, J. W., Downing, R., Shoemaker, T., Behu-
19Collier, N., Doan, S., Kawazoe, A., Goodwin, R. M., Conway, M.,
et al., “BioCaster: Detecting Public Health Rumors with a Web-based
20Nico, “4 Ways How Twitter Can Keep Growing,” Peerreach blog,
blog.peerreach.com/2013/11/4-ways-how-twitter-can-keep-growing/
(7 Nov 2013).
21Lyon, A., Nunn, M., Grosset, G., and Burgman, M., “Comparison of
Web-based Biosurveillance Systems: BioCaster, EpiSPIDER and
22Keller, M., Brench, M., Tolentino, H., Freifeld, C. C., Mandl, K. D.,
et al., “Use of Unstructured Event-based Reports for Global Infectious
23Freifeld, C. C., Mandl, K. D., Reis, B. Y., and Brownstein, J. S.,
“HealthMap: Global Infectious Disease Monitoring through Auto-
mated Classification and Visualization of Internet Media Reports,”


The Authors

Jacqueline S. Coberly is an APL infectious disease epidemiologist who applies her academic and field experience to the development, implementation, and evaluation of electronic disease surveillance systems. Clayton R. Fink is a senior software engineer at APL focused on applying natural language processing and machine learning in different data domains. Yevgeniy Elbert is an APL statistician who contributes to data analysis and method implementation and evaluation for electronic disease surveillance tools developed at APL. In-Kyu Yoon is a physician and virologist and currently Chief of the Department of Virology at the U.S. Armed Forces Research Institute of Medical Sciences. His colleagues from the Philippines AFRIMS Virology Research Unit, John Mark Velasco and Agnes D. Tomayao, coordinated the field aspects of the project. Three additional medical epidemiologists from the Republic of Philippines were essential to this study for arranging access to critical data sets and providing perspective on the challenges of disease surveillance in the Republic of Philippines. They are Vito Roque Jr., Head of Public Health Surveillance and Informatics Division of the National Epidemiology Center; Enrique Tayag, Assistant Secretary, Public Health Surveillance and Informatics Division of the National Epidemiology Center; and Durinda Macasocol, Assistant Epidemiologist at the Cebu City Health Office. Sheri H. Lewis is the Global Health Surveillance Program Manager in the Asymmetric Operations Sector’s Homeland Protection Program Management Office and is responsible for development of business opportunities in public health. For more information on the work reported here, contact Jacqueline Coberly. Her e-mail address is jacqueline.coberly@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
Development of the Respiratory Disease Dashboard for the Identification of New and Emerging Respiratory Pathogens

Timothy C. Campbell, Zarna S. Mistry, Gabriel N. Gorelick-Feldman, Charles J. Hodanics, Steven M. Babin, and Sheri H. Lewis

After the 2009 H1N1 influenza pandemic, the Global Emerging Infections Surveillance and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS), asked the Johns Hopkins University Applied Physics Laboratory (APL) to develop a system that would allow for easier collection and visualization of respiratory disease data collected from their worldwide laboratories. As part of the Suite for Automated Global Electronic bioSurveillance (SAGES) program, APL developed the Respiratory Disease Dashboard (RDD) as a secure Internet-accessible database with user-friendly entry, analysis, and visualization of infectious disease laboratory data. Global AFHSC-GEIS laboratories, as well as other partner laboratories in various countries, use RDD to submit their weekly respiratory disease laboratory data to AFHSC-GEIS; RDD also serves as their central repository for these data.

INTRODUCTION

Since the 1940s, there has been a significant rise in occurrences of infectious diseases emerging in the human population for the first time. Emerging respiratory diseases can spread rapidly and globally, especially with international air travel. This rapid spread is primarily a result of the diseases’ mode of transmission: infected people produce aerosolized droplets when coughing or sneezing, and others inhale those droplets or infect themselves by touching contaminated surfaces and then touching their eyes, mouths, or noses. Resistance to antibiotics and antivirals is increasing at a rate that threatens to limit the arsenal of drugs to fight respiratory diseases caused by bacteria and viruses, respectively. For viral respiratory diseases, vaccination is usually effective if it occurs at least 2 weeks before initial exposure and if the vaccine contains the same or similar antigens as the current circulating virus. Influenza viruses are of particular concern because their high rates of mutation mean that vaccines must be reformulated at least annually. Rapid identification of emerging respiratory diseases can provide longer lead times for the development of prevention strategies such as drugs and vaccines. The 2003 Severe Acute Respiratory Syndrome (SARS) outbreak and the 2009 H1N1 influenza...
pandemic emphasize the importance of improved global disease surveillance.

After the 2009 influenza pandemic, the Global Emerging Infections Surveillance and Response System, a division of the U.S. Armed Forces Health Surveillance Center (AFHSC-GEIS), asked the Johns Hopkins University Applied Physics Laboratory (APL) to develop a system that would allow for easier collection and visualization of respiratory disease data from U.S. DoD laboratories on five continents around the world. These laboratories, along with their partner network laboratories, support global surveillance for febrile and vector-borne infections, gastrointestinal infections, antimicrobial-resistant organisms, sexually transmitted infections, and respiratory infections. Among these surveillance objectives, monitoring influenza and other respiratory infections is a continuing priority for the AFHSC-GEIS network.5,6

The AFHSC-GEIS respiratory disease surveillance network was dramatically expanded in 2008, processing more than 26,000 specimens from 491 collection sites in 75 countries.7 This expansion of respiratory disease surveillance within the AFHSC-GEIS partner network has underscored the need for a robust data management system to log, analyze, monitor, and visualize trends among respiratory pathogens. Management of data generated by such a diverse collection of partners requires a flexible system that is user friendly for a wide array of personnel. This article describes the Respiratory Disease Dashboard (RDD), an Internet-based system designed to serve as a central database for visualizing and tracking respiratory pathogens collected in the AFHSC-GEIS partner network. RDD makes significant contributions to

**DEVELOPMENT AND FEATURES OF RDD**

**Original RDD Prototype**

The initial web-based RDD application was built on a Grails stack. Grails (http://grails.org) is an open-source “coding-by-convention” web framework that allows programmers to develop web applications quickly. Coding by convention, also known as “convention over configuration,” allows for rapid prototyping with maximum flexibility. Grails applications are mainly written in Groovy (http://groovy.codehaus.org),8 an open-source dynamic language for the Java Virtual Machine9 platform. To support users’ requests for geographic information system (GIS) mapping capabilities, PostgreSQL (http://www.postgresql.org) geographic object-relational databases, along with the PostGIS (http://postgis.net) spatial extensions, were added on the server to support location-based SQL queries. OpenLayers (http://openlayers.org), a JavaScript mapping library, was added on the client to integrate mapping interface components. The front end was updated to use the open-source framework ExtJS (http://www.sencha.com/projects/extjs/) to provide interactive JavaScript widgets for additional flexibility with the interface.

The initial RDD prototype focused on rapidly communicating reports of novel respiratory pathogens from AFHSC-GEIS partner labs. In the case of influenza,
isolation is identified using nomenclature established by the International Committee on Taxonomy of Viruses. Two types of influenza viruses are known to cause seasonal epidemics in humans: influenza A and influenza B. Influenza A viruses are further divided into subtypes that are classified by their two proteins (antigens) called hemagglutinin (H) and neuraminidase (N). A particular influenza virus is then identified as H1N1, H3N2, etc. It was important for RDD to have an easy means of organizing and tracking changes in these subtypes, as their classification and tracking aids in vaccine development and monitoring of disease transmission. Therefore, prototype algorithms were established to identify and highlight on a map (Fig. 1) the greatest percent change in the number of new cases during the most recent 2-week interval compared with the previous 2-week interval. In the case of influenza, algorithms were used to determine the number of new cases in the current week for an influenza subtype divided by all new cases reported for the current week (e.g., the number of H1N1 influenza cases/the number of all influenza cases) subtracted by the number of cases in the previous week for the same influenza subtype divided by all the cases reported for the previous week. A positive result was displayed as an increase from week A to week B, whereas a negative result was displayed as a decrease. Users could then use the map display to observe and monitor the region or country flagged with the greatest percent change over the last 4 weeks of data. The map provided a filter window for users to look at all observed categories, category groups (user-defined sets of categories), and individual categories. Built using the GeoExt (http://geoext.org) JavaScript toolkit for integrating OpenLayers with ExtJS, this interactive map provided zoom, pan, and selection capabilities. Selection via click-through displayed a time series with 90 days of observed values for the selection as well as an informational tab detailing the labs supplying data to the region/country. The system was configured to allow AFHSC-GEIS administrators to access data at the country level to observe counts. The other available user roles, denoted lab and dashboard, could access only calculated percentages by week at the country level.

