EpiViewer: An Epidemiological Application For Exploring Time Series Data

Swapna Thorve

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
in
Computer Science and Applications

Madhav V Marathe, Chair
Anil Kumar S Vullikanti
Achla Marathe

27 November, 2018
Blacksburg, Virginia

Keywords: Web services, model-view-controller, data visualization, epidemiology, metrics, charts, cataloguing, diverse datasets

EpiViewer: An Epidemiological Application For Exploring Time Series Data

Swapna Thorve

(ABSTRACT)

Visualization plays an important role in epidemic time series analysis and forecasting. Viewing time series data plotted on a graph can help researchers identify anomalies and unexpected trends that could be overlooked if the data were reviewed in tabular form. However, there are challenges in reviewing data sets from multiple data sources – data can be aggregated in different ways and measure different criteria which can make a direct comparison between time series difficult. In the face of an emerging epidemic, the ability to visualize time series from various sources and organizations and to reconcile these datasets based on different criteria could be key in developing accurate forecasts and identifying effective interventions. Many tools have been developed for visualizing temporal data; however, none yet supports all the functionality needed for easy collaborative visualization and analysis of epidemic data. In this thesis, we develop EpiViewer, a time series exploration dashboard where users can upload epidemiological time series data from a variety of sources and compare, organize, and track how data evolves as an epidemic progresses. EpiViewer provides an easy-to-use web interface for visualizing temporal datasets either as line charts or bar charts. The application provides enhanced features for visual analysis, such as hierarchical categorization, zooming, and filtering, to enable detailed inspection and comparison of multiple time series on a single canvas. Finally, EpiViewer provides a built-in statistical Epi-features module to help users interpret the epidemiological curves.

EpiViewer: An Epidemiological Application For Exploring Time Series Data

Swapna Thorve

(GENERAL AUDIENCE ABSTRACT)

We present EpiViewer, a time series exploration dashboard where users can upload epidemiological time series data from a variety of sources and compare, organize, and track how data evolves as an epidemic progresses. EpiViewer is a single page web application that provides a framework for exploring, comparing, and organizing temporal datasets. It offers a variety of features for convenient filtering and analysis of epicurves based on meta-attribute tagging. EpiViewer also provides a platform for sharing data between groups for better comparison and analysis.

To my mother - Minakshi and my younger brother - Sanket, who have always been a great source of happiness in my life.

Acknowledgments

I express my gratitude to many people who have played an important role in making this journey a wonderful experience. First, I thank Dr.Madhav Marathe for constantly challenging me and guiding me. A special thanks to Mandy Wilson for her patient guidance, enthusiastic encouragement and useful critiques of this research work. I am grateful to have worked with Dr.Bryan Lewis, Dr.Samarth Swarup, and Dr.Anil Vullikanti for this project. I also thank Dr.Achla Marathe for her valuable suggestions to the thesis. This work has been funded by the following sponsors: DTRA Contract HDTRA1-11-D-0016-0001 (CNIMS), DTRA Contract HDTRA1-11-D-0016-0005 (Biosurveillance Ecosystem -BSVE), DTRA Contract HDTRA1-17-D-0023, HDTRA117F0118 (Technical Reachback CNIMS), and Virginia Tech Internal Funds. And finally, last but by no means least, to the NDSSL family - it was great spending time with all of you during last two years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the face of an emerging epidemic, like the Ebola outbreak in West Africa in 2014 or the Zika outbreak in Brazil in 2017, authorities often turn to epidemiologists to help determine the likely severity of the outbreak and to identify strategies to curtail the spread of the disease. Epidemiologists have a number of approaches they can use to assess the situation, including reviewing historical outbreaks and strategies that have been tried in the past; however, visualization of different kinds of spatiotemporal datasets are key in interpreting the scope of the outbreak [31].

However, sometimes the review of time series data is not straightforward. During the 2014 Ebola crisis, for example, epidemiologists from many organizations were tasked with identifying measures likely to be effective in stopping the spread [28, 34]; this required a good understanding of the spread and prevalence of the infection, as well as the likely progression if left unchecked. There were a number of sources for surveillance data, including local gov-

ernment tallies and statistics provided by the World Health Organization (WHO) [1, 7, 17]. Meanwhile, several public health agencies and university laboratories offered forecasts of how Ebola was likely to progress in those regions, including the Centers of Disease Control and Prevention (CDC), Columbia University, the Laboratory for the Modeling of Biological Socio-technical Systems (MoBS Lab), and the Network Dynamics and Simulation Science Laboratory (NDSSL); these forecasters also released frequent updates to these datasets as new surveillance data surfaced, in order to provide policymakers with the most current information [1, 5, 10, 27].

1.2 Motivation

During an international crisis such as Ebola Outbreak of 2014, numerous experts from diverse disciplines all around the world were working in a collaborative fashion so that they can devise a plan for curbing the epidemic as soon as possible. Data measuring various criteria such as (deaths, case counts, hospitalizations) were collected from the ground as well as forecasts were generated and released on various online platforms for public usage or kept in private repositories.

As the researchers attempted to evaluate these datasets, they found that discrepancies in the data, aggregation type, data formats, category, and scope made it difficult to tell a cohesive story from the various datasets. Some of these problems were rooted in how the data was collected, including incomplete or overestimated reporting of the surveillance data, as well as different modeling methods for the forecasts [1, 30]. More fundamentally, however, there were inconsistencies in how the time series were reported (i.e., as incidence or cumulative counts), differences in the criteria measured (cases, hospitalizations, or deaths), varied reporting dates and frequencies, as well as differences across regions [30]. As the outbreak progressed, the

researchers realized that there was no central repository to catalogue these useful data. The sheer number of datasets to evaluate was an additional complication, especially because the datasets were often published in incompatible formats (such as Excel vs. PDF); this made it difficult to compare trends across datasets or to identify outliers or unreliable time series. A number of tools (Excel, R, and SAS) are used by epidemiologists to address these issues, however, they do not solve the fundamental issue of *standardizing formats* and allowing *open access* to these data. Additionally, as the ad-hoc team responding to this crisis was international (there was the problem that collaborators were not always co-located, so being able to share data easily across locations was also a factor), a persistent, standardized, and open way of *visualizing and sharing these data* was needed.

All these gaps and challenges were realized by researchers once the outbreak started. These challenges were identified as crucial since they hindered the process of quick evaluation of the current epidemic situation. The lack of a collaborative platform for sharing, visualizing, and cataloguing these time series in a standard format was realized. To address these challenges, the concept of a tool such as EpiViewer was developed. The landing page of the application is shown in Figure 1.1. To overcome the lack of quick visual aids for understanding the current state of the outbreaks and to address the incoherent nature of the datasets, an easy to understand user interface is developed to explore temporal data with a minimum learning curve.

There are 3 parts of the application (Figure 1.1): Filters, User actions, Canvas area. The filters are present on the top row and right hand side. These are dynamically populated depending upon the data in the system. The bottom row facilitates other user actions such as data sharing, zooming, playing a movie and uploading/cataloguing datasets and controlling visibility of these groupings. Users can easily load time series data from disparate data sources into the application via the interface or services. The temporal data (may include

margins of error) can then be displayed using a variety of different visualization options, including incidence vs. cumulative displays and the ability to use dual Y-axes to compare graphs of differing orders of magnitude. The line charts portray the temporal effect and the bar charts are used to evaluate the cumulative effect. Users assign metadata attributes to their time series, which EpiViewer, in turn, leverages to provide advanced filtering capabilities to limit which time series are visible on the canvas at a given time. Time series can also be organized into workspaces, called *Views*, to allow users to group data in meaningful ways, such as separating epidemic data by year or separate archiving of different disease related data. These workspaces can be shared with other users by using visibility controls, downloading datasets and creating snapshots. These three options facilitate collaboration between researchers. The application also provides a table of the statistical values of some important epidemic features such as total count, peak time, peak value, first take-off time and value for each time series.

1.3 Contributions

Major contributions to the system are outlined as below:

1. EpiViewer is an *open source application* that provides an easy-to-use interface requiring no special skill-set for managing visualizations. It is a simple yet robust application that fills a very particular set of requirements in the field of epidemiology. The application facilitates *sharing data* among researchers and collaborators in a *standardized format* and allows *open access to the data*. The user study shows that the application has a *quick learning curve* irrespective of prior knowledge of epidemiology. The application is suitable for all types of users novice to domain experts.

2. The motivation for developing the system has significantly influenced the design choices for

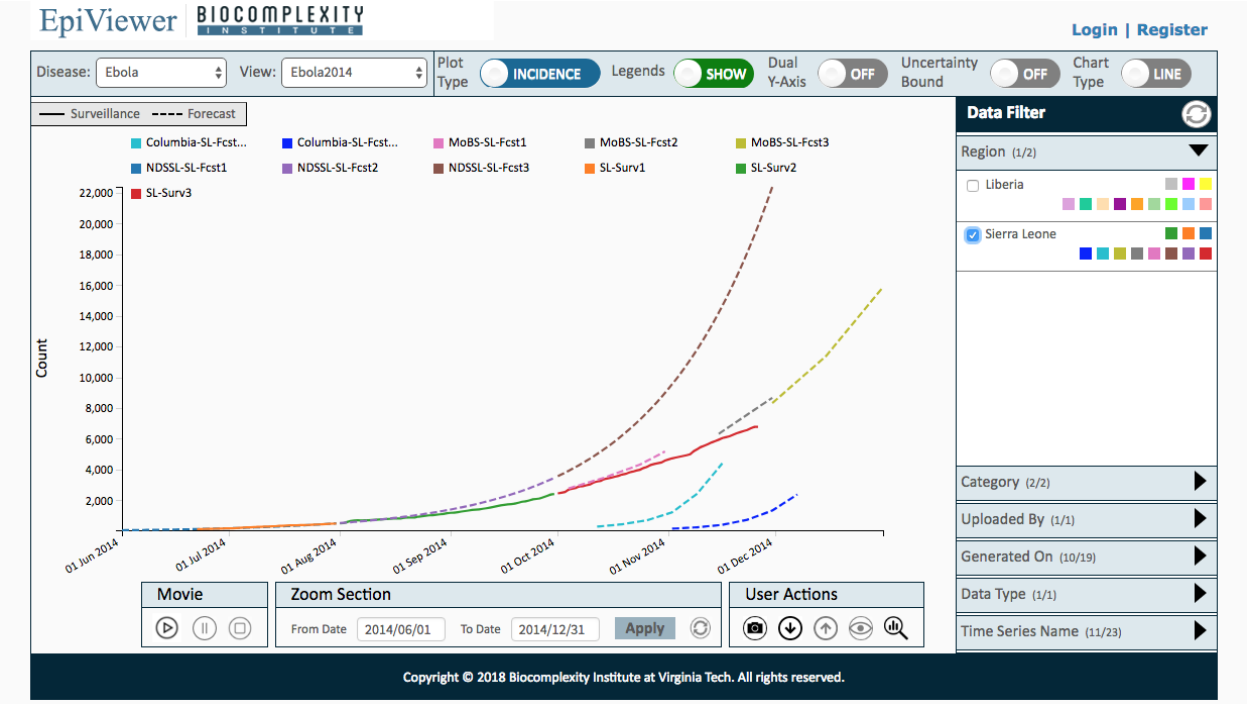

Figure 1.1: 2014 Ebola outbreak graphs from Sierra Leone. EpiViewer was originally developed to help epidemiologists review time series data for the 2014 Ebola outbreak. The forecasts generated by the MoBS laboratory (pink, grey, and olive green) and some generated by NDSSL (blue and purple) were ultimately found to be close to the actual ground truth data (solid orange, green, and red).

building the system. It was extremely important to dsign a lightweight app yet had a reusable and extensible architecture. The system employs a *web service-oriented* architecture. This type of architecture provides the flexibility of adding such a webapp as an integrated plug-in application to larger web-based systems. An example of this is the *integration of EpiViewer into the Biosurveillance Ecosystem (BSVE)*, a large-scale analytics platform funded by the Defense Threat Reduction Agency (DTRA) for the analysis, visualization, and curation of real-time global epidemic and outbreak data.

3. The app provides controlled sharing of datasets in standard formats (JSON, CSV format). Data can be shared in 3 ways: canvas snapshot (image), data files (csv format), workspace sharing (public, private). Data can be collated from disparate sources, stored and viewed in

the same workspace. Users can import these time series in their own collections.

4. EpiViewer provides a statistical module called as *Epi-features*. They are statistical characteristics of an outbreak curve that can help researchers interpret the quality of the epidemic curves, and to identify outlier time series [36]. The module implements 5 such features. These are useful for the expert users of the system.

In summary, EpiViewer fills a particular niche in the space of tools for visualization and exploration of epidemiological data, complementing the available spatial visualization tools with a tool focused on time series data.

The presented thesis has been a part of the CNIMS and BSVE project grants mentioned in the Acknowledgement section. The work reported in the thesis has been published as a full paper in BMC Bioinformatics Journal in Nov 2018 [37]. It was also presented as a poster in CBDS&T conference in Nov 2017. Multiple faculty were involved in the project at different phases. Below are the detailed contributions of various people in the project. The central idea of the project was established by Bryan Lewis, Madhav Marathe, Mandy Wislon and Keith Bisset. The user study was suggested by Samarth Swarup. Mandy Wilson was responsible for the IRB and user task list for the study. Mandy Wilson and Swapna Thorve conducted the user study. Concept of workspaces was realized by Mandy Wilson, Swapna Thorve and Bryan Lewis and implemented by Persistent Systems Limited. The concept of epi-features was suggested by the domain expert Bryan Lewis and implemented by Swapna Thorve. Anil Vullikanti, Mandy Wilson, and Madhav Marathe provided overall guidance while writing the EpiViewer paper.

Contributions of Swapna Thorve:

(i) Designed the standalone instance alongwith Persisten Systems. (ii) Designed the system architecture for extension to the BSVE platform.

(iii) Balancing the user interface design for EpiViewer between BSVE and standalone in-

stance.

(iv) Understanding various aspects of the BSVE ecosystem: architecture of the platform, user management system, deployment, etc,.

(v) Importing data sources from the BSVE system and external sources into EpiViewer automatically via user interface and externally.

(vi) Developing the web services platform for both instances.

(vii) Integrating EpiViewer as a whole into BSVE.

(viii) Conducting and analyzing the user study results.

(ix) Implementation of epi-features.

(x) Major contributions (first author) to the BMC paper as well as the poster.

Chapter 2

Literature Review

The history of epidemiological visualization goes back to the 1800s when John Snow first plotted cholera cases on a map of London [24]. Since then, graphs and visualizations have played important roles in epidemiology by supporting communication, numerical analysis, and the use of data for hypothesis testing and decision making. These days, charts, maps and analytical reports make tremendous use of visualization tools to supplement individual-level clinical data and population-level statistics, which help epidemiologists in various disease analyses [8, 9, 11, 12, 23]. A variety of tools have been developed to help public health professionals analyze and visualize the complex data used in understanding diseases and their effects on the population. To better comprehend the information needs of public health officials and epidemiologists, it is necessary to consider several social aspects of the population and environment, as well as the nature of epidemic propagation. Due to the challenges of integrating, analyzing, and displaying public health data, visualizations tend to lay special emphasis on the spatial and temporal aspects of the disease incidence [15]. While some recent tools have focused on the spatial aspect of epidemiological data visualization [20, 35], there aren't comparably easy-to-use tools for temporal data.