**RDD DEVELOPMENT**

As often happens with information technology initiatives, once users were able to visualize their initial requirements via the prototype, they realized they desired additional features and capabilities. After the RDD prototype began to be used, the APL development team proposed migration of the RDD prototype to leverage components of the ongoing Suite for Automated Global Electronic bioSurveillance (SAGES) OpenESSENCE (OE) development effort. Using the OE backbone for RDD took advantage of OE’s data-agnostic design that could easily incorporate the RDD data sources. Therefore, RDD was migrated from Grails to OE. However, additional functionality was needed within OE to support several of the customized visualizations and interfaces required by RDD, such as a conditional report panel, interactive dashboard map, and role-based access to data. These features were initially developed as static, custom interfaces but were later incorporated within the core OE system so that any data source in addition to RDD could use these features.

**Flexibility for Use by Both AFHSC-GEIS and Partner Global Laboratories**

To more closely match data collection by the global partner laboratories, RDD has undergone several redesigns of the back-end data models. Initially, the data were to be collected by AFHSC-GEIS regional labs, where each laboratory reported information for one or more countries. The report’s web form was arranged in a tabular format, with categories listed vertically in rows and each associated country in a column. Although data were and continue to be reported weekly, data were submitted only as an aggregate count by category, with each country’s disease information supplied by a regional AFHSC-GEIS lab. The data collection model was later modified to allow the creation of additional in-country laboratory sites so that one or more labs within each country can submit a data report. So that non-AFHSC-GEIS labs can be included in the future and retain the correlation with past AFHSC-GEIS data collection, the new in-country site can be associated with a AFHSC-GEIS regional laboratory. Therefore, the new RDD version can continue to incorporate results from AFHSC-GEIS laboratories as well as those from in-country laboratory sites. In addition, the new RDD version allows information to be provided via separate country and category reports. That is, when entering data via a new web-based report, users can choose from filtered pick lists to select a country and then an associated in-country site. In most cases, these site associations are currently limited to a single country and one reporting site; however, RDD can now accommodate the need for a single regional AFHSC-GEIS site to report information from one or more countries and/or in-country sites.

**Automatically Accounting for Lags in Test Results**

The latest data model in RDD is designed to align better with how test results are captured by different laboratories. Although a patient’s respiratory specimen may be tested as soon as it is collected, different tests take varying amounts of time. For example, rapid influenza diagnostic tests may give results within 30 minutes, indirect or direct fluorescent antibody tests take several hours, polymerase chain reaction may take from one to several hours, and viral culture takes several days or even weeks. These different types of tests are all useful because they have different capabilities for identifying
different diseases. Furthermore, each lab may be capable of performing only certain types of tests. Because of the lag in availability of test results, laboratory personnel previously had to backfill data and recalculate counts manually. The users had to manually edit several weeks of reports for each new week of data. RDD now uses the sample collection date to take into account these lags in availability of test results, and it automatically backfills previous weeks’ data and updates the counts.

Enhancing Data Accountability and Reliability

For accountability and reliability, RDD also provides robust information about the data. Multiple fields routinely used by the laboratories were isolated to allow RDD to collect information from the results of individual lab samples and use that information to calculate the multiple fields that users previously computed manually. Therefore, RDD now allows the labs to submit more information for tests during the current reporting week instead of inputting only aggregate data for a lab. Sometimes a single test may report multiple results because patients may have more than one pathogen (e.g., co-infection). Therefore, RDD now provides the user with the ability to add information about each test, including a sample identifier, the type of test and how it was performed, and whether the results revealed more than one pathogen. These data are automatically added to the RDD disease counts so that the total values, including data from co-infections, are computed automatically. The new system now automatically associates results with the relevant reporting week/date by using information from the sample identifier and the collection date and location to compute values for co-infections and total pathogens isolated by week. Therefore, labs can submit data as results become available, and the new RDD eliminates manual data backfill. Another advantage of submitting data in a single report as results become available is that data could theoretically be submitted daily if individual labs see large numbers of samples. The user can also submit lab reports in bulk across multiple weeks of data without the manual post-processing required to compute and update values across multiple weeks of data. Having more information available about each laboratory test serves as a check on the accountability and reliability of data entry from each laboratory. For example, one can observe that the entered results are consistent with the type of test described.

Accommodating Differences in Weekly Data Reporting

Although each laboratory reported its results weekly, many labs used different definitions for the beginning of the week, and these definitions varied depending on local reporting requirements. It became apparent in the original version of RDD that such reporting could cause discrepancies in the way the disease presence was represented across multiple laboratories. To avoid this problem, the new version of RDD allows the user to record the actual collection date for the test and keeps track of that date when the results become available. Therefore, RDD does not have to rely on each lab’s definition of the reporting week and can instead standardize the weekly reporting to allow for more accurate comparisons of disease activity across multiple labs and regions. To allow for data comparison, the U.S. Centers for Disease Control and Prevention uses a standardized method for counting weeks (http://wwwn.cdc.gov/nndss/document/MMWR_Week_overview.pdf). The first “epi week” of each year is defined as ending on the first Saturday in January as long as that Saturday falls at least 4 days into the month. After week one, then each “epi week” begins on a Sunday and ends on a Saturday. Actual collection dates are used and then translated within RDD. The user now has the ability to use the CDC “epi week” definition or one that is requested by his or her sponsoring agency.

Allowing Easier Data Tracking and Mapping

The newer version of RDD allows the use of several other data attributes to help track and map the data. For example, a user can include a data flag to exclude mapping of multiple mobile or ship-based laboratory stations. Doing so allows the user to exclude labs that are considered too mobile, thereby avoiding tracking inconsistencies.

Other attributes include the ability to denote the in-country site populations and assay classifications. These in-country site populations include categories used for disease surveillance purposes, such as U.S. military, local civilian, those presenting with influenza-like illnesses, etc. An assay classification is a group of diagnostic tests (i.e., assays) assigned to each lab by AFHSC-GEIS on the basis of local lab capabilities. For example, a particular lab may be capable of only certain types of tests, whereas other labs may be able to do tests that are more sophisticated. One or more tests may be available per site and are included in the assay classification for that site. These assay classifications are used in the new RDD to maintain compatibility with previous aggregate disease reporting so that data can be compared over multiple years.

Flexibility in Laboratory Data Collection Modalities

Another challenge was that different laboratories used various data collection and storage modalities. The RDD user-defined template mirrored how sample information was already being collected at most labs, but other labs incorporated sophisticated local databases that require assistance to export the data into the correct weekly reporting template for RDD. For example, some labs record their test results in text files, Microsoft Excel files, Microsoft Access files, Statistical Analysis Software (SAS) files, or other file types. These labs may be submitting their results to multiple entities in addi-
User Interface and Data Input

In the latest version of RDD, the customized data entry panel provides a role-based access control interface for AFHSC-GEIS partner laboratories to enter data directly into RDD. This role-based approach, implemented with Spring Security (http://projects.spring.io/spring-security/), means that the permissions for accessing different features are based on the job functions of the user. The access control interface also filters the lab and country reporting fields to those the site administrator has assigned to the particular user role. For secure data transmission, Hypertext Transfer Protocol Secure (HTTPS) is used. A custom data entry grid with reporting countries as columns and reporting categories as rows provides users with an interface that is aligned with reporting requirements. The grid also provides a means to dynamically adjust the number of columns and rows as countries and/or categories are added or removed from the system. Figure 2 illustrates the data input interface, showing the location and different types of influenza.

Bandwidth Challenges in Resource-Limited Settings

Another challenge has been the speed of RDD in resource-limited settings. Although many of the reporting laboratories are operating on U.S.-acquired equipment, the labs themselves are still located in resource-limited settings where data transmission may encounter slow speeds and limited bandwidth. The team examined the RDD data structures for ways to reduce the bottlenecks in the RDD web application and found that certain code and debugging libraries could be eliminated. For example, some of the JavaScript library and debug files were more than 2 MB. These JavaScript libraries could be minified to sizes less than 200 KB. Doing so reduced the bandwidth requirements while maintaining data accountability and reliability.

Data Visualization

The RDD map portal enables filtering of data to show various category groups and data for different time periods. For example, the results could be filtered by type of respiratory disease, such as influenza A, influenza, other non-flu pathogens, all pathogens, and all specimens. In addition to countries, the user can select regions including...
RDD supports time-series graphing and the application of outbreak detection thresholds that assist the user in visualizing weekly biosurveillance results (Fig. 4). Either single or multiple pathogen results can be plotted over time. Figure 4 uses simulated data for H1N1, pandemic H1N1 (pH1N1), H3N2, influenza B, and adenovirus to illustrate plots of multiple pathogens. The plots for each type of pathogen may also show warnings (yellow dots) or alerts (red dots) based on particular algorithm thresholds for that pathogen. These algorithms are described in detail elsewhere (e.g., Ref. 15).
Because RDD can now integrate the older aggregate reports with the newer data entry methods, users can make year-over-year comparisons, allowing them to compare past seasons with each other and with the current season (Fig. 5). These data can also be filtered by specific pathogens (including different influenza subtypes), pathogen groups (e.g., all influenza), country, region, assay type, surveillance population, etc. Examination of these time series may also allow users to spot sudden changes in disease reporting.