Recently, visualization motifs, such as progression of epidemic curves over time, choropleth maps, social network graphs, phylogenetic trees, and multiple-views involving different aspects of the data (spatial, temporal) and chart types (line chart, bar chart, heat maps, filters), have increasingly been used to characterize disease outbreaks [8, 9, 11, 12]. Visualizing such complex and diverse datasets have unique requirements and properties that can be used to enhance their effectiveness. For example, public health practitioners and researchers are faced with integrating diverse data types, such as mortality data, clinical data, geographical data, patient and pathogen genetics, and epidemic timelines. Each of these types can be recorded, stored, and displayed in many different formats. When data tends to accumulate over time, visualization tools can improve comprehension of this data.

EpiViewer is specific to time series data. There are numerous applications that cater to time series plotting needs; some popular examples are Excel, SAS, SPSS Statistics, and R. Although Excel has a quick learning curve and many useful statistical functions, it can be limiting when more advanced data manipulation, filtering and graphics are needed. Other popular frameworks require the user to have a background in computer programming and statistics in order to develop visualizations. For example, there are five SAS procedures available for plotting time series data; one of these, the 'PROC SQL', requires the user to know Structured Query Language (SQL). Users need a good understanding of these procedures in order to select the right one for visualization of their time series. Also, SAS is not an open source application. R is used widely by statisticians, but is a programming language that requires a background in computer scripting. SPSS is similar to Excel for plotting simple graphs and conducting simple analysis, but also like SAS, it is not open source. Analysis of temporal epidemic data involves estimation of different metrics, some of which (e.g., the first take-off time) involve a specific definition, and have to be estimated from the temporal data. Additionally, there are many different and often inconsistent metrics for comparing

epidemic time series and forecasts [36]. EpiViewer is an open source application that provides an easy-to-use interface requiring no special skill-set as a prerequisite for managing visualizations.

The tools above compared to EpiViewer provide temporal visualizations, but spatial visualization tools are also useful to epidemiologists. One popular spatial visualization is the heat map, which plots the distribution of an epidemic over a geographical space; this allows researchers to see where the hot spots are, as well as the spread and prevalence of the disease. One application that supports spatial visualization is DotMapper [35] which uses dot maps to provide geographic displays of disease incidence in a user-friendly way. VoroGraph [20] employs Centroidal Voronoi Tessellation and Border-Encoded Maps to show local commuting and long-distance flight relationships to help illustrate how diseases can spread.

There are a few other examples present in the literature that combine different visualization motifs, but aren't focused on time series visualization and manipulation. The Pandemic Influenza visualization tool [25] presents an interactive approach where the visualization modeling methodology makes complex modeling and simulation tools easily accessible to public health officials and decision makers for their own use. The combination of heat maps and timelines in this application makes it possible to look up detailed information quickly. The HealthMap project [3, 13] presents data acquired weekly from a variety of freely available electronic media sources to provide a comprehensive view of the current global state of infectious diseases. This data can be visualized on maps, and additional information is displayed via tool tips and alerts. However, not all the tools support easy integration, filtering, and analysis of multiple temporal datasets and forecasts, and comparison of performance metrics across the datasets.

Chapter 3

System Design

This chapter outlines the the system in a formal nature through various architecture specifications, web service descriptions, instance deployments, statecharts and sequences of various workflows. A separate section describes important methods in the application.

3.1 Architecture

The system architecture of the application is made up of three components: the presentation tier, business tier, and data tier, as shown in Figure 3.1.

Presentation Tier: EpiViewer is a Single Page Application (SPA) implemented using a model-view-controller (MVC) architecture. An SPA is a web application that loads a single HTML page that is dynamically updated as the user interacts with the application. The presentation tier is implemented using HTML5, CSS3, Javascript, JQuery, AJAX, and D3. The controller mostly processes the requests internally and sometimes may use AJAX to communicate with the REST API Layer in the Business Tier, allowing parts of the page to

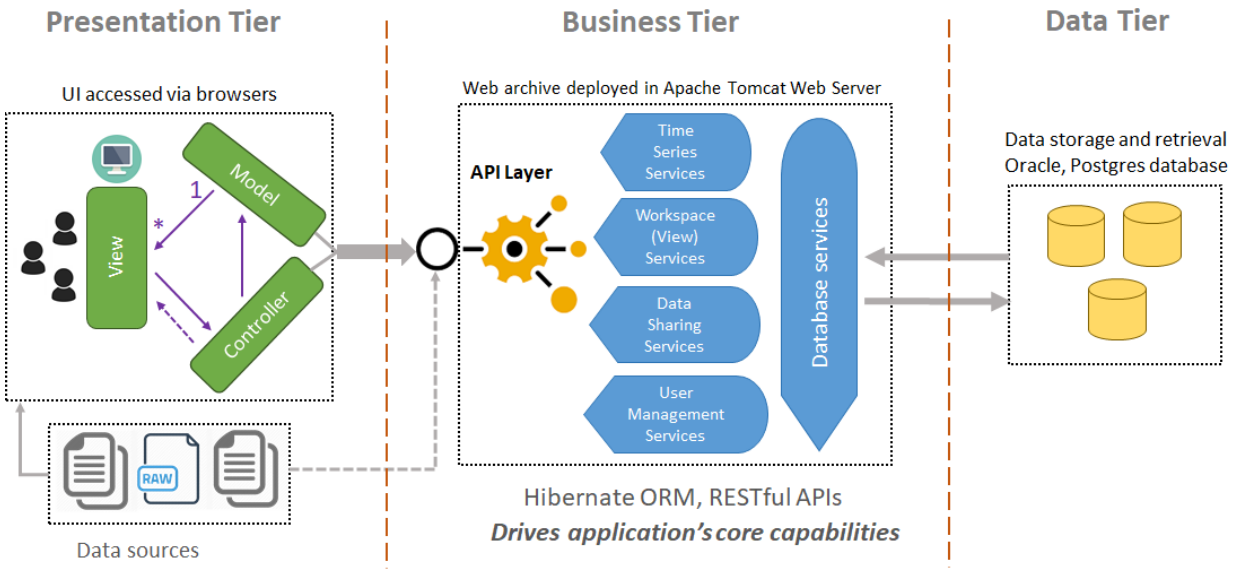

Figure 3.1: Architecture of EpiViewer. EpiViewer is developed as a three-tiered architecture. The Presentation Tier includes the user interface and application functionalities, and communicates with the business tier via an API layer. The Business Tier contains the core logic of all major application functionalities (e.g. upload data, data sharing). The Data Tier consists mainly of the relational database storage. Data can be loaded into the system via the user interface or externally via services from the API layer.

be refreshed without the overhead of reloading the entire page. Data can be loaded into the system through the presentation tier/browser or via REST API into the system.

The MVC architecture plays an important role in *separation of duties* for the Presentation Tier. Refer Figure 3.2 for details. The model denotes entity and data of the system (e.g. workspace, user profile). The model does not store/process any business logic. The view renders user interface templates. The controller is the binding glue of the MVC architecture. It processes event requests and communicates between model and view. A model can be related to multiple views to render data in different contexts or tasks (e.g. Graph data can be used to upload new graph view as well as edit graph view).

When the user interacts with the system via the browser, the view passes the event request to the controller. The controller processes the request and assembles the model as required by

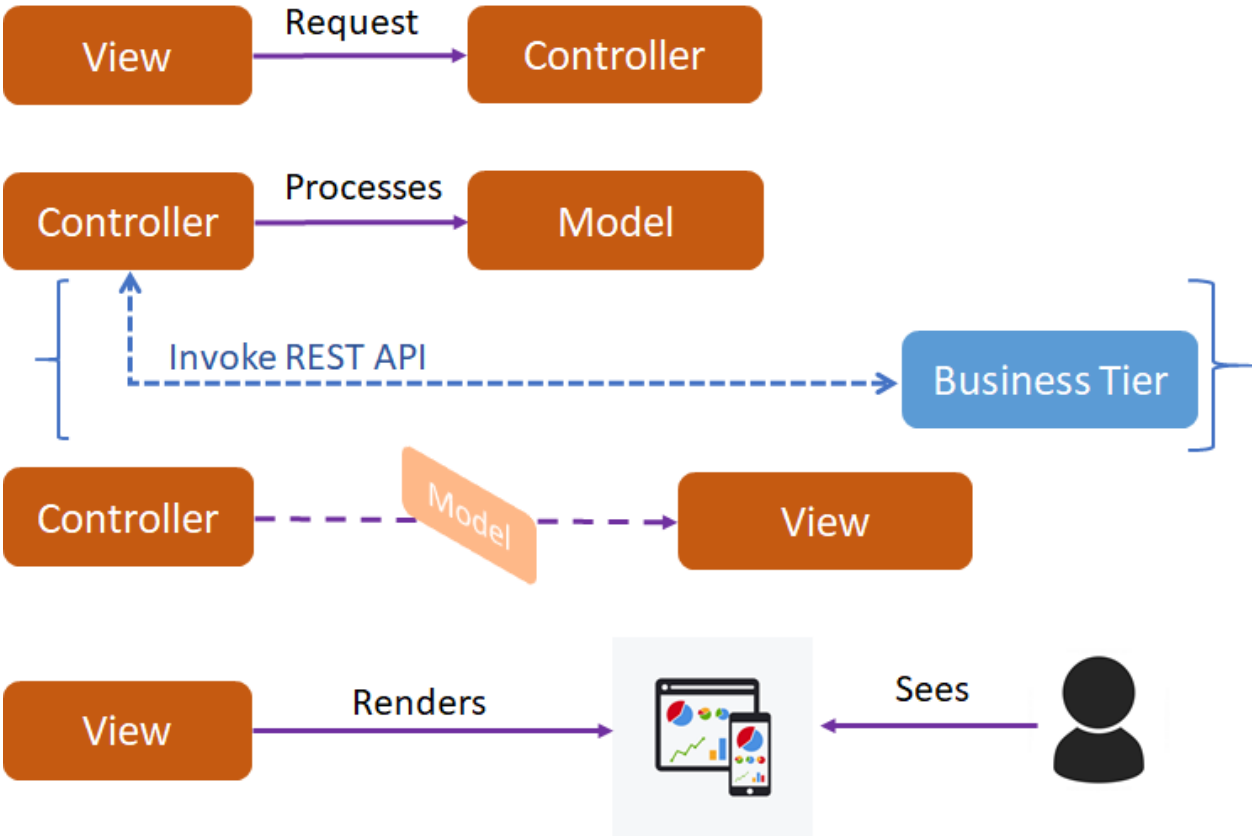

Figure 3.2: The MVC architecture plays an important role in *separation of duties* for the Presentation Tier. The model denotes entity and data of the system (e.g. workspace, user profile). The view renders user interface templates. The controller is the binding glue of the MVC architecture. It processes event requests and communicates between model and view.

the request. The controller also decides the view to be displayed in response of the triggered event. The view then converts model data into the UI template. Figure 3.1 shows that the Presentation Tier and Business Tier communicate via REST APIs. Such a workflow is initiated only when the controller does not have the data and business logic to process system functions. The blue workflow in Figure 3.2 is then triggered. An example of such functionality is when a user logs in and has to be authenticated OR the user uploads a new graph into the system. In these situations, the controller first invokes the API and waits for the API response from Business Tier for that request. Once the response arrives, controller proceeds with its default workflow of assembling model and invoking view. The controller

processes all events on first come first serve basis. All response from Business Tier arrives in the form of HTTP objects and JSON objects.

Business Tier: The business tier is the layer that drives the application's core capabilities. The business tier supports service-oriented computing by using Representational State Transfer (REST) APIs for data transfer. This request-response architectural style involves communication with a specific application service by sending all requests for that service to a specified endpoint. These endpoints consider data and functionality as resources and are accessed using Uniform Resource Identifiers (URIs), typically implemented as web links [6]. Imagine these communications as message passing tasks using standard HTTP protocol. The APIs may perform a single functionality or combine multiple business logic processes if required for a requested event. Refer Figure 3.3 for the details. We use the Jersey RESTful Web Services framework [4], an open source framework for developing RESTful Web Services in Java. Entity management and the database service layer are managed using the Hibernate Java framework. Data formatted in JavaScript Object Notation (JSON) is used for communication between the tiers.

Data Tier: EpiViewer uses a relational database for data storage. The application currently supports both PostgreSQL and Oracle. The conceptual schema diagram for the entities is shown in Figure 3.4.

EpiViewer can be viewed as a web-service application using a combination of resource-oriented and service-oriented architectural styles. This architecture style facilitates reusability and ease of interconnection with other systems. This style supports the interoperability of services by abstracting service details from the end-user application. This facilitates and increases vendor diversity options. An example of this is the integration of EpiViewer into the BSVE analytical framework.

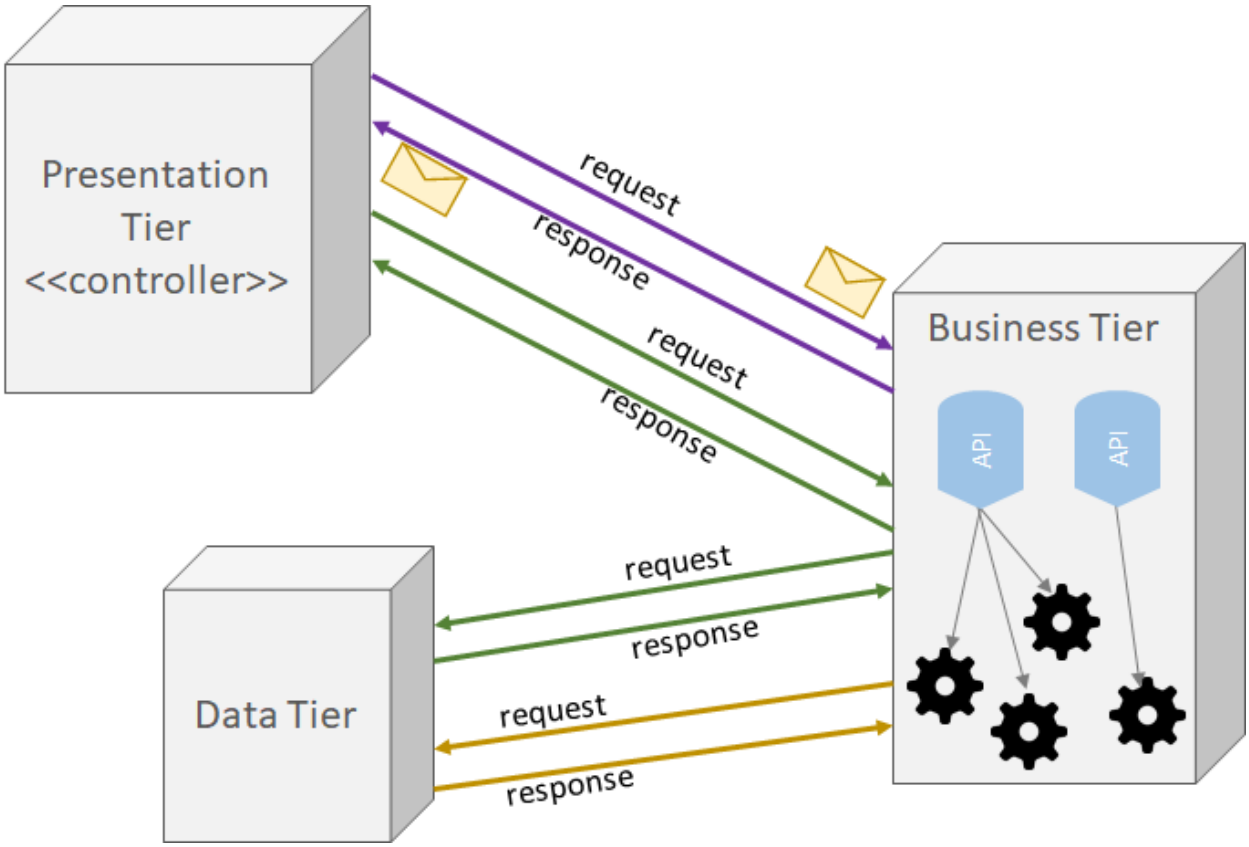

Figure 3.3: The business tier is mainly operated by services and core task logics. The layer supports interoperability, extensibility, deployability, reusability, loose coupling, complex event processing, and operate on standardized protocol and provide service composition.

Technology Stack

The Presentation Tier uses Javacsript, D3.js, Underscore.js, Ajax, Backbone.js, HTML, CSS. The Business Tier uses Hibernate, Java, JSON, Java Servlet, Jersey REST service implementation. The Data Tier uses Oracle and Postgres.

3.2 Statechart

We will use the statechart to understand the software components of the system. A finite state machine can be treated as a statechart in software terms that can be defined by list of

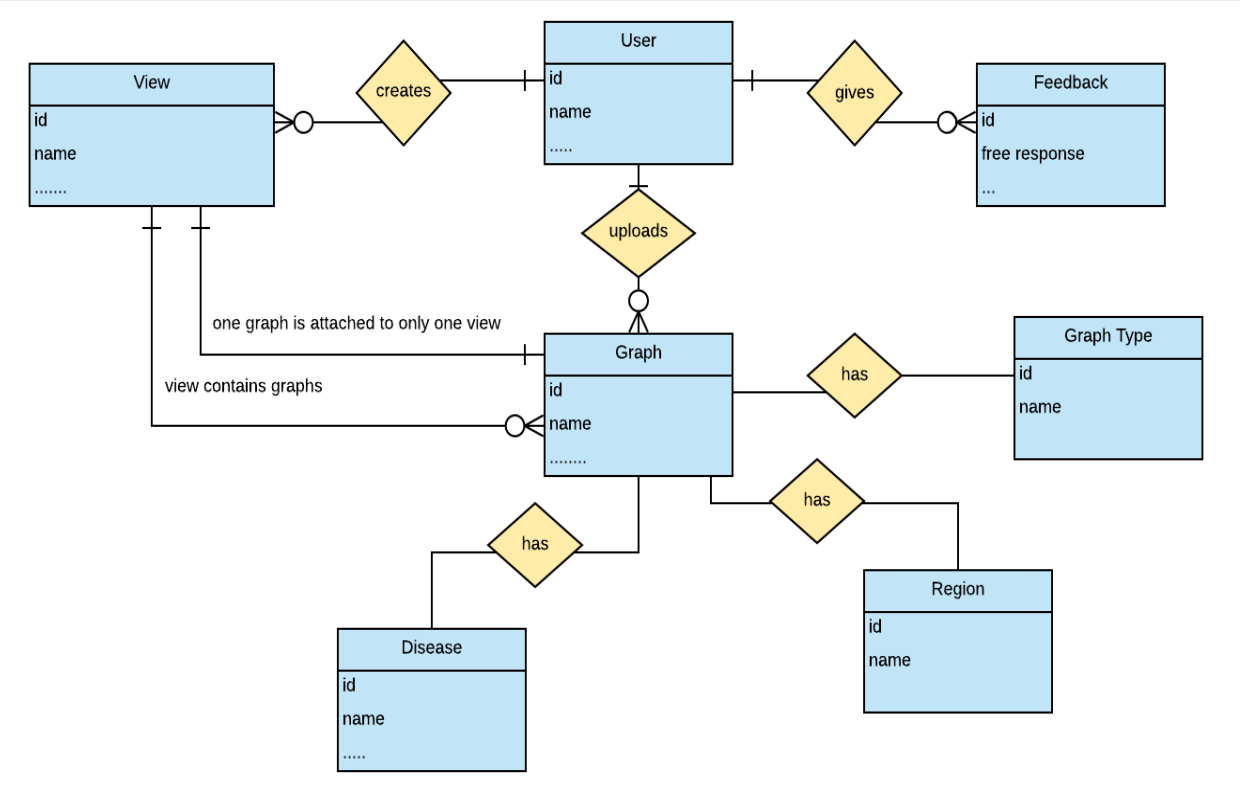

Figure 3.4: Conceptual Schema of EpiViewer. The diagram depicts the operational and transactional entities of the system. The conceptual schema is useful to show the overall scope of the model and portraying the system in a conceptual manner through interactions of entities.

events/states, how an object moves within various stages of its lifetime. Statecharts are a good way to depict event-driven objects in a reactive system.

Figure 3.5 highlights the overall states and events of the application during the course of its lifetime. It shows the possible transitions between states ad events that initiate these transitions. Two types of states exist: Composite, Simple. A composite state denotes that there are additional steps/workflows within this state. All composite states have active involvement of Presentation Layer and Business Layer. Green composite states can co-exist with other green composite states of the system. E.g., snapshot event and download datasets event can occur together, even though snapshot requires only Presentation Layer

and download requires Presentation and Business Layer. Blue composite states disable the system until it is completely executed or the user explicitly triggers an event in the internal state of the composite state. Pink states are similar to blue states except that these states are reachable only if the system has a logged in user. E.g., when user uploads a new graph in the system, the system is disabled for the user to interact with until the graph is uploaded into the system or an appropriate message is sent by the REST API.

The entry state of the system is the landing page where the user is not logged in and only the public datasets are available for the user. The user can then register on the web application or log in. All the pink event boxes indicate that these actions can be reached only if the user is logged in. As mentioned before all the events shown in green and blue states can be reached irrespective of the user's login state. The only difference is that the events might occur on different set of epicurves because the graph and workspace collection of a logged in user may be different than that of a user who is not logged in. When the user exits the system, it is either when the user is logged in or logged out.

3.3 Deployment

There are two deployment options for this application. First, standalone instance accessible directly through the browser; second, is the instance integrated into a larger analytic ecosystem.

Standalone version: This deployment option offers EpiViewer as a standalone web application that can be run in a web-browser. It is an independent instance with its own database, and is not directly connected to any other application. The deployment/component diagram in Figure 3.6 depicts the standalone instance. There are three physical nodes in the system. One is the user machine where the app is rendered on the browser via an http connection

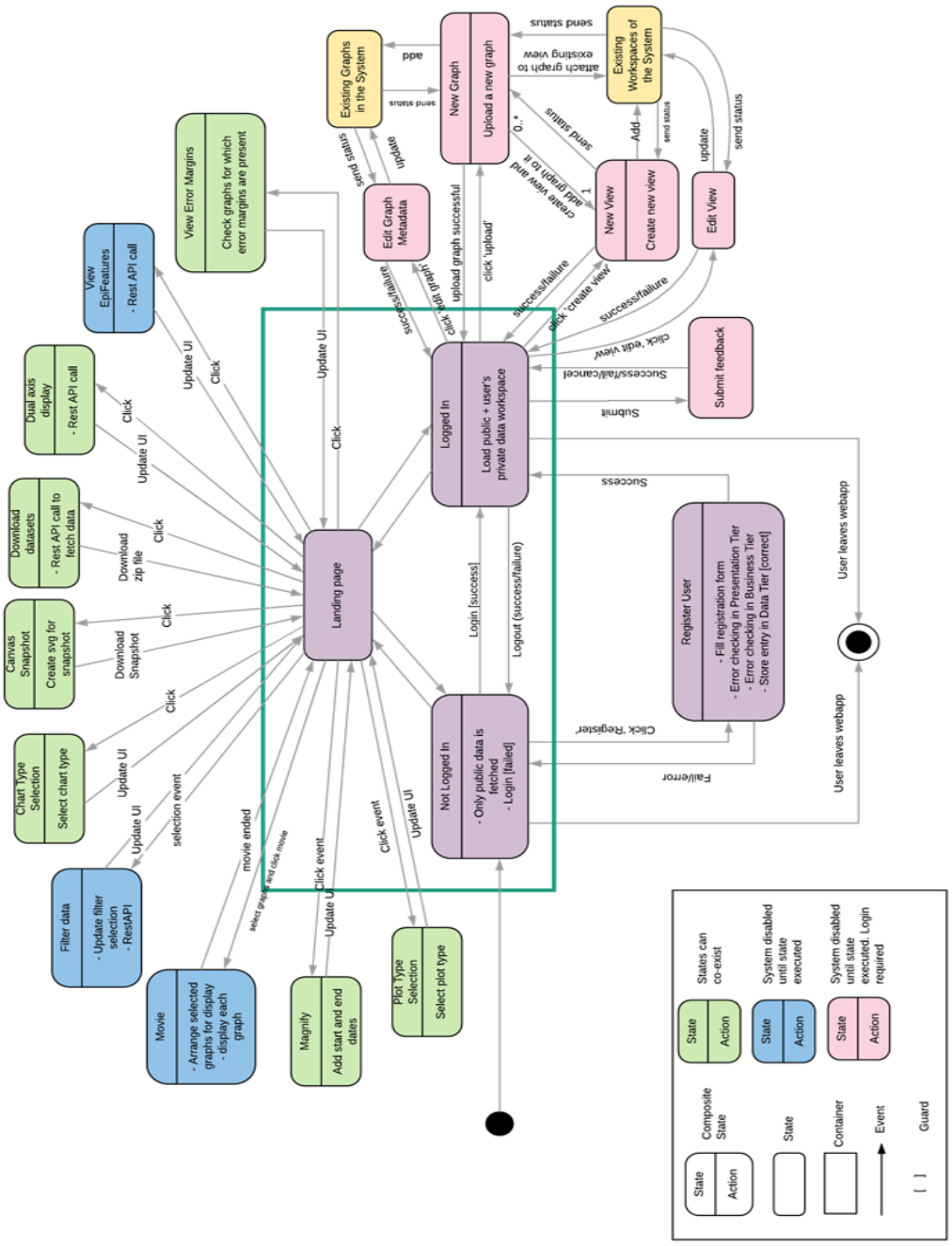

Figure 3.5: Statechart for EpiViewer. The statechart depicts high-level events and actions of the system.

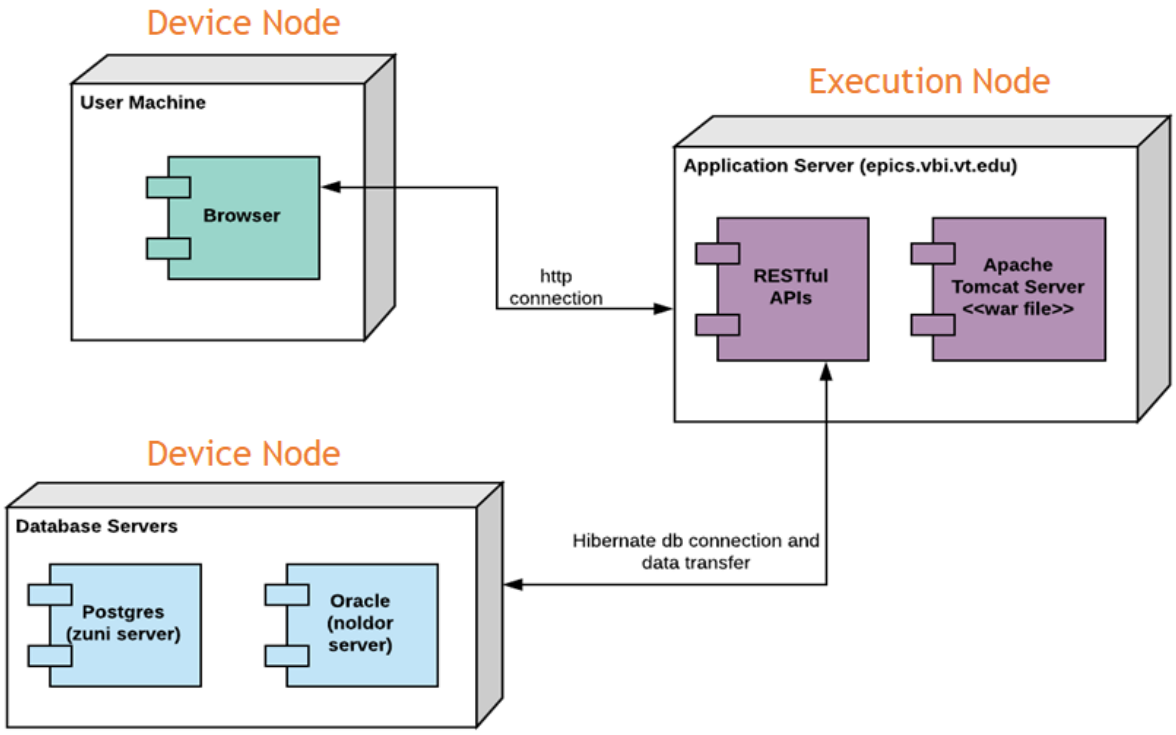

Figure 3.6: Component/deployment diagram for standalone instance of EpiViewer. The application loads on the user machine browser via an Http request. The application server contains the web archive artifact deployed in an Apache Tomcat Web Server. The RESTful API services communicate between the Application Server and User Machine nodes. The database servers host the different RDBMS databases supported. RESTful APIs alongwith the Hibernate framework is used for querying and data transfers between the Application Server and Database Server.