Role-Based Data Analysis and Results Reporting

In addition to the visualization of analytical results described above, RDD supports data reporting at different levels that vary with the role of the user. Country-level reporting allows laboratory users role-based access to data for all countries aggregated to the country level but not below. In-country laboratory users are allowed role-based access to more detailed all-level report data but only for their particular countries. Individual laboratories can view their own data at the individual sample level with a tabular view and can export weekly data capturing all fields specific to their weekly reporting needs. A combined report allows users to query both legacy aggregate data and the newly adopted individual lab sample reporting data. These data were combined by aggregating the individual lab samples to align with the legacy aggregate report data. In the combined data reports, users are able to run multiple queries that span the transfer to the new collection system, including site comparison reports and a newly developed year-over-year time-series plot. RDD also supports tabular output of data, including pivot tables, for more advanced analysis in other statistical software packages commonly used in epidemiology.

Administrative Functions and Determining Reporting Latency

Several administrative reports and site maintenance portals have been added to the RDD web application. Via the web interface, administrators have the ability to create and modify almost all attributes related to continuing system expansion. These functions include administering user accounts, managing reporting sites, and updating all reported data fields labels, associations, and colors. For example, administrators can use the role-based access control to assign individual users to different roles that allow different types of data access.

To maintain the timeliness of the submitted reports, an administrator can perform a time-series analysis (Fig. 6) to determine the number of days from an expected report date to the actual date the report was submitted and the number of days since the last modification to the expected report date. RDD allows the administrator to determine minimum, maximum, and average values for both of these metrics. The administrator can thereby monitor reporting quality by observing latency in reporting and modifications/backfilling of data.

![Figure 5. Year-over-year time-series plot showing seasonal disease variations.](image_url)
DEVELOPMENT OF THE RESPIRATORY DISEASE DASHBOARD

JOHNS HOPKINS APL TECHNICAL DIGEST, VOLUME 32, NUMBER 4 (2014)

labels, groupings, and user accounts. APL still typically handles additional reporting components requiring core system changes, and the system will likely always require a developer in the loop. An ongoing effort for the core OE project is to redesign the core system to adhere to accepted and well-documented standards so that it is easier to transition future development efforts to the user community.

CONCLUSIONS

APL developed RDD as a secure Internet-accessible means of collecting and sharing laboratory results in a respiratory disease database with user-friendly entry, analysis, and visualization of infectious disease data. RDD serves as the central repository for respiratory disease specimen data generated by the AFHSC-GEIS laboratories. These laboratories support surveillance efforts on five continents, test hundreds of respiratory isolates each week, and report influenza subtype information to AFHSC-GEIS headquarters. Additionally, many partner laboratories perform an expanded panel of respiratory pathogen tests that include other viral and bacterial pathogens.

Future Efforts and Sustainability

There are continuing efforts to migrate the custom elements of RDD into the core OE open-source release. To quickly respond to sponsor/end user requirements, features are often developed statically and loaded into the web application dynamically, outside of the core OE code base. When users identify desirable features, the features are typically folded back into the core OE code base and integrated with the Groovy data source configurations for inclusion in other SAGES projects/efforts.

APL designed RDD so that local administrators can use the interface to configure essentially all components needed for day-to-day system administration. This includes any fields that have associated colors, labels, groupings, and user accounts. APL still typically handles additional reporting components requiring core system changes, and the system will likely always require a developer in the loop. An ongoing effort for the core OE project is to redesign the core system to adhere to accepted and well-documented standards so that it is easier to transition future development efforts to the user community.

CONCLUSIONS

APL developed RDD as a secure Internet-accessible means of collecting and sharing laboratory results in a respiratory disease database with user-friendly entry, analysis, and visualization of infectious disease data. RDD serves as the central repository for respiratory disease specimen data generated by the AFHSC-GEIS laboratories. These laboratories support surveillance efforts on five continents, test hundreds of respiratory isolates each week, and report influenza subtype information to AFHSC-GEIS headquarters. Additionally, many partner laboratories perform an expanded panel of respiratory pathogen tests that include other viral and bacterial pathogens.

In addition, an administrator can generate a high-level lab/week report to quickly view reporting statuses by week. This report includes a grid (Fig. 7) styled with colored checkmarks and "X"-marks to help quickly identify latent reporters. Data can be aggregated/grouped by region, country, AFHSC-GEIS laboratory hub, and in-country site.

Future Efforts and Sustainability

There are continuing efforts to migrate the custom elements of RDD into the core OE open-source release. To quickly respond to sponsor/end user requirements, features are often developed statically and loaded into the web application dynamically, outside of the core OE code base. When users identify desirable features, the features are typically folded back into the core OE code base and integrated with the Groovy data source configurations for inclusion in other SAGES projects/efforts.

APL designed RDD so that local administrators can use the interface to configure essentially all components needed for day-to-day system administration. This includes any fields that have associated colors, labels, groupings, and user accounts. APL still typically handles additional reporting components requiring core system changes, and the system will likely always require a developer in the loop. An ongoing effort for the core OE project is to redesign the core system to adhere to accepted and well-documented standards so that it is easier to transition future development efforts to the user community.

CONCLUSIONS

APL developed RDD as a secure Internet-accessible means of collecting and sharing laboratory results in a respiratory disease database with user-friendly entry, analysis, and visualization of infectious disease data. RDD serves as the central repository for respiratory disease specimen data generated by the AFHSC-GEIS laboratories. These laboratories support surveillance efforts on five continents, test hundreds of respiratory isolates each week, and report influenza subtype information to AFHSC-GEIS headquarters. Additionally, many partner laboratories perform an expanded panel of respiratory pathogen tests that include other viral and bacterial pathogens.

Figure 6. Time-series analysis of results reporting latency.

Figure 7. Grid showing AFHSC-GEIS hub site reporting.
bacterial agents. Medical laboratory technicians at partner institutions can enter data directly into RDD via a web-enabled form. These data become instantly available for analysis by the laboratory user. Through RDD, the user can tabulate disease prevalence, distribution of influenza subtypes, and the frequency and distribution of other respiratory pathogens. These tabulations can be filtered by geographic region, assay type, and surveillance population. To provide an instant visual snapshot of influenza subtype distribution worldwide, the main RDD page overlays pie charts on global maps that can also depict the WHO disease transmission zones (Fig. 3).

The development of RDD has helped the APL team to identify a number of informatics issues that provide overarching benefit to many similar initiatives at APL. Using the OE backbone benefited both the SAGES and RDD projects by minimizing duplicative architecture and security efforts. In addition, the overall ability of both systems to ingest data improved, as did the team’s ability to analyze and test the systems. The decision to merge the projects was based on the sponsor’s desire to continue the RDD effort, the available resources currently working on the OE platform, and the similarities in system requirements. Instead of building in parallel two separate systems with similar functionality, APL made better use of sponsor funding to ultimately provide far more functionality in a single aligned application.

RDD may be used in a variety of global settings to improve the timeliness of data collection, the efficiency of analyzing results by disease and country, the tracking of disease spread, and the communication of results among different users at different locations. RDD enhances the early detection of disease outbreaks, allowing more time for mitigation of the effects of these outbreaks.

Acknowledgments: We acknowledge the collaboration with and financial support of AFHSC-GEIS and the invaluable assistance from our colleagues, including Commander David L. Blazes (Uniform Services University of the Health Sciences), Matthew Johns (U.S. Department of Health and Human Services Liaison Officer to the U.S. Pacific Command Surgeon), and Jennifer Cockrill (Armed Forces Health Surveillance Center).

References


The Authors

Timothy C. Campbell was the SAGES Technical Lead and Project Manager for the RDD application. Zarna S. Mistry, Gabriel N. Gorelick-Feldman, and Charles J. Hodanics designed and developed features for the RDD application. Steven M. Babin was the subject-matter expert in medicine and public health. Sheri H. Lewis is the Program Manager for Global Health Surveillance. For further information on the work reported here, contact Timothy Campbell. His e-mail address is timothy.campbell@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
A Scalable Data Mining Approach for Providing Public Health with Disease Incidence Predictions Weeks in Advance

Anna L. Buczak, Erhan Guven, Steven M. Babin, Erin N. Hahn, David W. George, Yevgeniy Elbert, Liane C. Ramac-Thomas, Benjamin D. Baugher, Jacqueline S. Coberly, and Sheri H. Lewis

The Johns Hopkins University Applied Physics Laboratory (APL) has developed a novel and scalable data mining and fuzzy association rule-making approach to deriving disease incidence predictions several weeks in advance of an outbreak. This capability provides a new set of information that may be used by decision makers in conjunction with other complementary information about the country (e.g., infrastructure, disease history, agriculture, and U.S. and local military and civilian populations) from a variety of other sources (e.g., intelligence and disease experts). The prediction of the future infectious disease incidence provides the decision maker with enhanced ability to determine whether to enable deployment of measures to increase and focus biosurveillance and/or to plan and enable mitigation efforts to reduce morbidity and mortality well in advance of the start of the outbreak.