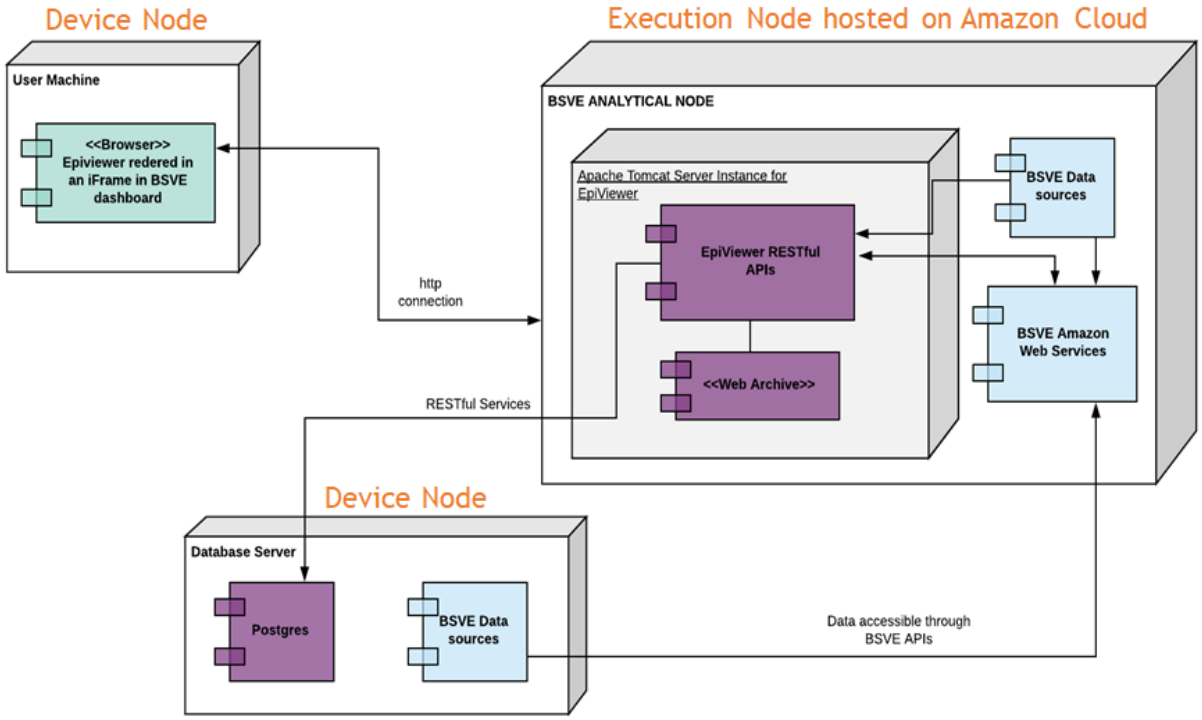

Figure 3.7: Component/deployment diagram for BSVE instance of EpiViewer. The BSVE Amazon Web Services and EpiViewer RESTful APIs interact with each other for user management, data exchange and import, etc. The EpiViewer application is rendered in an iFrame HTML container in the BSVE Analytical App Dashboard on the browser. The application is rendered via an Http request and user authentication token. The Apache Tomcat Server container hosts the web archive artifact and the application RESTful services.

with the app server node. The Apache server deploys the web archive artifact and the API infrastructure. This is the execution node since it holds the core logic processing capabilities. Data exchange takes place from the database server via REST APIs and hibernate framework.

Integration with the BSVE: Biosurveillance Ecosystem (BSVE) is a large-scale analytics platform funded by the Defense Threat Reduction Agency (DTRA) for the analysis, visualization, and curation of real-time global epidemic and outbreak data. BSVE has a repository of data sources collated by DTRA, Los Alamos National Laboratories, and others [19, 29]. While BSVE offers applications customized to provide visualizations and analytical meth-

ods for specific data sources within its repository, EpiViewer allows users to compare data across multiple data sources (from BSVE system) along with its own datasets. EpiViewer is incorporated in the Analyst Workbench of the BSVE. This implementation offers the functionality of the standalone version as well as additional features that allow coordination with various BSVE components and data sources. Figure 3.7 shows the overall view of how EpiViewer has been integrated into the BSVE platform.

All the application features have been retained in this deployment instance except the user authentication module. A few add-on features such as import mechanisms, BSVE specific navigations and displays have been added. Three nodes exist in this system as well. The BSVE has its own database server instance. All data from apps in BSVE and from EpiViewer are hosted on the BSVE database node. REST APIs from BSVE and EpiViewer are used for data exchange. EpiViewer has its own Postgres database schema on the BSVE Database Server node. This data is accessible via the Amazon web services. EpiViewer RESTful APIs and BSVE Amazon Web Services interact to import data into EpiViewer. The user machine renders BSVE platform in the browser and renders EpiViewer upon user request in an HTML 'iFrame' (inline frames) in the dashboard. IFrames can be hidden and displayed dynamically which makes them a perfect candidate for loading and unloading EpiViewer App from the Analytical Dashboard of the BSVE platform. The important node is the BSVE execution node that is hosted on the Amazon Cloud platform. This node hosts all the BSVE applications in its own server instance containers. EpiViewer is hosted in the BSVE environment similar to the standalone style. The Apache Tomcat Server container hosts the web archive artifact and the application RESTful services. The BSVE node also hosts the Amazon Web Services and the dossier datasets (user created excel like datasets). The value added by EpiViewer in the BSVE system is that it is the only app that allows users to explore time series for any disease across multiple data sources from BSVE as well as

data internal to EpiViewer. BSVE offers data exchange through Amazon web services which serve the purpose of in-between app communications, authentication and user management, and other access controls. In order for EpiViewer to interact with this system, we did not have to rebuild EpiViewer. The flexible and extensible web-service oriented architectre was carefully re-designed and made scalable and usable. The same endpoints were used to perform different action depending upon user request and an extensive number and variety of modules and workflows were added in the business tier and attached to an existing endpoint. We exploited the SOA paradigm of 'services using other services to accomplish a task'. We will look at a simple example, when a user requests for a BSVE related data for graph display or addition into user's personal catalogue then, the EpiViewer REST APIs combine with BSVE APIs, execute the business logic or combine various complex workflows to assemble data and send/render the appropriate response to the end user.

Note, in both Figures 3.6 and 3.7, the user machine nodes can be many and access the system concurrently. Multiple http connections can be established with the execution node. The user machine essentially represents the Presentation Layer, the execution node represents the Business Tier and the Database node represents data layer in a conceptual manner.

Noteworthy Challenges:

Important challenges were overcome in the successful design of this system. The same codebase has been used for the standalone as well as the BSVE instance. A switch variable accounts for adding all the additional services when deploying a BSVE instance. Rendering different user interface in both instances was challenging and required careful attention to detail for the design in terms of screen size, visibility of BSVE icons and dropdowns, etc,. The MVC architecture was modified for the BSVE instance. Additional views and models were added. A separate controller was added to route BSVE requests to the existing Con-

troller or to the REST API endpoints.

The service architecture needed careful design and balance so that the same service endpoint could be used to request data without the user having to worry about it. Internal services were added to combine complex workflows and assemble data or check for errors and inconsistencies.

3.4 Important Methods

When a graph is uploaded in the system a few workflows are initiated by the system: e.g., conversion of a graph from incidence to cumulative and vice versa. Similar calculations are performed when a graph is uploaded with error margins in an incidence or cumulative form. When data exhibits large spread in terms of magnitude then a naive procedure is invoked for grouping data on dual y-axes. A statistical epi-features module is designed. These epifeatures are calculated for each time series. Total count, peak time, peak value, first take-off time, first take-off value are the epifeatures available for each graph in the system.

3.4.1 Epi-Feature module

Epi-features are statistical characteristics of an outbreak that can help researchers interpret the quality of the epidemic curves and identify any anomalous time series [36]. This is useful information for the expert user base of EpiViewer such as scientists and epidemiologists. The system provides a module that calculates three of all the epidemiological metrics for the time series in the system. Three most important metrics are chosen – Total count, peak time, peak value, first take-off time and first take-off value. Total count is the cumulative

count at the end of the time series. Peak time and value is the entry in the time series data that has the highest incidence value for a given day. The calculations for these two metrics were straightforward.

Usually epicurves remain slow in spread or are dormant at the start and suddenly exhibit a sharp rise in the number of cases just as the season picks up or an epidemic takes off. This value is the first take-off value and the first take-off time. According to Tabataba et al. [36], "Mathematically, first take-off is the time at which the first derivative of the epidemic curve exceeds a specific threshold". Refer equation

$slope(x, \Delta t) = \frac{x(t+\Delta t)-x(t)}{\Delta t}$

where Δt =time steps, x = incidence count at time t

The first take-off threshold value typically depends on the type of disease and the outbreak severity, so this threshold is normally established by domain experts. The influenza analysis carried out in [36], used an expert to set this value. However, a web application that allows users to add new diseases, it is not reasonable to expect domain experts to establish the threshold values for every disease added to the system. Another option was employing switching regression method [33]. This method estimates parameters of a linear regression system satisfying two separate regimes. This method has been widely used in the fields of economics, housing demand, and urban science [14, 22, 26, 32] for finding discontinuities in urban population densities, understanding housing demand regime for primary and secondary house ownership, finding wage determinants in public and private sectors. The method estimates the coefficients and variances in each regime using Quandt's method. Once the coefficients are estimated we will know which points come from which regime. A maximum likelihood estimate can then be calculated for every iteration of linear regression for the two regimes. The best fit can then be found. However, for simplicity and quick on-the-fly calculations, we use a piecewise linear regression approach to determine the first

take-off value and time. Piecewise linear regression, or "broken-stick regression", is a method of regression analysis in which the independent variable is partitioned into intervals, and a separate line segment is fit to each interval. When applied to epidemic curves [16, 39, 40], this technique can be useful for identifying the time when an epidemic first takes-off. A similar approach is adopted by Cecile Viboud's groups in their paper [16, 39, 40] which deals with estimating the onset time of influenza for a season. The curve from start until the peak value is reached, is the minimum requirement for this method to work successfully. The optimal breakpoint found by the method is the threshold value. We simplify the method for estimating this breakpoint.

The original line is successively partitioned into two parts and regression lines are fit to each of these partitions. The partitions for which the difference of the sum of squares is the least is taken as the best fit. The value of the line segment at which the best partition occurs is the first take off point for the line segment. This point best describes the start of the steep slope for the epidemic curve. Figure 3.8 illustrates the process of partitioning the data points and applying the linear fit to the partitions. Algorithm 1 describes the procedure for calculating the first take-off time.

3.4.2 Assignment of Time Series to Dual Y-Axes

When graphing multiple time series on a single canvas, differences in orders of magnitude between the time series (i.e., between cases and deaths) can effectively cause one time series to be "flattened", which can complicate identification of trends. To address this issue, EpiViewer provides the option of splitting time series across dual y-axes; assignment is performed using the following steps:

(i) Fetch the time series data from the database. This acts as the source data.

Algorithm 1 Calculating Epi-feature 'First Take-off Time and Value'

Input := Time series T having n records, where T_i is the $i^{th} < date, value >$ tuple
Output := $firstTakeOffPointDate, firstTakeOffPointValue$
procedure CALCULATEFIRSTTAKEOFFPOINT
    Let $SSE_{partition}$ <> be the object list storing date, count, difference of sum of squared errors (SSE) of the partitions
    Sort n records of T in ascending order by date.
    for $(counter = 2; counter < n; counter++)$ **do**
        $partition_{left} \leftarrow [T_1, ..., T_{counter}]$
        $partition_{right} \leftarrow [T_{counter+1}, ..., T_n]$
        for each *partition* **do**
            Find best linear fit
            Record sum of squared errors (SSE)
        end for
        $SSE_{partition} \leftarrow < date_{partition}, value_{partition}, (SSE_{left} - SSE_{right}) >$
    end for
    $firstTakeOffPointDate \leftarrow date$ at $min(SSE_{partition})$
    $firstTakeOffPointValue \leftarrow value$ at $min(SSE_{partition})$
end procedure

(ii) Calculate the maximum value across all the time series on the canvas from the source data. (This maxima is recalculated every time a time series is added to the view.)

(iii) Divide the overall maxima (derived in Step 2) by 2.

(iv) The maxima is now calculated for each time series and compared with the value obtained in Step 3. If the time series maxima is less than the Step 3 value, then it is assigned to the left axis; otherwise, the time series is assigned to the right axis.

3.4.3 Uncertainty Bounds Calculation

When a user imports data from a text file or another web service, the application saves the time series in its current plot type (incidence or cumulative), but also converted to the other plot type (i.e., uploaded incidence data is saved also in cumulative format and vice versa); converting the time series upon data load improves performance when the user toggles

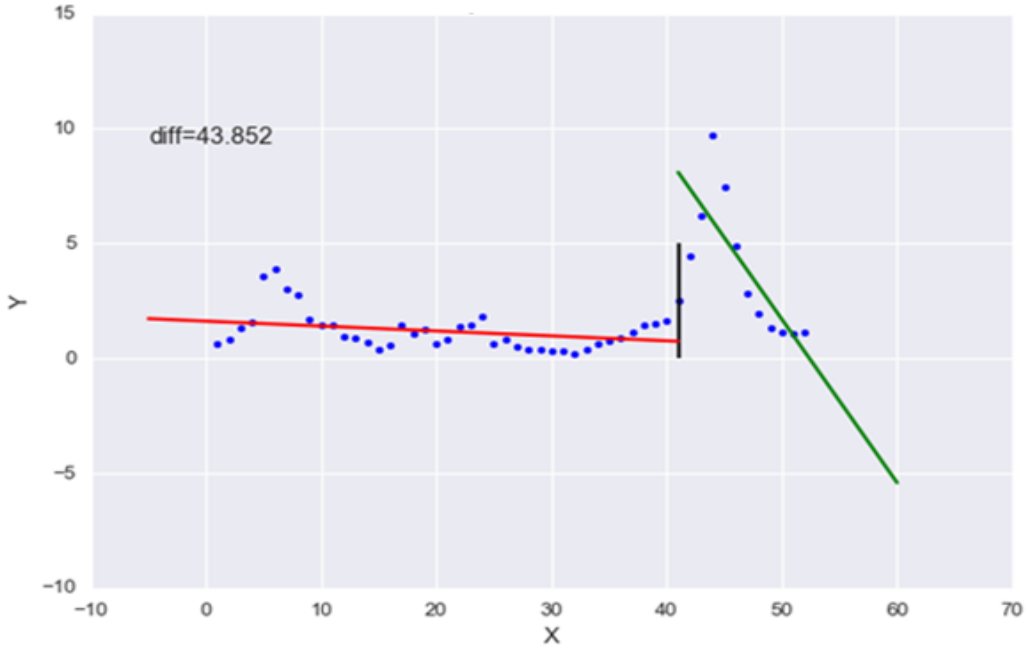

Figure 3.8: Depiction of the piecewise regression method utilized for calculating the Epi-feature 'First Take-off Point'. In this illustration, the blue dots represent time series points on the epicurve; the black line indicates where the partition is for the current iteration; and the red and green lines indicate the line segments fitted to the two intervals.

between the plot types later on. If the uploaded time series includes uncertainty bounds, then the uncertainty bound data also has to undergo a similar conversion.

Calculating cumulative data points from incidence time series is fairly straightforward, since the cumulative total for each date is simply the sum of the infection counts from the beginning of the time series through that date. However, calculating the cumulative uncertainty bounds is more complicated; simply summing the values would result in an ever-increasing cone of uncertainty. To mitigate this issue, we take the following approach:

(i) Sort the incidence time series dataset in ascending order by date.

(ii) Looping through the sorted dataset, calculate the cumulative value for each date as the sum of the infection counts thus far.

(iii) Also within the loop, for each date calculate the incident lower delta value as the

difference between the incident value and the lower bound, and the incident upper delta value as the difference between the upper bound and the incident value. For example, if the infection count on a given date is 21, the lower bound is 18, and the upper bound is 22, then the lower delta value would be 3 and the upper delta value would be 1.
(iv) Taking the maximum upper delta value calculated so far, add this to the cumulative value calculated in Step 2, and that will be the upper uncertainty bound for this date. Similarly, take the maximum of the lower delta values calculated thus far, and subtract that from the cumulative value from Step 2 to establish the lower uncertainty bound for this date.

When cumulative data is uploaded with uncertainty bounds, a similar process calculates the incidence uncertainty bounds, except there is no comparison to the maximum delta values performed in step 3 above – we simply subtract the lower delta and add the upper delta from the cumulative uncertainty bounds to the resultant incidence value. These processes are illustrated in Figure 3.9.

3.5 Web Services

The Business Tier is structured using Representational State Transfer (REST) web services and the Hibernate Java framework. This architectural style considers data and functionality as resources accessed using Uniform Resource Identifiers (URIs), typically links on the web. We use the Jersey RESTful Web Services framework, an open source framework for developing RESTful Web Services in Java. Data formatted in JavaScript Object Notation (JSON) is used for communication between the tiers.

Date	Incident Value	Lower Bound	Upper Bound	Lower Delta	Upper Delta	Cumulative Value	Lower Bound	Upper Bound
10/10/2017	310	303	314	7	4	310	303	314
10/17/2017	299	295	305	4 (Max = 7)	6 (Max = 6)	609	602	615
10/24/2017	304	295	307	9 (Max = 9)	3 (Max = 6)	913	904	919
10/31/2017	325	321	329	4 (Max = 9)	4 (Max = 6)	1238	1229	1244

Incidence conversion to Cumulative Plot Type

Date	Cumulative Value	Lower Bound	Upper Bound	Lower Delta	Upper Delta	Incident Value	Lower Bound	Upper Bound
10/10/2017	310	303	314	7	4	310	303	314
10/17/2017	609	602	615	7	6	299	292	305
10/24/2017	913	904	919	9	6	304	295	310
10/31/2017	1238	1229	1244	9	6	325	316	331

Cumulative conversion to Incidence Plot Type

Figure 3.9: **Uncertainty bound conversion across plot types.** These illustrations show how the uncertainty bounds from an uploaded time series in Incidence format can be converted for display in Cumulative format, and from an uploaded Cumulative time series to Incidence Format. If converting from Incidence to Cumulative format and back again, the new Incidence uncertainty bound values may differ from the original set.

3.6 Services

Table 3.1 shows all the services of the EpiViewer application. Important services have been described following the table.

Table 3.1: List of services.

ServiceLayer	ServiceName
User Management	addUser() authenticateUser() logoutUser() getAllUsers() saveFeedback()
View (workspace)	getViews() createView() updateView() deleteView() getGraphsByView() downloadGraphData()
Metadata	getAllRegions() getAllDiseases() getAllDataTypes() fetchAllDiseasesCreatedByUser()
Timeseries	getGraphsByView() addGraph() updateGraph() deleteGraphData() uploadGraph() getMetrics() getGraphSelectionForView() getCumulativeDataForBarGraphs()

createView

Create a new view/workspace to load graphs. This feature is available once you have an account with EpiViewer. The service endpoint can be used to create a view external to the application (e.g. through a script).

Parameters

Name	DataType	Description
view	View object	Information wrapped in a object for creation of a new view

Response

Name	DataType	Description
response	String	A JSON object converted to a string containing 3 fields: Id: Unique identifier for the view. DiseaseId: Disease to which the view belongs. Status: Message describing SUCCESS or FAILURE for view creation

updateView

Updates an existing view/workspace details. The graphs in the view will not be altered unless the view is deleted.

Parameters

Name	DataType	Description
view	View object	This object contains the new information to be updated

Response

Name	DataType	Description
response	String	A JSON object converted to a string containing 2 fields: Id: Unique identifier of the view. Status: Message describing SUCCESS or FAILURE for view updating

downloadGraphData

Downloaded all graphs in the given viewId in csv format. The csv files contain all the metadata attributes and the actual data.

Parameters

Name	DataType	Description
viewId	Number	The unique view identifier for which you wish to download data

Response

Name	DataType	Description
data	Response	A class used to build Response instances that contain metadata instead of or in addition to an entity. It is a MIME attachment with the content type 'application/octet-stream' is a binary file.

addGraph

Add a time series and its tagged metadata into the database. This service is called when you upload a graph from the user interface. The updateGraph() service is called if you have to edit the metadata fields of the time series. The time series data itself cannot be altered.

Parameters

Name	DataType	Description
HTML elements data	Multipart form data	Multipart/form-data allows entire files to be included in the data.

Response

Name	DataType	Description
response	String	A JSON object converted to a string containing 4 fields: Id: Unique identifier for the view. DiseaseId: Disease to which the view belongs. Status and Message: String describing SUCCESS or FAILURE for graph upload

uploadGraph

Add a time series and its tagged metadata into the database. This service is called external to the system. The service endpoint can be called by a script in any language. The url and formatted JSON object should be invoked to add the graph into the system. This service is used by BSVE's event trackers and data sources to import data. This API facilitates uploading multiple timeseries (should belong to a single view) in the system. The updateGraph() service can be used from the UI if you have to edit the metadata fields of the time series. The time series data itself cannot be altered.

Parameters

Name	DataType	Description
inputJSON	A JSON object containing fields: data, username, graphName, fileName, description, disease region, plotType, categoryType dataType, dateFormatType forecastedOnDate, graphVisibility	JSON should be converted to string and then passed as input to the service call.

Response

Name	DataType	Description
response	String	A JSON object converted to a string containing 5 fields: viewId, diseaseId, diseaseName, status, message

getMetrics

This service calculates epi features for every graph selected in the filters or present in the canvas area. All the epi features are calculated on the fly every time the user clicks the epi features button in the 'user actions' panel.

Parameters

Name	DataType	Description
graphIds	String	Comma separated list of graph ids

Response

Name	DataType	Description
metricJSON	JSON Array	A JSON array where each JSON object contains: graphId, graphName, peakValue, peakTime, totalCount, firstTakeOffValue, firstTakeOffTime

getGraphSelectionForView

Get all the graphs / time series required while creating new view or editing an existing view. This is an important service since users have access to their own private collection of workspaces and then publicly available workspaces. A logged-in user is entitled to choose graphs from any of these collections into a new view or add them into an existing view.

Parameters

Name	DataType	Description
diseaseId,viewId,username	Number,Number,String	Attributes for graph selection in a view

Response

Name	DataType	Description
graphs	Map[String, List[GraphMetadata]]	A dictionary containing graphs from the respectives user's public and private views. If editing a view, then the existing graphs in the view are also shown. Three keys are present in the map, namely, 'publicGraphs', 'privateGraphs', 'existingGraphs'

Chapter 4

Results & Discussion

4.1 Application Features

The application is divided into 3 panels: Data Configuration and Filters, User Actions, and the Canvas. Figure 4.1 shows a snapshot of the application.

Views: Views are workspaces which allow users to organize their time series in logical ways. For example, a user who studies Influenza could create a view for each Influenza season instead of trying to crowd multiple seasons on the same canvas. Views are private by default, which means that the data is visible only to the owner, but users can also make their views public to allow other researchers to build on their collated data. Time series can be added to the views by importing comma-separated-value (CSV) time series files, or by copying time series from other views. Time series can also be removed from the views, either temporarily by filtering, or permanently via deletion.

Import data: Importing time series data from CSV files is one of the main features of the system. The file format for the data file should include a row for each data point in the

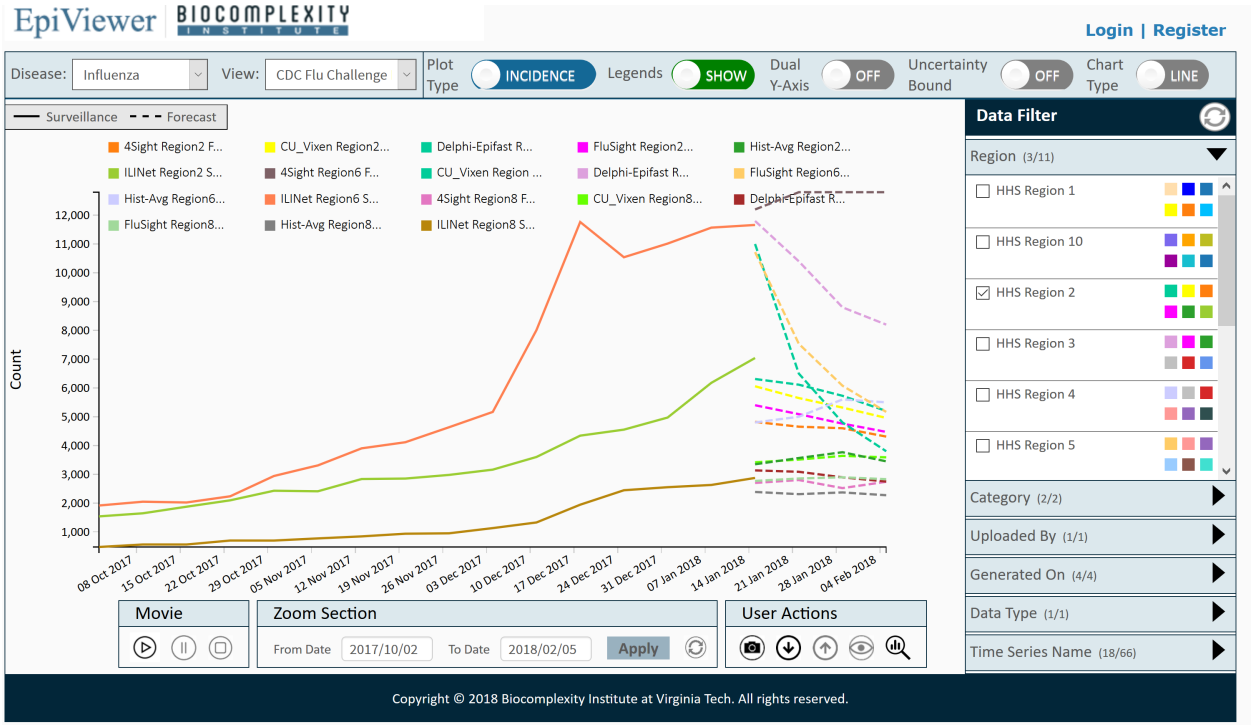

Figure 4.1: EpiViewer User Interface has 3 panels. The canvas is the area where time series graphs from a workspace (view) are displayed. The user actions panel located at the bottom of the canvas helps users perform multiple operations on time series present in the canvas area. These actions include upload time series, download view data, snapshot of the canvas, play a movie, zoom data, epi-features for time series in the canvas, workspace (view) functionalities. There are 2 data configuration and filter columns on the top and right-hand side of the canvas. The filters on the right facilitate drilling down using metadata attributes and the top data configuration panel configures the output in the canvas area. Color legends are also shown in the 'region' and 'time series name' filters.

series, with the date in the first column and data counts in the second column; if the data contains an uncertainty bound, the lower bound would be entered in the third column, and the upper bound would be included in the fourth column. A single row in an import file might look something like this:

10/17/2017, 237, 228, 245

When uploading a data file, users are prompted to define metadata attributes that apply to the time series, such as the time series name, whether the dataset is surveillance or

Figure 4.2: **Incidence line chart view.** When the Plot Type is set to "Incidence", then the line chart displays the number of reported cases (or deaths, etc., depending on data type) on that date. This means that the line graph is likely to rise and fall with the epidemic activity.

forecast data (category), the region where the data was collected, whether the imported data is aggregated at the incidence or cumulative level (plot type), the timestamp when the data was obtained, etc.; this metadata can be used to filter the time series on the canvas. An additional metadata attribute is Associated Graph: a time series can be associated with another time series within that view to indicate a relationship between the datasets. Associated time series are displayed in the same color so researchers can easily identify those dependencies; the only constraint for graph association is that the two time series must be of different categories (i.e., surveillance time series can only be associated with forecasts, and vice versa.)

When EpiViewer is deployed within the BSVE, there are custom services for importing data from different BSVE data sources, including EpiArchive and the Event Trackers.

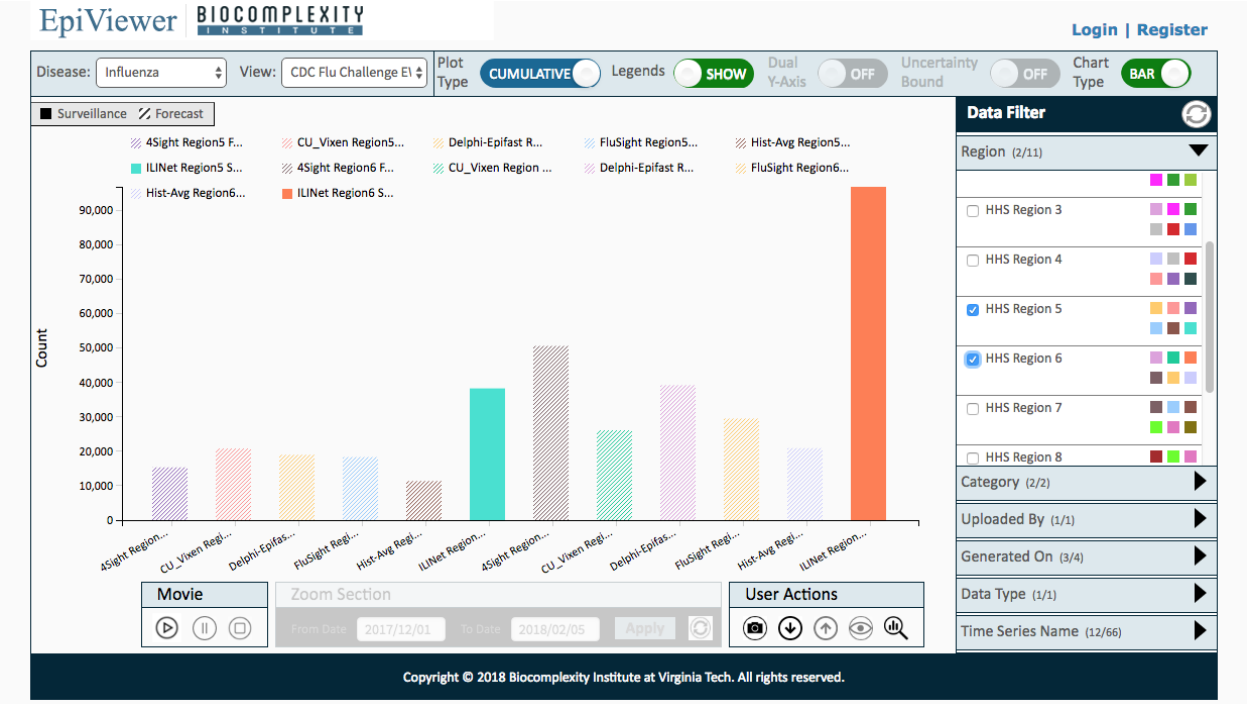

Figure 4.3: **Cumulative bar chart view.** When the Plot Type is set to "Cumulative", then each date would typically show the total number of cases (or deaths, etc.) as of that date. In line chart view, this would show the time series going up until possibly leveling out at the end of the epidemic; in bar chart view, the cumulative total is shown for each of the time series.