INTRODUCTION

Infectious disease outbreaks result from interactions among the host, the pathogen, and the environment, which are components of the epidemiological triad. For many years, study of these outbreaks has focused on using models for the dynamics of disease spread or for outbreak surveillance and detection. The sooner an outbreak is detected, the more timely and effective are the measures that can be deployed by public health agencies to mitigate the morbidity and mortality due to the disease. Recent studies have moved from outbreak detection to the prediction of outbreaks before they occur. Most of these studies rely on varying types of regression analysis, while others use such techniques as neural networks. Because the incidence of many diseases is recognized as being influenced by the environment (e.g., vector-borne diseases), investigators have turned to using environmental data such as temperature and rainfall to make predictions. One advantage of using such environmental variables is that many of them can be obtained by satellite remote sensing, thereby providing the ability to study remote areas and avoid expensive field measurements. Other types of environmental variables include climate indices, such as the Southern Oscillation Index, the West Pacific Index, and the NINO3, which is an eastern Pacific Ocean sea surface temperature anomaly index. Such cli-
Veg tendency is used to enhance the predictive capability because they are known to be leading indicators of future changes in seasonal and nonseasonal weather patterns. Vegetation indices are derived from satellite remote sensing data and provide indications of vegetation types and conditions, soil moisture, and the effects of fires and human land use, all of which may have impacts on disease vector habitats.

It is important to emphasize that the software system described herein substantially differs from systems designed for the early detection of disease outbreaks, such as the Johns Hopkins University Applied Physics Laboratory (APL) Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE). Early-detection systems use a variety of data to detect a disease outbreak that has already begun but is not yet obvious because, for example, there are few cases or the disease is still in its prodromal stage. This article will instead describe a system that makes a prediction of disease incidence several weeks into the future, even before a disease outbreak has begun. When testing their predictive capabilities, most authors of published modeling studies have a tendency to (i) use input data that were already used in model development, (ii) assume all input data are available at the exact time the prediction is made (time T), or (iii) both. Both of these tendencies will lead to exaggerated measures of model performance compared to how the model would be expected to perform in an operational environment. Models developed and tested on the assumption that all the most up-to-date data are available for model input at time T are, in effect, making retroactive predictions compared to how the model would be used in a realistic operational environment. Assume, for example, that the prediction model requires weekly disease incidence input data and the users want a prediction made on Monday, but the data for the previous week will not be available until Friday. Because data are not actually available at the time a prediction is made, the model will either fail to perform at its tested level of accuracy or will not run at all. Because the APL team is focused on the needs of the user (e.g., force health protection and local public health professionals), we take great care in avoiding these two tendencies that our tested model prediction accuracy will more reliably indicate how the user may expect the model to perform. The APL disease prediction system is designed and tested by using data that are actually available to the user at the time the prediction is made, so the resulting prediction accuracy is more realistic for the user. The APL team sought and received input from users ranging from civilian public health departments in other countries to U.S. military public health professionals. There was consensus among these users that a system that can predict disease incidence a month or more in advance would be especially valuable to them both for planning purposes and for implementation of mitigation measures (e.g., ranging from public education efforts such as reducing standing water to spraying insecticide).

This article will further show that using techniques involving fusion of data from disparate sources along with fuzzy association rule mining (FARM) can result in the development of prediction models with promising results for the public health professional responsible for mitigating the effects of disease outbreaks. The project described in this article is called the PRedicting Infectious Disease Scalable Method (PRISM) and was developed for the Joint Program Manager–Information Systems of the DoD. The PRISM software system was built to automate the FARM process prototyped previously by APL for predicting infectious disease and to provide an easily interpreted visualization of the results.

PREDICTION METHOD

The PRISM algorithms and software suite have three tasks to accomplish (i) building a prediction model; (ii) establishing, defining, and maintaining a database, as well as automating input data download, data preprocessing, and prediction generation; and (iii) visualizing prediction output. The first task is the most computationally intensive of the three tasks, but it does not have to be repeated once the prediction model is finalized for a particular disease and geographic region. Running the model to generate a prediction and visualizing the output are not computationally intensive tasks and can be performed on a typical laptop computer.

Building a Prediction Model

Figure 1 provides an overview of the method for building a prediction model and generating predictions. In step 1, subject matter experts and analysts determine what variables might be relevant for a certain disease and location (e.g., dengue fever in a district in Peru) and the data sources for these variables. Most of the time, relevant variables can be determined by reviewing the scientific literature. Once the variables are determined, the sources of data for those variables must be found. Step 2 (Fig. 1) is building the prediction model and testing it. This is first done one time for a particular disease and location to see whether a model can be built that performs with reasonable accuracy. If this is the case, a new model should be retrained about once a year to make certain that the accuracy of prediction remains reasonably high. In building a prediction model, an extended historical database is especially valuable in performing the data mining process. RapidMiner software (http://rapidminer.com/products/), with a large number of extensions developed by APL, is used to implement the FARM methodology and build a classifier based on these historical data. The classifier uses machine-learning techniques to predict class member-
ship for data instances. For example, this classification can be for HIGH or LOW disease incidence, where a threshold between these two classes is based on the scientific literature, opinions of subject matter experts, and the desire of the user to have as low false-positive and false-negative rates as feasible for operational use. While the system so far has used classification to define two classes, the system is not limited to this number.

The classifier uses a set of rules to define the classes. Fuzzy association rules are of the form

\[
\text{IF } (X \text{ is } A) \text{ THEN } (Y \text{ is } B),
\]

where \( X \) and \( Y \) are variables, and \( A \) and \( B \) are membership functions that characterize \( X \) and \( Y \), respectively. \( X \) is called an antecedent and \( Y \) is called a consequent of the fuzzy association rule. The rules are fuzzy because there can be overlapping memberships where the amount of overlap is quantified. As an example, four fuzzy membership functions (SMALL, MED, LARGE, and VERY LARGE) for the variable rainfall are shown in Fig. 2. Fuzzification is defined as the process in which a number (e.g., rainfall value in millimeters) is transformed into a membership value lying between 0 and 1, thereby allowing for a smooth transition between full membership (1) and nonmembership (0). The degree of membership in a set is generally considered to be the extent to which a corresponding fuzzy set applies.

In Fig. 2, a rainfall of 50 mm will be transformed into two membership functions: SMALL, with a degree of membership 0.5; and MED, with a degree of membership 0.5.

Because the model builder typically discovers thousands of such rules based on the historical data, a set of criteria must be used to select the best rules to be used in the final prediction model. These criteria include metrics called confidence, lift, and support. Confidence is the conditional probability that if the antecedents of a rule are true, then the consequent of the rule is also true. A confidence equal to unity means that if the antecedents are true, the consequent is always true. Support is a metric for how general the rule applies. For example, a support equal to 0.01 means that the rule applies to 1% of the data. Lift is another metric, and the higher it is, the more dependent the variables are on one another. More details on these metrics may be found in Agrawal et al. and Buczak et al. The selection of the rules and

![Figure 1. An overview of the PRISM method for creating and using a prediction model.](image)

![Figure 2. Example of fuzzy membership functions (SMALL, MED, LARGE, and VERY LARGE) for the variable rainfall.](image)
A. L. Buczak et al.

JOHNS HOPKINS APL TECHNICAL DIGEST, VOLUME 32, NUMBER 4 (2014)

A viewer based on the Esri ArcGIS application programming interface (API) for the Microsoft Silverlight application (http://www.microsoft.com/silverlight/). The tool uses a catalog service to create a list of currently available results and a results service to display graphically the locations of the predictions color-coded by the classification used for the prediction results (e.g., red for HIGH and green for LOW). In addition to these two Extensible Markup Language (XML) services, Keyhole Markup Language (KML) files saved locally may be browsed and loaded to visualize the results. Further details will be described in the Software Architecture section. An example of the results map generated by this process is shown in Fig. 3.