4.1.1 Canvas

The canvas panel is where the time series graphs are displayed. Surveillance graphs appear on the canvas as solid lines, while forecasts are displayed as dashed lines. Metadata and epi-feature information for a particular epi curve can be viewed on the canvas by hovering the mouse over the legend entry; this action also highlights the curve in the canvas area for readability. When the mouse cursor is hovering over a data point on the selected curve, the curve is highlighted and a small pop-up shows information about the data point, including the data count and date for that entry.

In Figure 4.1, the canvas area shows multiple Influenza time series for three different HHS

regions of the United States. These time series are from the CDC Flu Challenge. The solid lines represent surveillance curves. The others are forecast time series from different teams participating in the challenge. Four of these time series were uploaded with uncertainty bounds, or margins of error for the forecasts; these bounds will be visible once the 'Uncertainty Bound' option is selected. The curves have been filtered using the Region filter in the panel on the right hand side. Different HHS regions exhibit different magnitudes of Influenza case counts. The data type represented by these counts are calibrated case counts. The user can add this data type while uploading the curves or simply edit the type once the user imports it in his private view. We also understand that there are 6 types of colored curves belonging to each of the HHS regions. This is indicated by the legends under the Region filter. Similar legends are present in the time series name filter. If the user has ground truth data for deaths or Flu vaccinations from each HHS region, the user can compare those graphs with the existing case type graphs in the system.

4.1.2 Data configuration and filters

EpiViewer offers a wide variety of display options and filtering capabilities to help researchers identify trends and make comparisons between time series that would be difficult to achieve through examination of standard chart data.

Users start off by selecting the disease they wish to investigate from a drop-down list located at the upper left corner of the page; this selection will filter the dropdown list of views the user can choose from, which include those that are privately owned by the user as well as views that have been made public. The top panel section offers a 'Plot type' display option where users can choose to view the time series in either incidence or cumulative format. The user can also display the uncertainty bounds if margin of error data is available for their

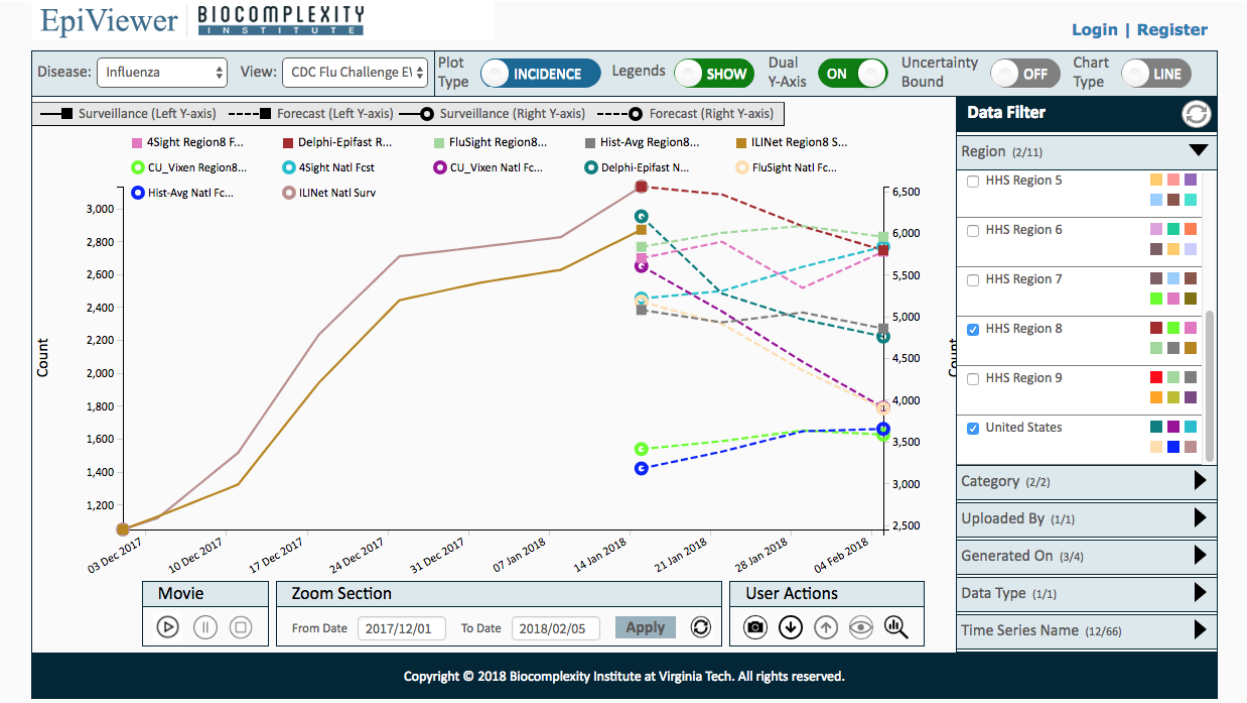

Figure 4.4: **Distributing time series across dual y-axes.** If the user is having trouble interpreting a time series because it appears flat compared to other time series on the canvas, toggling the Dual Y-axis switch to "On" will add a second, separately scaled y-axis to improve visualization of the time series.

uploaded time series. Users can toggle between line and bar chart formats; these different perspectives can be useful for differentiating between the temporal aspect (line chart) as well as the cumulative (overall) impact of the disease (bar chart). Distribution of time series across dual y-axes based on order of magnitude is also available in this section via the 'Dual Y-Axis' option. Refer to Figures 4.2, 4.3, and 4.4 for details. The dual y-axis option is handy for comparison of data having a wide range. For example, in Figure 4.4, using two y-axes makes it possible to compare the time series trends between a smaller region (HHS Region 6) and a country (U.S.). We can now clearly see that the surveillance curves follow the same pattern/trend, but the forecasts for both the regions have different trends.

Users can filter which time series appear on the canvas via data filters available on the right side of the EpiViewer display. The data filters rely on the metadata entered for each time

series; they can be filtered based on Region, Category (Surveillance/Forecast), Uploaded By (owner of the original time series), Data Type (Cases, Deaths, or custom data types), Generated On (the date the surveillance data was collated or the forecast was predicted) and Time Series Name.

An example of data collected during 2014 Ebola outbreak from different sources such as World Health Organization, Columbia Lab, MoBS lab and NDSSL lab are displayed in Figure 1.1. The forecasts produced by MoBS and NDSSL are better than the others for Sierra Leone.

4.1.3 User actions

This section describes the actions that users can perform on time series present in the canvas area.

Movie: The movie feature allows users to watch as the plots are laid out on the canvas in order of Generated Date so users can see how surveillance and forecast predictions have evolved over time. This can be especially useful when investigating a volatile epidemic, or if evaluating how epidemic predictions are made. To begin the movie, click on the arrow icon under the Movie header. While the movie is running, it can be paused or stopped at any point for closer examination of the time series laid out thus far. A movie use case of Ebola 2014 outbreak data has been shown in Figure 4.5 for the Sierra Leone region. The canvas has 3 surveillance curves serving as ground truth. Then, data was collected from various organizations to see how each of them was predicting the outbreak, given the surveillance curve. Each snap shows plots laid out in the order of timestamp. It is interesting to see that some organizations' forecasts could predict the nature of the epidemic.

Zoom section: Users can reduce the scope of the canvas display by zooming in on a specific

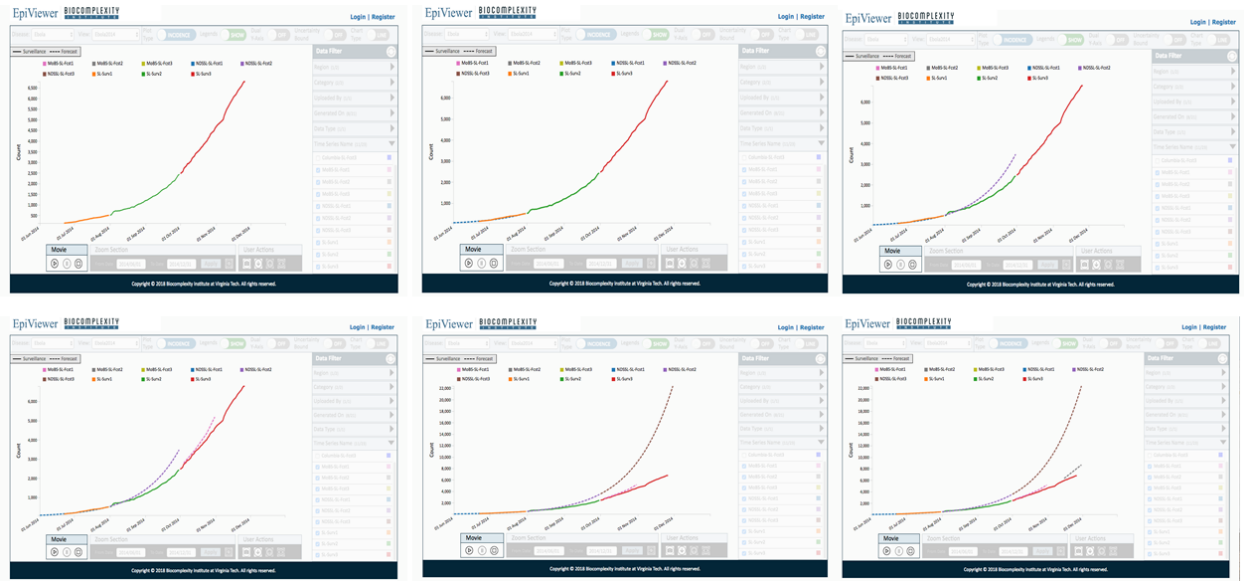

Figure 4.5: **Movie of the 2014 Ebola outbreak progression.** This figure shows how data obtained over time shows information about the outbreak.

date range. This is done by clicking in the 'From Date' box of the 'Zoom Section' area at the bottom of the page; this will pop up a calendar widget where the user can select the desired start date. The user then repeats the process in the 'To Date' field to enter the end date for the zoom. Finally, the user can click on the 'Apply' button to zoom into the specified date range. When they want to return to the full canvas, they can click on the refresh icon to the right of the 'Apply' button.

Download data: Users can download the loaded view's underlying time series and epi-features to CSV format for evaluation outside of the EpiViewer system. The down arrow under the User Actions header will generate a zip file containing all of the plots, each in their own CSV file, as well as a CSV file with a list of the performance metrics.

Snapshot: The Snapshot feature under User Actions allows the user to download a picture of the graph. This can then be sent to other users, or used in documentation or publications.

Epi-features:

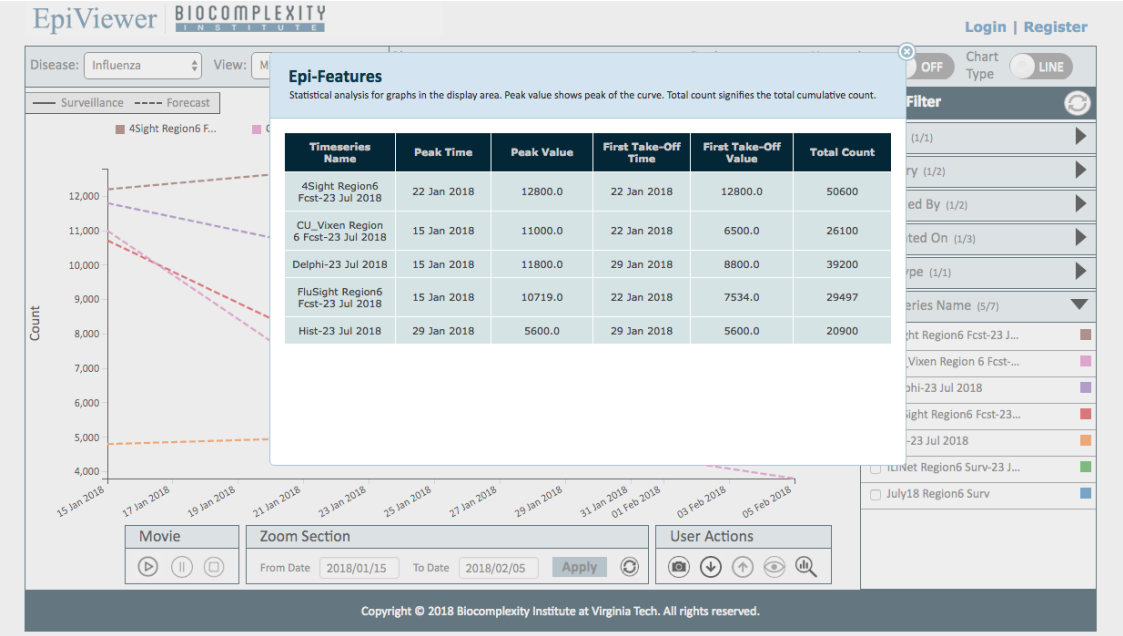

Figure 4.6: **Epi-Features (Performance Metrics)** Users can view performance metrics of the time series on the canvas to try to assess the quality of the data sets. For example, if the viewer believes the peak value of a time series is too high, or that its first take-off value is too early in the season, they may give more credence to a different forecast with better characteristics.

The Epi-features provided by EpiViewer are described below:

Peak time and value: Peak value is the highest infection count over the course of the epidemic time series. The date when the peak value occurs is called the peak time.

Total count: Total count is the total (cumulative) number of infections over the duration of the time series.

First take-off time and value: Some infectious diseases, like Dengue, start out almost dormant in the beginning, then suddenly exhibit a sharp increase in the number of cases just as the season commences. The point where this sharp increase occurs is the first take-off value.

On clicking the 'analyze button in the 'User Actions' section, a pop-up display will present three epidemiological statistics - peak value and time, first take-off value and time, and total count - for every time series present in the canvas area. This layout depicts a comparative

view of these values across the time series. An example of this feature can be viewed in Figure 4.6.

4.1.4 Supplementary features

EpiViewer has a user management system which tracks users uploaded graphs and views. Users can upload graphs and perform view manipulations only if they are logged in. The users can provide input about the application or any issues they face by clicking on the Feedback link next to the Logout link on the upper right-hand corner of the display.