SOFTWARE ARCHITECTURE

An overview of the PRISM software architecture is shown in Fig. 4. PRISM has two major software components that are used to create disease incidence prediction models and prediction result displays for the user: the model builder and the PRISM web application. The model builder is responsible for creating a classifier that, along with a current data feed, produces the prediction results for a geographic region. The PRISM web application handles the extraction, transformation, and loading (ETL) of external data sources for use in both model building and prediction. In addition, the PRISM web
The model builder is based on the RapidMiner data mining software framework. RapidMiner uses the Java programming language, which provides an easy way to modify its functionality for use with the PRISM system. APL has extended this framework with specific operators that implement the FARM and classifier building algorithms. This model building begins with the selection of the appropriate predictor variables (e.g., rainfall, incidence rates, etc.), which depends heavily on a literature review and the subject matter expertise of the analysts. Historical data that comprise all of the selected predictor variables for the geographic region of interest are collected. As with any data mining technique, the process is more effective with more data. These data are then divided into three disjoint subsets: training data, fine-tuning data, and testing data. The training data are then used in performing the FARM data mining methodology to extract the rules and build the classifiers. The best performing classifier on the fine-tuning data subset becomes the prediction model. The testing data subset, which has not been previously used, is used to measure the accuracy of this model. The model builder process needs to be repeated for every new disease and geographic region. Also, when a new complete year of data becomes available, the model builder process should be repeated so that the training on these new data will help the model maintain reasonable accuracy of its predictions.

The PRISM web application is responsible for scheduling and executing collection and normalization of data from the identified sources, scheduling and running the classifier used to compute a prediction, implementing the web services that display the prediction results, and hosting the standalone viewer (see example of viewer results in Fig. 3). The web application is logically separated into three primary functional areas: the ETL functionality, the prediction generation functionality, and the web services functionality.

1. ETL refers to a process in database usage and especially in data warehousing that involves extracting (downloading) data from outside sources, transforming (or normalizing) the data to fit operational...
needs, and loading the data into the end database or operational store. In the context of PRISM, the raw predictor-variable data are downloaded, transformed to work with the prediction model, and loaded into a geospatially enabled database for easy retrieval by the model. The database can be loaded with static or downloaded files, including large amounts of dynamic data. The ETL also handles missing data errors, including error notifications and repeated attempts at data downloading. The raw data are referenced by jurisdictional division and mapped to geographical resolution. The data are also selected and arranged with respect to both geographic and temporal resolution for exporting to the model.

2. Prediction generation is the process of generating prediction results by using the normalized data and a classifier (prediction model) developed during the model-building phase. The necessary data include data from previous weeks (e.g., T-1, T-2, T-3, T-12) but never from the present week (T) because it always takes a few days for those data to become available. For certain variables (such as Normalized Difference Vegetation Index and Enhanced Vegetation Index) that come in 16-day averages, the most recent week that can be used is T-4 to ensure that the data are really available for downloading. In addition, predictions can be made when some data

Figure 5. Prediction generation architecture.

Figure 6. Web service architecture. The catalog and results services on the left side are in xsd format.
The APL data mining system for developing and using predictive models involves a large amount of disparate types of data (see Fig. 7). Some of these data, such as land elevation, are static. Some data sources are updated only annually and manually, such as population and demographics. The Southern Oscillation Index and sea surface temperature anomaly indices are automatically updated monthly and weekly, respectively. Satellite-derived vegetation indices, land surface temperature, and rainfall are automatically updated every 16 days, every 8 days, and every 3 hours, respectively. Disease incidence data may be automatically updated daily or weekly. Therefore, assembling these disparate data into a database compatible for use by the disease prediction model involves preprocessing into a uniform spatiotemporal resolution.

As mentioned earlier, the data were divided into three disjoint subsets: training data, fine-tuning data, and testing data. Only the first two subsets were used to develop a prediction model, while the third set was used to measure the accuracy of the prediction.

RESULTS

The model is built using only the training data, and it is fine-tuned using the fine-tuning data set mentioned above. Current data are input to the prediction model, and the predictions (4 weeks ahead) can be compared against the testing data set that the model has never seen. For such a comparison to be made, at least two

The PRISM predictions are made accessible through several REST web services. REST web services facilitate transactions between web servers by allowing loose coupling between different services. REST is used because it allows greater flexibility in output format than the Simple Object Access Protocol (SOAP) and, unlike SOAP, it requires less configuration and does not need a message header. The web service architecture is shown in Fig. 6. The PRISM web services list the available prediction model results in both XML Schema Definition format (xsd) and KML. The resulting catalog of available disease/location/model results uses the xsd format to define the required and optional fields and provide the interface to the web service. The KML format is used to allow the catalog results to be displayed in any KML-compliant application, such as Google Earth.

Figure 7. Examples of disparate types of data used in the disease prediction model.
disease states have to be defined. In this case, we defined HIGH incidence as a weekly incidence rate greater than or equal to a fixed number (e.g., 1.5) of standard deviations above the mean historical incidence based on the training and fine-tuning data sets. LOW incidence is anything below that rate. Note that there is no uniform definition of disease outbreak that involves standard deviations above mean historical incidence, and furthermore, the statistical distribution of disease incidence is often not Gaussian. Different diseases exhibit different temporal fluctuations in incidence, and this may also vary by region and population. For the user, the threshold should be operationally meaningful, meaning that it is set at some level above which the user would typically take some kind of action. Depending on where this threshold is set, the accuracy of the prediction can then be evaluated to see when it predicted a HIGH or LOW incidence and how this compared to the actual data.

True positives (TP) are defined as instances where a prediction of HIGH disease incidence corresponded to actual data confirming a HIGH disease incidence. True negatives (TN) are similar except that the prediction of LOW disease incidence corresponds to actual data confirming this result. A false positive (FP) occurs when the prediction says incidence is going to be HIGH but the actual data show it to be LOW. A false negative (FN) occurs when the predicted incidence is LOW, but the actual data show it to be HIGH. To determine the accuracy of these predictions, four commonly used metrics were used:

1. Positive Predictive Value (PPV): \( PPV = \frac{TP}{TP + FP} \) or the proportion of positive predictions that are outbreaks;
2. Negative Predictive Value (NPV): \( NPV = \frac{TN}{TN + FN} \) or the proportion of negative predictions that are non-outbreaks;
3. Sensitivity: \( Sensitivity = \frac{TP}{TP + FN} \) or the proportion of correctly predicted outbreaks (also called Probability of Detection);
4. Specificity: \( Specificity = \frac{TN}{TN + FP} \) or the proportion of correctly predicted non-outbreaks; note that \( 1 – Specificity \) is the False Alarm Rate.

The public health users in Peru wanted a prediction of 4-week dengue incidence for the district of interest made 4 weeks ahead (i.e., 4 weeks from the date the prediction was made), and they wanted a threshold between HIGH and LOW of 2 standard deviations above the multiyear mean\(^{9}\) because this was the level above which they would consider a response. Accordingly, for the Loreto district of Peru, our predictions of 4-week incidence made 4–7 weeks into future resulted in a PPV, NPV, Sensitivity, and Specificity of 0.81, 0.98, 0.64, and 0.99, respectively. For the Philippines, a threshold between HIGH and LOW of 1.5 standard deviations above the multiyear mean was used on the basis of feedback from the users and the quality of the data (e.g., noise, missing data, etc.). The users in the Philippines wanted predictions of weekly incidence made 4 weeks in advance. In this case, for year-2011 predictions 4 weeks in advance of weekly dengue incidence in the Abra province in the Philippines, the PPV, NPV, Sensitivity, and Specificity were 0.75, 0.82, 0.64, and 0.88, respectively.

**LESSONS LEARNED FROM THE TABLETOP EXERCISE**

Over the course of PRISM’s development, end users and stakeholders were familiarized with PRISM through a series of discovery workshops. The purpose of the discovery workshops was to give individuals and organizations tasked with addressing infectious disease outbreaks an opportunity to learn about PRISM, provide feedback, and influence the development of a culminating tabletop exercise (TTX). The purpose of the TTX was to obtain and document more substantial feedback from a broader community of potential end users and to support further development of the model and the creation of a formal concept of employment.

The TTX was held March 5–7, 2013, at Lawrence Livermore National Laboratory in Livermore, California. The event brought together 35 key attendees, including representatives from seven Combatant Commands, the Armed Forces Health Surveillance Center, the National Center for Medical Intelligence, and the Centers for Disease Control and Prevention. The TTX used different scenarios to examine the value to the end users of a disease prediction capability such as PRISM and to better understand customized features the end users recommended to make the tool a useful part of their workflow.

The TTX participants were very enthusiastic about PRISM, and the TTX resulted in several key findings. First, it became clear that an infectious disease predictive capability could provide significant improvement to accomplishing the Combatant Commands’ force health protection mission by conserving time, funding, and resources. Second, when this capability is used in conjunction with other information, it enables Combatant Command planners to choose from a wide range of proactive options to address a potential infectious disease outbreak. Third, resources for infectious disease mitigation and consequence management can be applied more discriminately in a proactive rather than reactive manner. Overall, a predictive capability like PRISM can help improve operational readiness.

The TTX concluded that PRISM should be further developed and should focus, as much as possible, on user-identified features. The TTX identified the desire among users to develop predictive disease incidence models for
all designated infectious diseases of interest in areas of concern, including risks from both natural and man-made outbreaks. While such an objective cannot be achieved in a short period of time, this is a clear demonstration of the end users' enthusiasm for the capability. Some of the other user-desired features included the ability to provide three levels of risk prediction (e.g., red, yellow, and green, instead of only red and green); a broader selection of spatial and temporal granularity; the ability to provide predictions that can be integrated into existing health risk assessment procedures in order to achieve an integrated composite view of the health risk; greater flexibility for models to be easily updated with real-time information (e.g., to account for a socio-economic crisis); and the ability to do “what if” analysis within the model to anticipate the influence of certain variables on the likelihood of a disease outbreak.

The TTX was a capstone event for PRISM, the results of which will be used to guide future development. End user and stakeholder feedback are critical for the model to be used as broadly and effectively as possible. The TTX format was particularly effective because it allowed participants to brainstorm and build on each other’s ideas. The collaboration that took place led to clear recommendations and highly useful feedback for future development of PRISM.

CONCLUSIONS

Among the many challenges faced by public health professionals are the large efforts in planning and allocating resources when disease outbreaks occur. Having a reliable predictive capability can help them to anticipate and prepare more effectively and efficiently for an outbreak. A predictive capability does not replace an early detection capability, which allows for prediction validation and more refined efforts to mitigate the disease outbreak that is already happening. Therefore, prediction and detection systems complement one another. This article describes a prediction capability that is unique as well as novel. It combines advanced techniques in FARM and disparate data fusion to provide a prediction of disease incidence several weeks in advance, even when there is not yet any evidence of an actual outbreak.

The method developed by APL requires one-time model development (CPU intensive) for a specific disease in a specific country or region. This resulting predictive model (not CPU intensive) is then run when requested and when new data are available. New epidemiological data are typically collected and compiled by the public health user weekly so current versions use weekly data. Output of a model run is a country map showing provinces with disease incidence higher than normal or not; this output is provided at least a month in advance and possibly before an outbreak has even begun. PRISM is a tool for prediction and not detection. Detection of an outbreak can occur only when the earliest stages of the outbreak have begun.

The users of a predictive capability for disease incidence have indicated that they prefer results with as low a proportion of false positives and false negatives as possible. Too many false positives can lead to wasted resources and alarm fatigue. Too many false negatives may lead users to wonder whether the predictive capability is adding any value to what they have traditionally done. The method developed by APL reduces both false positives and false negatives to levels deemed acceptable by the public health user, thereby providing added value. According to these users, having at least 4 weeks lead time before a prediction of high disease incidence allows for a wide range of mitigation options. For example, the public health user may choose to begin enhancing ongoing biosurveillance by instigating more frequent data collection and the addition of more data sources, thereby improving the odds for early detection. Alternatively, instead of waiting for early detection to confirm the predicted outbreak, the user may decide that more proactive measures are needed. Depending on the disease, the potentially impacted population, and other factors, these measures may range from less to more aggressive, including such measures as intensified public health educational outreach, using resources to reduce vector habitat, or even employing disease quarantine or personnel evacuation at the earliest signs of positive detection. APL has begun efforts to extend the methodology described in this article to new diseases, including malaria in the Republic of Korea and influenza in the United States.

ACKNOWLEDGMENTS: Funding for this work is provided by the U.S. Department of Defense Joint Program Manager, Medical Countermeasures Systems, JPEO-CBD.

REFERENCES


Anna L. Buczak is a project manager, section supervisor, and Principal Professional Staff data mining specialist. She leads the PRISM team. Erhan Guven is a Senior Professional Staff computer scientist. He developed the majority of PRISM software. Steven M. Babin is a Senior Professional Staff medical doctor and remote sensing specialist. He contributed medical expertise as well as satellite remote sensing and atmospheric science expertise to the analysis and interpretation of the data. Erin N. Hahn is a Senior Professional Staff member. She serves as the Assistant Project Manager for PRISM. David W. George is an Acting Assistant Group Supervisor and a Senior Professional Staff software engineer. He serves as software lead for PRISM. Yevgeniy Elbert is a Senior Professional Staff statistician. He performed preprocessing of the epidemiological data. Liane C. Ramac-Thomas is a Senior Professional Staff Data fusion specialist. She developed the dengue and malaria prediction models. Benjamin D. Baugher is a Senior Professional Staff mathematician. He developed the dengue and malaria prediction models and algorithms. Jacqueline S. Coberly is a project manager, section supervisor, and Senior Professional Staff epidemiologist with a focus in infectious disease epidemiology. Sheri H. Lewis is a Principal Professional Staff public health specialist. She serves as the Program Manager for Global Health Surveillance. For further information on the work reported here, contact Anna Buczak. Her e-mail address is anna.buczak@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
Public Health Applications in the Cloud

Wayne A. Loschen, Miles A. Stewart, and Joseph S. Lombardo

Public health departments use information technology every day to help protect their communities from disease. With the advent of cloud technology, public health professionals are now discussing whether and how cloud technology should be used. This article describes the advantages and disadvantages of putting applications in the cloud, along with lessons learned from moving an existing public health disease surveillance system, Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE), from a locally installed system to a cloud environment. The article includes specific details on the use of the Amazon GovCloud environment, performance issues, and potential issues that could arise from use of the cloud in international settings. Although cloud technologies will not solve every problem, they provide a powerful and flexible foundation for applications—something that public health professionals should consider using for their future applications.

INTRODUCTION

For the past 15 years, the Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE) has been used by public health officials to monitor for outbreaks of disease. A web-based system, ESSENCE has primarily been installed in individual public health jurisdictions’ facilities to support the local users. These jurisdictions may be at the city, county, state, or even multistate level. Each system is configured and deployed so that it scales depending on the needs of the individual jurisdiction.

With constantly reduced budgets, it is important for public health jurisdictions to strive for efficiency with all their expenditures, especially information technology (IT) costs. Some IT costs can be extremely burdensome; jurisdictions in Virginia have been charged more than $200,000 per server for a year of maintenance and San Diego has had costs of $125,000 per server per year. These types of expenses are forcing public health departments to develop strategies to find more efficient ways to use IT systems. They need IT systems that are
reliable, scalable, and cost effective. One strategy is cloud computing.

WHAT IS THE CLOUD?

Cloud computing is a buzzword that is often used and can have different meanings depending on context. The term can often be used to define computing resources that run algorithms across thousands of computers in parallel. It can also be used to define massive storage capabilities that can manage petabytes or exabytes of data. Some people will use the term to describe web-based applications that serve millions of users, such as Gmail or Amazon. Each of these examples illustrates an aspect of cloud computing. Each example entails a number of virtual computing resources that can be created with the click of a mouse or automatically by computer applications.

At its simplest a cloud is just that, a configurable set of virtual computers that can be allocated dynamically. When dealing with clouds, the types of virtual computing resources that can be configured vary from basic computing resources, instances with Windows or Linux as the only installed software, or entire application systems that can be arranged together in a multitude of ways to meet the needs of the user. A system can be designed to take advantage of tens or hundreds of individual cloud computers for a short time to perform a calculation and then release those resources back into the pool to become available to other systems in the cloud. This ability to configure a resource, use it, and then release it allows for increased system flexibility. Instead of purchasing equipment that may be used for only minutes a day and would then sit idle otherwise, organizations can use resources more efficiently, leading to lower costs.

Beyond efficiency, the cloud can provide a flexibility that can change the way a system works. Simple examples include dealing with full hard drives. Almost every system will experience complications as its hard drive becomes full. Frantically, administrators will try to remedy the problem and frequently they will eventually have to purchase bigger hard drives or additional systems. With cloud technology, adding an additional hard drive takes about 30 seconds and costs approximately $0.10 per gigabyte per month,1 much cheaper and quicker than ordering a new or replacement hard drive for a local server and installing it. Additional resources can be added on demand in the form of storage or computational power. Just like additional storage capacity, adding additional servers or migrating to more powerful platforms can be accomplished in the order minutes instead of weeks. This flexibility allows for systems to start small and scale with need rather than requiring large amounts of money initially for storage and computational power that might seldom be needed.

After efficiency and flexibility, the next best aspect of cloud technology is its reliability. Although there have been noticeable exceptions, such as the 36-hour downtime of Amazon’s Elastic Compute Cloud (EC2),2 most clouds are built across multiple physical locations, allowing a system to remain active even if one of its data centers stops working. Losing an entire data center could impact performance, but if the system takes advantage of the redundancy available in the cloud environment, the system can typically remain active. For the majority of administrators who run a single server room in a single building, a cloud is a substantial improvement to reliability.