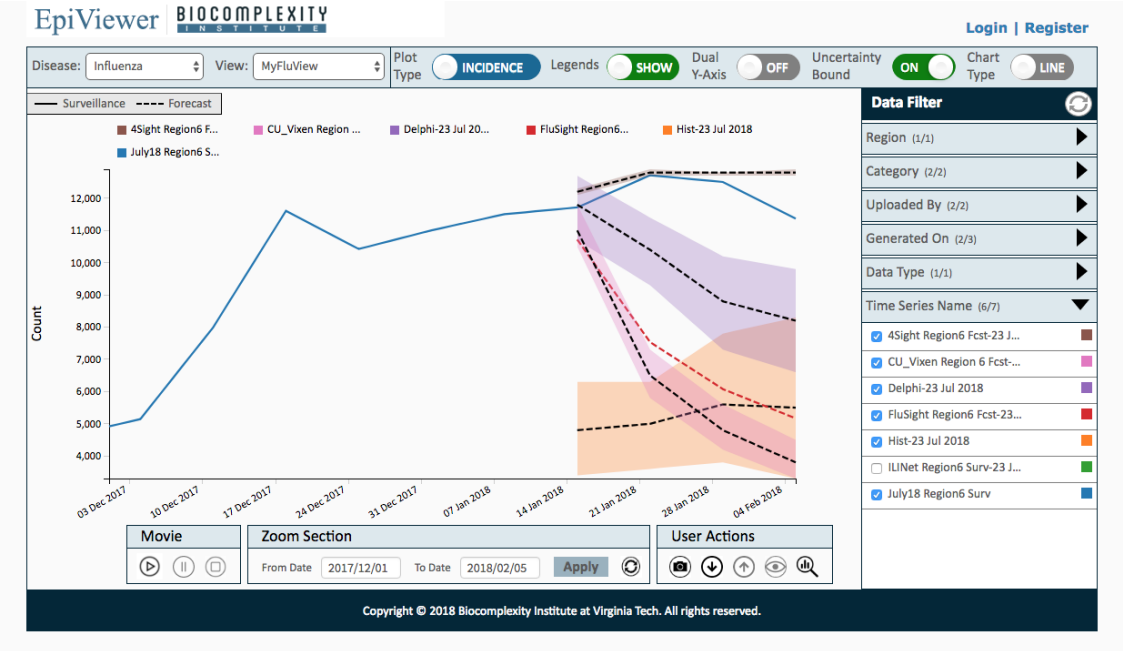

Figure 4.7: Example scenario depicted by the canvas area

The canvas area in Figure 4.7, shows six Influenza time series for HHS Region 6 of the United States for the current season 2017-18 CDC Flu Challenge. The solid blue line represents a surveillance curve for HHS Region 6. The other five forecast time series represent different teams that participated in the challenge. The error limits on four of these time series are visible since the 'uncertainty bound' option is selected. The curves have been filtered using

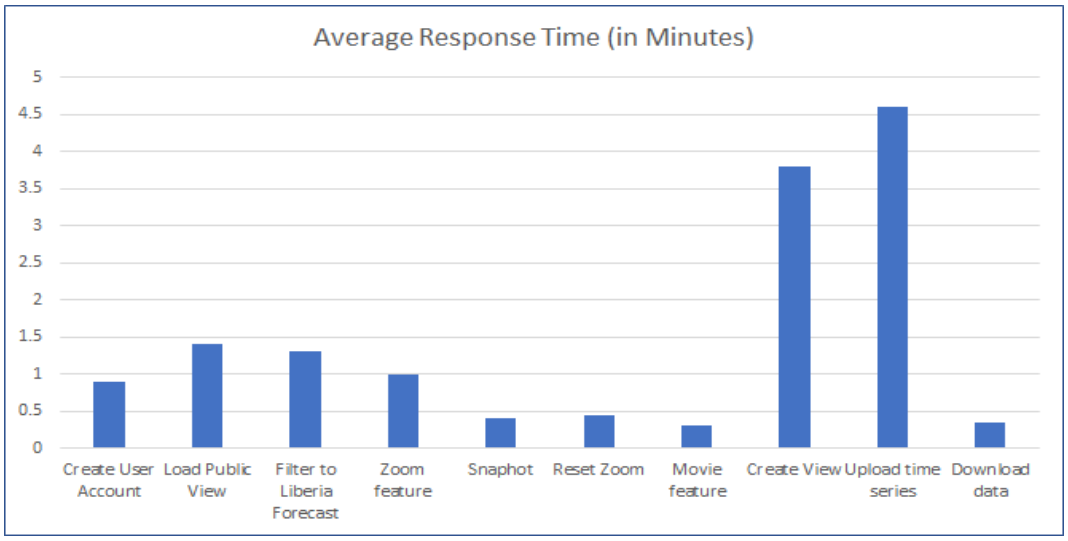

Figure 4.8: Average response times for performing assigned tasks in EpiViewer. In the user study, participants were able to complete an assigned list of tasks in a reasonable span of time. The average total time spent on the 11 tasks was 24 minutes.

the panel on the right hand side. A quick observation shows that the forecast generated by '4Sight' team for HHS Region 6 is the most accurate forecast as compared to the others.

4.2 User Study

We conducted a user study to assess EpiViewer's ease of use. The ten participants (faculty, staff, and students at Virginia Tech) had not used EpiViewer before, and came from a variety of academic backgrounds, including computer science, epidemiology, and public health.

At the beginning of the user study, an instructor provided a brief overview of the application, including an explanation of the problem it was designed to solve. Users were then given an opportunity to try out the system by performing a checklist of 11 tasks covering the important utility functions of the application, like importing and filtering time series, and user actions like taking snapshots of the data; a complete list of the tasks are included in the supplementary materials.

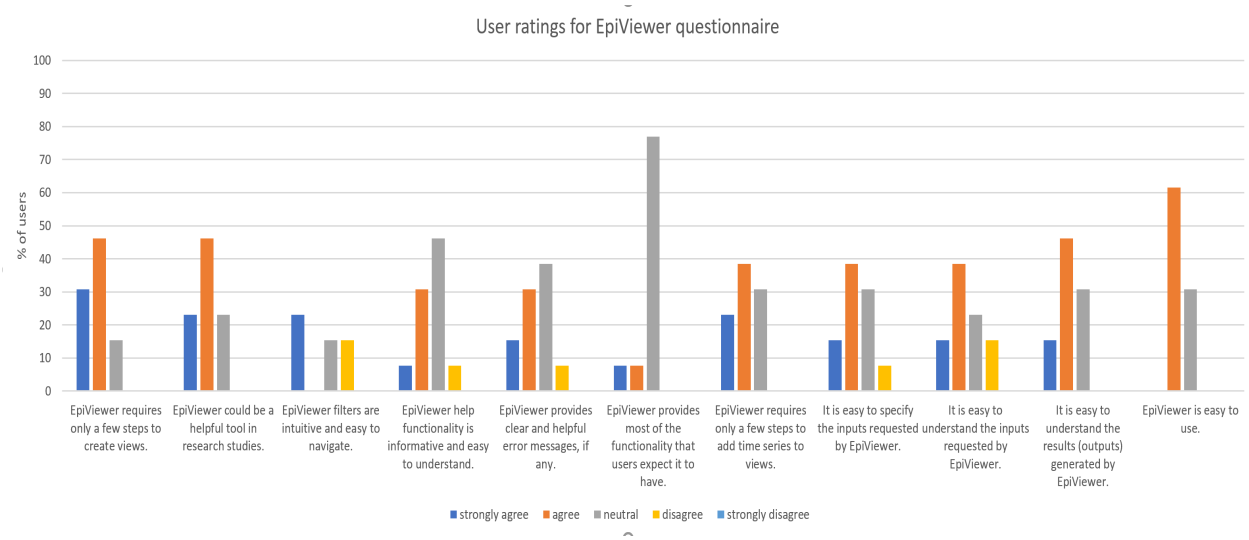

Figure 4.9: User ratings of various tasks performed on the user study.

Both quantitative and qualitative data were collected over the course of the study. Quantitative data included the start and end times recorded for each task so we could assess how intuitive the application is. Qualitative data included handwritten observation notes from the instructors documenting the sequence of actions users took to complete major tasks like importing and filtering data, along with problems they encountered in performing the tasks. Participants recorded overall user experience via an online survey, which included prompts for ease of use, problems faced, and open-ended questions like applications for this tool and recommendations for improvements. Refer to Figure 4.9 for a breakdown of the participants' reactions.

We observed that 80% of the users were able to use the application without difficulty (Refer to Figure 4.8). Important application functionalities like uploading data, user account creation, view creation and filtering were performed easily by users, and they were able to complete the tasks within the allotted time. They cited the ability for grouping datasets as views/workspaces and for visualizing data from different sources and attributes on one screen to be helpful features. Many felt that analysis and navigation of the time series were

simplified by the metadata filters and zoom functionality. The application was found to have a quick learning curve overall. Users also communicated interest in using the application in other research areas for analyzing data feeds.

The participants did indicate that they felt that the distinction between public and private views and time series was unclear. They also requested more detailed feedback messages after performing important tasks like uploading datasets, zooming, and resetting filters. These suggestions were used to guide enhancement decisions for improving EpiViewer's user interface. Detailed messages, as well as help text, are incorporated in the current version of EpiViewer. To refine the distinction between public and private time series in a view, different tabs have been introduced in the View page to segregate the time series belonging to private and public views, as well as those included in the current view.

4.3 Benefits of the system

One of the major benefits of EpiViewer is as a platform for sharing and comparing epidemic curves. Multiple articles[18, 27, 30] have highlighted the need for researchers to share data during a pandemic. Even when individuals and organizations are willing to share their data, the harder question they face is: how does one go about doing it? Simply putting it on a website does not facilitate comparison with data from other research organizations because there is no standard format for sharing that data. Usually, individual institutions publish their data in a format convenient to their specific systems. This makes re-usability that much harder, and ultimately leads to situations where the data is not shared effectively.

EpiViewer is a step towards addressing these challenges for temporal epidemiological datasets. With its easy-to-use interface for loading, publishing, and comparing data, EpiViewer is designed so that data from multiple parties can be shared and visualized in a straightforward

manner, and executive reports can be constructed in an expedited fashion. The implementation of Epi-features in EpiViewer allows users to evaluate the time series data from a statistical standpoint. Expert domain users can draw informed conclusions about the time series, especially in determining the quality of forecasts across different sources. The 'First Take-off Time and Value' Epi-features would normally require expert intuition to determine a threshold value for a given disease. This manual interference is eliminated by adopting the segmented regression approach.

EpiViewer was originally designed to be a lightweight, standalone web application, but has been enhanced to support easy integration within larger ecosystems; this is made possible because of the service-oriented computing style provided by the REST APIs to support interoperability of services.

Apart from epidemiology, timeseries visualization is used in many other domains like stock market, housing demand, energy usage, temperature, rainfall and so on. It will be possible to extend EpiViewer to visualize timeseries from these domains as well. This will require new domain entities and contexts for filters. Once this is designed it should be simple to integrate these UI frames into the application. Most of the existing REST Services will take care of the basic functionalities of the app.

4.4 Comparison with similar existing software systems

EpiViewer's simple and intuitive web interface offers a time-effective way for scientists to upload and visualize their time series data. It natively offers a variety of filtering mechanisms and display features to enhance visual analysis of the data. Although these features are available in Excel, R, SAS, or Matlab, EpiViewer's interface makes transitioning between different visualizations quick and seamless. These filtering mechanisms also enable

comparisons between different data types of different scales (deaths, hospitalizations, cases, etc.) so that trends, correlations or anomalies can be swiftly perceived across the dataset. In addition, the application's 'Movie' feature adds a temporal component to visualizing the data that would be harder to achieve in applications like Excel, R, SAS, or Matlab. The supplementary material contains a movie example.

The web-based platform also makes it easy for users to share their data with other scientists in a standard format. To achieve data sharing in R, Matlab or similar tools, the user has to write scripts to process the data, to visualize it, and then to share it via a CSV, PDF or image files. Even applications like Dotmapper [35] have reported the need for improved data sharing. Through EpiViewer's public views, researchers can make their data available to other scientists for download, either in CSV format or via Rest APIs (JSON input and output), or even create a snapshot of the system simply by clicking a button.

EpiViewer is already configurable for integration within larger analytic systems (like the BSVE); this, along with the built-in REST APIs, can be used to automate the process of loading data from other data sources, either from within or outside of the parent system. Although this functionality could be achieved with R, it would require the development of custom scripts and interfaces compatible with the target system. With EpiViewer, the user need not worry about the implementation details of the services and API.

Another noteworthy web application is FluSight [2, 38]. Forecasting teams that participated in the CDC Flu challenge created FluSight in 2017 as a tool to visualize the CDC surveillance flu data and the forecasts submitted by the different teams. Although the teams collaborated outside of the application to share their models and forecasting data, the interface provides filtering by HHS region to help the viewer understand how the teams modeled the Influenza progression at different time stamps through current and past seasons. However, while FluSight offers visualization features that are essential for exploring epidemiological datasets,

the website is built specifically for the U.S. CDC Flu Challenge, and the features are tailored to that challenge. Furthermore, users cannot upload and compare their own data within the system.

4.5 Limitations

The system exhibits a few limitations. First, the canvas area looks cluttered if a view contains more than thirty graphs. Second, the application currently supports only two chart types - line chart and bar chart. It does not support other visualization motifs such as chloropleth maps, social network graphs, or phylogenetic trees. The application does not support multi-views combining different visualization motifs which could help to analyze data more effectively.

Chapter 5

Conclusions

We present EpiViewer: a lightweight visualization framework for viewing and sharing, surveillance and forecast time series data. The framework facilitates exploring, comparing, filtering and organizing temporal datasets to allow researchers to conveniently manipulate time series through the use of meta-attribute tagging. Importantly, EpiViewer supports data sharing and computation of general epidemiological metrics for time series on the fly. Finally, EpiViewer can be configured to support easy integration within larger software systems. We believe that EpiViewer fills a particular niche in epidemic science.

Future plans for enhancing the application include:

Spatial view: A heat map view that colors a geographical map based on the severity of the outbreak across the subregions would add a spatial aspect to epidemic analysis in addition to the existing temporal aspect. Users could then better identify trends on a geographic scale, and also identify hot spots where applying interventions could curtail the epidemic.

Error measure metrics: In addition to the Epi-features, error measure metrics like Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) [36] can be calculated for forecasts once ground truth data is available. These metrics will quantify the error across

the duration of the forecasted time series into a single statistic, allowing the user to quantitatively understand the quality of the different forecasts.

Advanced graph association: Currently, only one-to-one relationships can be established between surveillance and forecast time series through the Associated Graph feature. In real life applications, there may be multiple forecasts associated with a particular surveillance curve; being able to visualize how surveillance-related forecasts differ may make them easier to compare visually.