Unfortunately, there are two factors working against the reliability of clouds. The first is the lack of control if something bad does occur. In the case of a server failure in a non-cloud environment, the administrator can immediately begin putting solutions in place to fix the issue; however, with a cloud, the administrator is at the mercy of the cloud host to fix any issues. (Depending on the local administrator’s level of skill with the particular issue, this actually could be an advantage instead of a detriment.) The second factor is network reliability. Most organizations use two types of networks: internal and Internet. It is possible that an organization could keep its internal network functioning during an outage on the network connecting it to the Internet. When the system is being served to a local user base via the internal network, the loss of Internet connectivity would affect a cloud-based solution more than a locally hosted one. When users require Internet connectivity to reach the server, the reliability of an organization’s network compared with that of the redundant networks of many cloud operators actually favors the cloud-based solutions. Knowledge of the individual situation will help determine the reliability of cloud solutions relative to a locally hosted application for that particular situation.

ISSUES WITH THE CLOUD

Although there are many benefits to using a cloud, cloud technology is by no means a perfect solution and there are concerns that must be addressed. The first concern is security. Hosting sensitive information in servers that sit in someone else’s data center can be a deterrent. Each cloud must help secure an organization’s data from attacks through the network, the underlying hardware, in the physical space in which the data centers live, and from insider attacks from legitimate personnel. Each aspect is different.
For network security, the strategies used for a normal data center are similar to those used in a cloud data center. A cloud-created server connected to the Internet is not much different than a real server connected to the Internet. A cloud provider can add some extra protections, such as firewall tools and anti-spoofing applications, to protect itself and others from network attacks, but these protections can also be incorporated in non-cloud environments. A majority of the network security concerns are at the application and operating system levels. These areas are the responsibility of the system administrators who are using the cloud resources, not the responsibility of the cloud provider. If the application has a vulnerability that allows its data to be accessed, it does not matter whether that system is running on a cloud or in a local data center—the vulnerability is the same.

From the hardware perspective, many cloud providers rely on encryption and software systems to protect and control access to the data they are storing for various organizations. When using a cloud, an organization assumes additional risk compared with having its own physical server. In a cloud environment, a single physical machine may be running virtual servers for multiple organizations. Theoretically, this is less secure than having physical separation between the different systems because software is the only thing separating the shared use of memory and computational resources. There are three factors mitigating this risk. In addition to the expected improvements in the underlying cloud software used to manage security, some cloud providers have begun to allow more dedicated options that create cloud resources that use physical resources completely and do not share those physical resources across organizations. This setup eliminates one potential avenue of attack by preventing individuals outside of one organization from obtaining the data of another organization because access to the physical device is not shared; however, this sort of setup adds costs. The second factor is the creation of specialized clouds, such as Amazon’s GovCloud. These specialized clouds have higher levels of security and require that operators have particular backgrounds, such as being U.S. citizens, before they are given rights to work on the physical hardware the cloud is operating on. The last factor is the threat of revenue loss for the companies selling cloud solutions. Clouds are becoming big business and if they are proven insecure, these companies stand to lose a great deal of money. This threat will help keep the pressure on vendors to keep clouds as secure as possible.

The last aspect is the physical security of the system itself. With all the complexity of network security and the software systems to manage virtual access control, it is still possible for someone to just walk into a server room, remove a hard drive, and walk away with it. This risk may or may not be greater for those organizations with systems in a cloud environment. Certainly in some situations, vendors providing cloud environments have greater security in place to prevent physical theft. However, even if there is increased security, an organization assumes risk when allowing employees of another organization to have physical access to its servers. Unless an organization is already using outside vendors to support a local data center, physical access by non-employees will always be a risk of working with cloud vendors. Vendors can mitigate this risk by requiring certain qualities in their employees, such as Amazon GovCloud’s requirement that employees be U.S. citizens. Similar to the hardware aspect, the threat of lost business because of data theft keeps many vendors vigilant against physical security threats.

In addition to security, the true cost benefits of using a cloud can be difficult to understand. In some situations a cloud solution can be more cost effective than purchasing hardware. This is especially true in dynamic and unknown circumstances that may require different hardware configurations frequently. Examples include web applications that must be able to scale to handle a large event for a short period of time, systems that must be able to frequently increase storage capacities, and architectures that might require researching many different hardware and software configurations. Each of these situations can benefit from a cloud vendor’s pricing system, which allows administrators to pay only for what they use when they use it. Many cloud vendors charge by the hour per computational resource, gigabyte of storage, or gigabyte of network traffic.

If the needs of the system are more known and static, however, using cloud resources may not provide any cost savings. Cloud providers can provide discounted pricing for customers that reserve resources for longer periods of time, such as 1- or 3-year reservations. These discounts, however, still may not result in a cheaper cost for a server when compared to purchasing one directly. However, it can be difficult to determine the true cost of ownership of self-owned servers. It may be hard to understand and compute power, cooling, and manpower costs associated with maintenance and security for a single server, and these costs are specific to a particular organization. In general, cloud resources can be at least comparable and in some cases more cost effective than hosting an in-house system.

**MIGRATING TO A CLOUD ENVIRONMENT**

After considering all of these factors, the Centers for Disease Control and Prevention (CDC) made the choice to use Amazon GovCloud as the cloud platform for its BioSense 2.0 efforts. To provide public health entities with additional analytical tools to use with their data in the cloud, the ESSENCE system was also adapted to work in Amazon GovCloud. The process of migrating an existing system that worked in a non-cloud environment to a cloud environment is ongoing but so far has been successful. Even so, during the process many
lessons have been learned about the Amazon environment, cloud architectures, and possible adaptations to be considered for the future.

Amazon GovCloud is a specialized environment that is based on the Amazon Elastic Compute Cloud, or EC2. Using Amazon Web Services (AWS), GovCloud caters to systems that must meet certain U.S. government requirements in order to be visible on the Internet. From their website: “AWS GovCloud is an isolated AWS Region designed to allow US government agencies and customers to move sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements.”

The first hurdle in using GovCloud was determining what resources were needed. GovCloud offers many different options at many different price points. The ESSENCE system is a Microsoft Windows-based system that typically runs on three to five servers. In the architecture, the servers are constantly on and constantly busy. Therefore, there was no need for on-demand-style servers that could turn on and off as needed. Instead, options that gave discounts for longer-term reservations were more useful. Amazon provides two types of discounts in this regard: an option to pay a smaller up-front cost with a discounted hourly rate or an option to pay a larger up-front cost with no hourly rate. This second option was available only in GovCloud and was not available in the normal EC2 environment. For the needs of our pilot project, the second option for a 1-year reservation was chosen (3-year reservations are also available). It was easy to compute the hourly rate in addition to the up-front cost to determine how many months of constant usage would be required to make the second option more cost effective. This worked out to be close to 8 months depending on the specific configuration chosen.

The next decision was to determine the size required for the computation resource. Amazon GovCloud offers general-purpose instances that have “small,” “medium,” “large,” “extra-large,” and “double-extra-large” options that range from 1.7 to 30 GB of memory and from 1 to 26 EC2 compute units (synonymous to CPUs of a standard size). In addition, other nonstandard instances are available, including micro, memory-optimized, CPU-optimized, storage-optimized, and GPU instances. Inside many of these types are levels of servers including the small-, medium-, and large-style choices. After studying all the options, two standard large Windows servers and two high-memory extra-large Windows servers with Microsoft SQL Server were chosen for the ESSENCE project.

By default, each of these particular servers has a smaller (20 or 30 GB) hard drive that is attached. With two clicks of the mouse, additional storage space can be quickly assigned to each server and attached as a new drive. There are also processes in place to expand the size of existing drives, but adding entirely new drives worked for our purposes. It is also possible to assign Internet-accessible IP addresses, set up firewall rules between the servers and the Internet and between each of the servers, and set up keys and security permissions for management. Before connecting to the new servers, the firewall was configured to open up only certain ports needed; some administrative ports were limited to only specific IP addresses. The next step involved opening up similar ports on the local firewall so that administration could occur. Once all the networking had been configured, it was possible to use a remote desktop connection to begin installing the ESSENCE system in the cloud.

Our belief that the network was fully configured proved false early in the process as we encountered problem after problem with the servers transmitting between themselves and with the Internet. In the end, two culprits were found to cause most of the problems: Windows firewalls on the individual machines and the Amazon mechanisms for configuring firewalls. As with any Windows server, on a cloud or not, certain ports may be blocked by default and new firewall rules must be created to allow that traffic through. What made the problem slightly more complex was the nature of IP addresses inside the cloud. Amazon gives each server an internal IP address to the local network that is created just for individual accounts. In addition, for an additional cost, Internet-accessible IP addresses (called “Elastic” IP addresses) can be enabled as well. Finally, when using the firewall tools for the Amazon network, an administrator can assign rules by security groups. Servers can then be assigned to security groups, and the rules can follow a group instead of individual servers. Because of each of these options, the administrator must take special care to use the correct IP addresses in the Windows firewall relative to what is allowed in the Amazon firewall. If one firewall allows IP connections using the Internet-accessible addresses and the other firewall allows only the internal IP address ranges, traffic will remain blocked.