EpiJSON support: EpiJSON is a proposed standard format for exchanging time series data between applications [21]. Integrating support for uploading and downloading data in EpiJSON format would be helpful for promoting adoption of EpiViewer in the epidemiological community.

Chapter 6

Miscellaneous information

Availability and requirements

Project name: EpiViewer

Project home page: http://epics.vbi.vt.edu/epiviewer/index.jsp

Operating system(s): Platform independent

Programming language: Java, Javascript, D3, and Oracle or PostgreSQL

Other requirements: Java 1.7.0 or higher, Tomcat 7.0 or higher

License: None required.

Any restrictions to use by non-academics: None.

List of Abbreviations

API: Application Programming Interface

BSVE: Biosurveillance Ecosystem

CDC: Centers for Disease Control and Prevention

CSV: Comma Separated Value, a flat-file format for exchanging data.

DTRA: Defense Threat Reduction Agency

HHS Regions: US Department of Health and Human Services Regions are groupings of states used for aggregating epidemic activity.

IRB: Institution Review Board

JSON: Javascript Object Notation

MoBS Lab: Laboratory for the Modeling of Biological Socio-technical Systems

NDSSL: Network Dynamics Simulation Science Laboratory

REST: Representational State Transfer.

SAS: Previously called "Statistical Analysis System", SAS is now the official name of this statistical software platform.

SPA: Single Page Application

SPSS Statistics: formerly known as "Statistical Package for the Social Sciences".

SQL: Structured Query Language

SSE: Sum of Squares Error.

URI: Uniform Resource Identifier

Declarations

Ethics approval and consent to participate

The user study was approved by the Virginia Tech Institution Review Board (IRB) under IRB Number 17-506, Protocol Title "EpiViewer and My4Sight User Evaluations" The protocol was originally approved on May 31, 2017, and the most recent amendment to this protocol

was approved on September 8, 2017, by Virginia Tech Institution Review Board (IRB) Chair, David M Moore.All participants were briefed on the objectives and procedures of the user study, and informed consent to participate was obtained in written form from all of the participants before the study began. None of the participants were under the age of 18 or came from a vulnerable population.

Appendices

Appendix A

Exercise for EpiViewer Focus Group

Overview:

An instructor will give a quick overview of EpiViewer: what it is, and what problem it was designed to solve. Users will then have the opportunity to try out the system and give us feedback. EpiViewer offers the following features.

1. EpiViewer makes it possible for users to view and share EpiCurve time series data plotted on a graph for analytical purposes. The system is preloaded with Influenza data from EpiCaster and Ebola data from various sources during the 2014 crisis in West Africa.

2. Users can create their own Views to group time series in logical ways; users can also edit or delete their views.

3. Users can make their views public in order to share their data with other researchers.

4. Users may upload time series data in CSV format into their private views, or include time series from other public or private views.

5. They may edit their own time series data or delete it from the system.

6. Filtering capabilities allow users to temporarily add and remove time series plots from the view display.

7. Users can zoom in on a specific date range instead of viewing the entire plot.

8. The Movie feature allows users to see the time series laid out on the view in sequential order to see how epidemic surveillance and forecast data have evolved over time.

9. Users can download the time series from a view in csv format.

10. A Snapshot feature allows images of the view to be downloaded and shared, or included in papers or other documents.

User exercises:

Please go to http://epics.vbi.vt.edu/EpiViewer/epiviewer.html to attempt the exercises outlined on the following pages. It would be helpful if you could record the start and end time for each exercise. You may ask questions, but one metric we are trying to track is the intuitiveness of the interface, so please only ask a question if you are truly stuck. When you complete the exercises, please go to https://epics.vbi.vt.edu/wisdmmturk/epiviewerFocusGroup/ for some final evaluation questions.

1. Create a user account and log in; if you want to remain anonymous, make up a name. Report start time and end time.

2. Load the Ebola 2014 public view. Report start time and end time.

3. Filter the data so only Liberia Forecast data from NDSSL is shown. Report start time and end time.

4. Use the zoom feature to limit the view to December 1, 2014 December 31, 2014. Report start time and end time.

5. Take a snapshot of the graph and save it to the drive. Report start time and end time.

6. Reset the zoom feature to show the full range of the Ebola View again. Report start time and end time.

7. Use the movie feature to see how NDSSLs forecasts changed over time. Report start time and end time.

8. Create your own Ebola view; include some of the public Ebola time series in the view. Report start time and end time.

9. Upload a time series into your Ebola view (time series data will be provided.) Report start time and end time.

10. Remove one of the public time series from your view. Report start time and end time.

11. Download the data from your view. Report start time and end time.

Appendix B

List of Questions for EpiViewer Focus Group Evaluation

B.1 Part 1: Demographic Information

1. Sex:

(a) Male (b) Female (c) Other

2. Area of Citizenship:

(a) North America (b) South America (c) Europe (d) Asia (e) Africa (f) Australia

3. Which of the following best represents your ethnic group?

(a) Caucasian

(b) Asian or Asian American

(c) Hispanic or Latino/Latina

(d) Bi- or Multi- Racial

(e) Black or African American

(f) Native American or Alaskan Native

(g) Native Hawaiian or other Pacific Islander

(h) Other

4. I am (please select one of the following options):

(a) an undergraduate student

(b) a masters student

(c) a doctoral student

(d) a postdoctoral associate

(e) a faculty member

(f) a researcher

(g) other:

5. Please indicate your main area of study and/or research:

6. Have you used any tool similar to EpiViewer prior to this evaluation? Yes/No.

If yes, please specify:

B.2 Part 2: User Evaluation

Below are a number of statements regarding your experience with EpiViewer. Please read each one and indicate to what extent you agree or disagree with each statement for questions

1-11.

1= Strongly agree 2= Agree 3= Neutral 4= Disagree 5= Strongly disagree

1. EpiViewer is easy to use.

2. EpiViewer requires only a few steps to create views.

3. EpiViewer requires only a few steps to add time series to views.

4. EpiViewer filters are intuitive and easy to navigate.

5. EpiViewer help functionality is informative and easy to understand.

6. EpiViewer provides most of the functionality that users expect it to have.

7. EpiViewer provides clear and helpful error messages, if any.

8. It is easy to understand the inputs requested by EpiViewer.

9. It is easy to specify the inputs requested by EpiViewer.

10. It is easy to understand the results (outputs) generated by EpiViewer.

11. EpiViewer could be a helpful tool in research studies.

12. How likely are you to use a tool like EpiViewer in your future research?

(a) Very likely

(b) Somewhat likely

(c) Not sure

(d) Somewhat unlikely

(e) Unlikely

13. What username did you use when applying for an account?

14. Did you encounter any errors while using EpiViewer? What were they?

15. What improvements could be made to EpiViewer to make it easier to use?

16. What new features would you like to see in EpiViewer?

17. Please share any additional comments you have about EpiViewer.

Bibliography

[1] Data for the 2014 ebola outbeak in west africa.

[2] Flusightnetwork.

[3] Healthmap project.

[4] Jerseyrest.

[5] NDSSL: Informatics resources for ebola epidemic response.

[6] Oraclerest.

[7] Situation reports: ebola response roadmap, world health organization (2016).

[8] Timesearcher.

[9] Neil Franklin Abernethy. *Automating Social Network Models for Tuberculosis Contact Investigation*. PhD thesis, Stanford, CA, USA, 2005. AAI3187255.

[10] K A Alexander, C E Sanderson, and M Marathe. What factors might have led to the emergence of Ebola in West Africa? *Tropical Diseases*, 2014.

[11] Wladimir J. Alonso and Benjamin JJ McCormick. Epipoi: A user-friendly analytical tool for the extraction and visualization of temporal parameters from epidemiological time series. *BMC Public Health*, 12(1):982, Nov 2012.

[12] McKenzie Andre, Kashef Ijaz, Jon D. Tillinghast, Valdis E. Krebs, Lois A. Diem, Beverly Metchock, Theresa Crisp, and Peter D. McElroy. Transmission network analysis to complement routine tuberculosis contact investigations. *American Journal of Public Health*, 97(3):470–477, 2007. PMID: 17018825.

[13] J. S. Brownstein, C. C. Freifeld, B. Y. Reis, and K. D. Mandl. Surveillance sans frontires: Internet-based emerging infectious disease intelligence and the HealthMap project. *PLoS Med 5(7): e151.*, 2008.

[14] J Brueckner. A switching regression analysis of urban population densities. *Journal of Urban Economics*, 1984.

[15] Lauren N. Carroll, Alan P. Au, Landon Todd Detwiler, Tsung chieh Fu, Ian S. Painter, and Neil F. Abernethy. Visualization and analytics tools for infectious disease epidemiology: A systematic review. *Journal of Biomedical Informatics*, 51(Supplement C):287 – 298, 2014.

[16] Vivek Charu, Scott Zeger, Julia Gog, Ottar N. Bjrnstad, Stephen Kissler, Lone Simonsen, Bryan T. Grenfell, and Ccile Viboud. Human mobility and the spatial transmission of influenza in the United States. *PLOS Computational Biology*, 2017.

[17] Jean-Paul Chrétien, Steven Riley, and Dylan B George. Mathematical modeling of the West Africa Ebola epidemic. *eLife*, 4:e09186, 2015.

[18] JP Chretien, D. Swedlow, I. Eckstrand, D. George, M. Johansson, R. Huffman, and A. Hebbeler. Advancing Epidemic Prediction and Forecasting: A New US Government Initiative. *Online Journal of Public Health Informatics*, 2015.

[19] Timothy Dasey, Hayley Davison Reynolds, Nancy Nurthen, Christopher Kiley, and John

Silva. Biosurveillance ecosystem (bsve) workflow analysis. *Online Journal of Public Health Informatics*, 5(1):e86, 2013.

[20] Cody Dunne, Michael Muller, Nicola Perra, and Mauro Martino. Vorograph: Visualization tools for epidemic analysis. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 255–258. ACM, 2015.

[21] Thomas J.R. Finnie, Andy South, Ana Bento, Ellie Sherrard-Smith, and Thibaut Jombart. EpiJSON: A unified data-format for epidemiology. *Epidemics*, 15(Supplement C):20 – 26, 2016.

[22] J Gaag and W Vijverberg. A Switching Regression Model for Wage Determinants in the Public and Private Sectors of a Developing Country. *The Review of Economics and Statistics*, 1988.

[23] Rebecca A. Hills, William B. Lober, and Ian S. Painter. Biosurveillance, case reporting, and decision support: Public health interactions with a health information exchange. In Daniel Zeng, Hsinchun Chen, Henry Rolka, and Bill Lober, editors, *Biosurveillance and Biosecurity : International Workshop, BioSecure 2008, Raleigh, NC, USA, December 2, 2008. Proceedings*, pages 10–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[24] Tom Koch and Kenneth Denike. Crediting his critics' concerns: Remaking John Snow's map of Broad Street cholera, 1854. *Social Science and Medicine*, 69(8):1246 – 1251, 2009.

[25] Ross Maciejewski, Philip Livengood, Stephen Rudolph, Timothy F. Collins, David S. Ebert, Robert T. Brigantic, Courtney D. Corley, George A. Muller, and Stephen W. Sanders. A pandemic influenza modeling and visualization tool. *J. Vis. Lang. Comput.*, 22(4):268–278, August 2011.

[26] J Manrique and K Ojah. The demand for housing in Spain: an endogenous switching regression analysis. *Applied Economics*, 2010.

[27] Martin I Meltzer, Charisma Y Atkins, Scott Santibanez, Barbara Knust, Brett W Petersen, Elizabeth D Ervin, Stuart T Nichol, Inger K Damon, Michael L Washington, Centers for Disease Control, and Prevention CDC. Estimating the future number of cases in the Ebola epidemic–Liberia and Sierra Leone, 2014-2015. *Morbidity and mortality weekly report. Surveillance summaries (Washington, D.C. : 2002)*, 63 Suppl 3:1–14, September 2014.

[28] Stefano Merler, Marco Ajelli, Laura Fumanelli, Marcelo F C Gomes, Ana Pastore y Piontti, Luca Rossi, Dennis L Chao, Ira M Longini, M Elizabeth Halloran, and Alessandro Vespignani. Spatiotemporal spread of the 2014 outbreak of ebola virus disease in liberia and the effectiveness of non-pharmaceutical interventions: a computational modelling analysis. *The Lancet Infectious Diseases*, 15(2):204–211, feb 2015.

[29] Wai-Ling Mui, Edward P. Argenta, Teresa Quitugua, and Christopher Kiley. Nbic and dtra, an interagency partnership to integrate analyst capabilities. *Online Journal of Public Health Informatics*, 9(1):e046, 2017.

[30] Stephen F. Schaffner Pardis C. Sabeti Nathan L. Yozwiak. Data sharing: Make outbreak research open access. *Nature*, 518.

[31] O. Ola and K. Sedig. The challenge of big data in public health: An opportunity for visual analytics. *Online Journal of Public Health Informatics*, 2014.

[32] H Ophem. A Modified Switching Regression Model for Earnings Differentials Between the Public and Private Sectors in the Netherlands. *The Review of Economics and Statistics*, 1993.

[33] Richard E. Quandt. The Estimation of the Parameters of a Linear Regression System Obeying Two Separate Regimes. *Journal of the American Statistical Association*, 1958.

[34] Caitlin Rivers, Eric Lofgren, Madhav Marathe, Stephen Eubank, and Bryan Lewis. Modeling the Impact of Interventions on an Epidemic of Ebola in Sierra Leone and Liberia. *PLoS currents*, 2014.

[35] Catherine M. Smith and Andrew C. Hayward. Dotmapper: an open source tool for creating interactive disease point maps. *BMC Infectious Diseases*, 16(1):145, Apr 2016.

[36] Farzaneh Sadat Tabataba, Prithwish Chakraborty, Naren Ramakrishnan, Srinivasan Venkatramanan, Jiangzhuo Chen, Bryan Lewis, and Madhav Marathe. A framework for evaluating epidemic forecasts. *BMC Infectious Diseases*, 17(1):345, May 2017.

[37] Swapna Thorve, Mandy L. Wilson, Bryan L. Lewis, Samarth Swarup, Anil Kumar S. Vullikanti, and Madhav V. Marathe. Epiviewer: an epidemiological application for exploring time series data. *BMC Bioinformatics*, 19(1):449, Nov 2018.

[38] Abhinav Tushar and Nicholas G Reich. flusight: interactive visualizations for infectious disease forecasts. *Journal of Open Source Software*, 2017.

[39] Cécile Viboud, Ottar N Bjornstad, David L Smith, Lone Simonsen, Mark A Miller, and Bryan T Grenfell. Synchrony, Waves, and Spatial Hierarchies in the Spread of Influenza. *Science (New York, NY)*, 312(5772):447–451, April 2006.

[40] Cécile Viboud, Martha I Nelson, Yi Tan, and Edward C Holmes. Contrasting the epidemiological and evolutionary dynamics of influenza spatial transmission. *Philosophical transactions of the Royal Society of London Series B, Biological sciences*, 368(1614):20120199, March 2013.