Eventually all the networking issues were resolved, and the next step in the process involved adding the necessary security and management software to the servers.

Just like a server in a local data center, each server must have patches installed, antivirus software loaded, and monitoring tools enabled. All of the best practices used to maintain a secure platform must be performed. In addition to the security maintenance activities, backup steps must also be developed. These steps can take advantage of cloud capabilities because Amazon offers administrators the ability to take snapshots of currently running systems. These snapshots can be stored and re-created whenever needed. One additional discovery was that all GovCloud servers use GMT time by default, so we had to make sure all backup schedules took this into account.

Finally, with the servers created, the network configured, and the security software installed, the ESSENCE system was loaded onto the cloud servers. This process almost went smoothly, with the only problem being that Microsoft SQL Server was not installed on two of the
images as expected. After correspondence with Amazon GovCloud's representatives, a new image including SQL Server was available. At this point, ESSENCE was installed as if the servers were sitting in a local environment. Finally, we used the local Johns Hopkins University Applied Physics Laboratory (APL) domain name system (DNS) server to create an entry for our cloud web server so that we could give users a URL that was easier to remember.

**PERFORMANCE TESTING**

To evaluate the practical performance of this system in the cloud and to assess the validity of the suggested advantages and disadvantages highlighted in this article, a test was designed to evaluate the number of concurrent users the system could support in the cloud. Using a tool called JMeter and a set of common usage patterns from historic ESSENCE users, a series of workflows were designed that simulated a normal user's behavior in ESSENCE. These behaviors included performing queries, viewing time-series graphs, and then viewing data details pages, for example, among other common usage patterns. The JMeter tool could then be set up to simulate any number of concurrent users attempting to use a website. A test was designed to evaluate four combinations of servers:

- Web server: standard large; and database server: high memory extra-large (XL)
- Web server: standard large; and database server: high memory double extra-large (2XL)
- Web server: high CPU extra-large; and database server: high memory extra-large (XL)
- Web server: high CPU extra-large; and database server: high memory double extra-large (2XL)

The tests showed that the web server machine did not affect performance significantly. The size of the database server, however, did impact performance. Tests were performed with 10, 100, 150, 200, 250, and 300 concurrent users. The results are shown in Table 1 and Figure 1.

Based on these results, the XL servers are comparable to the 2XL servers if the number of concurrent users is 100 or less. This can be seen in similar average times. If the system is required to support more than 100 concurrent users actively using the system, then the more powerful server should be used. In our experience with ESSENCE usage in the past, we estimate that systems that have more than 200 registered users rarely have more than 20 concurrent users. This would indicate that, depending on their activity level, at least 1000 users could use the XL database server without a performance hit.

This performance testing proved that multiple jurisdictions could share cloud resources and still expect the level of performance that they are accustomed to in a local version of ESSENCE. This sharing of resources could decrease the costs for individual jurisdictions and allow for an easier path for scalability and flexibility for

<table>
<thead>
<tr>
<th>No. of Users</th>
<th>Min. Time (ms)</th>
<th>Max. Time (ms)</th>
<th>Avg. Time (ms)</th>
<th>Success Rate</th>
<th>No. of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XL Database Server</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>336</td>
<td>4,112</td>
<td>680</td>
<td>100</td>
<td>398</td>
</tr>
<tr>
<td>100</td>
<td>47</td>
<td>47,747</td>
<td>2,891</td>
<td>100</td>
<td>2,978</td>
</tr>
<tr>
<td>150</td>
<td>1,283</td>
<td>69,672</td>
<td>19,613</td>
<td>100</td>
<td>1,739</td>
</tr>
<tr>
<td>200</td>
<td>896</td>
<td>104,924</td>
<td>31,119</td>
<td>100</td>
<td>1,652</td>
</tr>
<tr>
<td>250</td>
<td>528</td>
<td>99,719</td>
<td>35,893</td>
<td>99.68</td>
<td>1,864</td>
</tr>
<tr>
<td>300</td>
<td>992</td>
<td>93,779</td>
<td>29,464</td>
<td>74.92</td>
<td>2,624</td>
</tr>
<tr>
<td><strong>2XL Database Server</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>42</td>
<td>6,175</td>
<td>467</td>
<td>100</td>
<td>367</td>
</tr>
<tr>
<td>100</td>
<td>42</td>
<td>19,513</td>
<td>2,387</td>
<td>100</td>
<td>3,150</td>
</tr>
<tr>
<td>150</td>
<td>43</td>
<td>22,221</td>
<td>1,909</td>
<td>100</td>
<td>4,930</td>
</tr>
<tr>
<td>200</td>
<td>50</td>
<td>72,870</td>
<td>7,128</td>
<td>100</td>
<td>4,248</td>
</tr>
<tr>
<td>250</td>
<td>44</td>
<td>147,909</td>
<td>14,270</td>
<td>100</td>
<td>3,596</td>
</tr>
<tr>
<td>300</td>
<td>292</td>
<td>171,364</td>
<td>26,562</td>
<td>76.09</td>
<td>2,769</td>
</tr>
</tbody>
</table>

![Figure 1. Performance results of ESSENCE usage based on XL- and 2XL-sized database servers.](chart.png)
the participating jurisdictions. Anecdotally, users of the cloud version of ESSENCE also found the day-to-day performance to be faster than it was in their local instances. So in addition to the efficiency, perceptions of performance may even be improved in a cloud instance over a local instance.

**INTERNATIONAL USE CONSIDERATIONS**

Although the CDC and other U.S. public health entities have begun using cloud technology, there may be additional barriers to its use in some international settings. Although there are many cloud services available throughout the world, some countries do not have cloud providers located inside their borders. This is not a technical barrier as long as the public health entity has an Internet connection, but it can be a political barrier. The issues of data privacy are serious and some jurisdictions have policies that forbid data from leaving their countries. In addition to the political issues, there are technical concerns if the public health entity does not have reliable or robust Internet connectivity. Although the cloud services may be available, if the public health entity cannot connect to the service to provide or analyze its data, the value of using a cloud is greatly diminished. Even if the entity has reliable Internet connectivity, if there is not enough bandwidth to allow the data to be transferred in a timely manner from the data providers to the cloud application, this too can render a cloud-based solution unusable. In these situations, preference may be given to local solutions that can incorporate data collection from a variety of methods, including hand-entering or loading collected data from media or text messaging-based collection systems. These collection methods can bypass use of a network and allow analysis to be performed despite the IT limitations. Although many public health entities in an international setting may be perfectly suited to use cloud technology, it is not something that should be assumed without first considering these possible barriers.

**CONCLUSIONS**

Although cloud technology is not a perfect solution to every problem, the ability to have a scalable and flexible foundation for applications is useful for tools such as ESSENCE. The cloud platform provides the ability to host a large number of users on a single hardware platform and therefore share costs across multiple jurisdictions, providing a more cost-efficient model for IT system hosting. It also has the added benefit of being able to adapt in the future by assigning additional resources to the application if needed. Both of these aspects allow for a cloud version of ESSENCE to be a much more efficient solution compared with numerous individual instances. Although technology and pricing are not the only issues to consider when deciding on the utility of a cloud-based application, we have shown through this pilot and our performance testing that a cloud-based system could be a valuable alternative that should be considered.

**ACKNOWLEDGMENTS:** The ESSENCE in the cloud initiative was supported by the CDC's Division of Notifiable Diseases and Healthcare Information (DNDHI) BioSense Program. Also, the project could not have succeeded without the hard work of many individuals from all three partner organizations for this project: CDC's BioSense team, Tarrant County Public Health Department, and APL's ESSENCE team.

**REFERENCES**


The Authors

Wayne A. Loschen is the Technical Lead for ESSENCE-related projects. He is an APL Senior Professional Staff member in the Bio-Threat Awareness Systems (QAI) Group. For ESSENCE projects, he provides technical guidance and software development and works with local public health partners to improve the utility of ESSENCE for its users. Miles A. Stewart is a software engineer in the QAI Group. He is responsible for back-end development, performance testing, and general feature development for the ESSENCE project. Joseph S. Lombardo is the Principal Investigator (PI) for ESSENCE and has led the program since its inception in 1997. For further information on the work reported here, contact Joseph Lombardo. His e-mail address is joe.lombardo@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.
Much of the work of the Global Health Surveillance program is focused on building the capacity of partner countries to implement and sustain electronic disease surveillance capabilities. Many of our partner countries face challenges such as routine flooding, lack of access to running water, and generally austere conditions. Technology that is introduced must be scaled appropriately to work within the target environment.