Integrating Predictive Analytics into a Spatiotemporal Epidemic Simulation

Chris Bryan *Student Member, IEEE*, Xue Wu, Susan Mniszewski, Kwan-Liu Ma *Member, IEEE*

Abstract—The Epidemic Simulation System (EpiSimS) is a scalable, complex modeling tool for analyzing disease within the United States. Due to its high input dimensionality, time requirements, and resource constraints, simulating over the entire parameter space is unfeasible. One solution is to take a granular sampling of the input space and use simpler predictive models (emulators) in between. The quality of the implemented emulator depends on many factors: its robustness, sophistication, configuration settings, and suitability to the input data. Visual analytics (VA) can be leveraged to provide guidance and understanding to the user. In this paper, we have implemented a novel VA interface and workflow for emulator building and use. We introduce a workflow to build emulators, make predictions, and then analyze the results. Our prediction process first predicts temporal time series, and uses these to derive predicted spatial densities. Integrated into the EpiSimS framework, we target users who are non-experts at statistical modeling. This approach allows for a high level of analysis into the state of the built emulators and their resultant predictions. We present our workflow, models and the associated VA system, and evaluate the overall utility with feedback from EpiSimS scientists.

Index Terms—Predictive Modeling, Visual Analytics, Epidemic Visualization, Spatial-Temporal Systems.

1 Introduction

Disease spread is a complex problem in today's globalized world. A number of factors influence a disease's impact, including transmission rate, incubation period, antiviral supplies, and population dynamics. Mosquito-borne illnesses such as chikungunya [19] have had recent outbreaks in North America, highlighting the need to understand the critical parameters in the spread and diffusion of these diseases.

Simulation is one way to do this. The Epidemic Simulation System (EpiSimS) is a scalable, complex, agent-based model for simulating the spread of infectious diseases within the United States [36]. A single EpiSimS run generates a large set of temporal, geospatial, and multivariate data, and requires significant compute and time resources. Because of this, it is impractical to test over the entire input parameter space. One solution is to take a discrete sampling of the input parameter space, and employ a simpler predictive model to fill in the gaps. Such "simulation of simulations" is sometimes called scientific emulation; the idea is that an emulator can map the high-dimensional inputs to outputs with a reasonable degree of confidence and without having to run the full simulation.

In this paper, we are studying the feasibility of visual analytics (VA) in an emulation workflow. Informally, the process is this: build an emulator, analyze the emulator, make a prediction, analyze the prediction. However, consider that a built emulator, defined as a predictive model, selected parameters, and a set of input data, can be good or bad at making accurate predictions. It's output and accuracy confidence can depend on many factors. Good VA helps inform the user whether the emulator is suitable for running predictions. If VA indicates aspects of the emulator have issues, such as outlier data points, the user can modify or select a different predictive model. When satisfied, the user makes predictions. Here VA lets the user analyze the prediction in relation to other runs in the dataset. In our view, the prediction itself is not the goal, but a place for more analysis.

In the context of EpiSimS, our motivation is to predict and analyze disease runs. The EpiSimS scientists, while experts at disease simulation, are not statistical modeling (or even visualization) experts, thus they are good candidates for this system. We employ two main panels: one for emulation and one for prediction, integrated into the existing EpiSimS viewer application, focused on using simplified visuals that combine color, comparison, and interactivity. From the emulator panel, users build and analyze an emulator. The prediction panel is used for making predictions, both temporally and spatially, and is where the user can review the prediction's place in both input parameter and spatiotemporal output space. Specifically, our contributions in this paper can be summarized as follows:

1. A workflow for non-statistical modeling experts to build emulators, run predictions, and analyze the outputs.
2. A set of models for predicting time-varying data, and a spatial predictor derived from output temporal predictions.
3. Novel visual interfaces for building and analyzing emulators and their predictions.
4. Integration into the visualization application for a large-scale, scientific simulation framework, specifically, EpiSimS.

2 Background and Related Work

There are two main facets to this work. Disease simulation and visualization is the first, and EpiSimS is our particular flavor of it. The second is predictive and parameter space analytics.

EpiSimS and Disease Visualization

The Epidemic Simulation System (EpiSimS) is a large-scale, discrete-event, agent-based model for simulating disease spread within the United States. It is highly customizable, scales to large population and geographic sizes, and can implement a number of response mechanisms for disease mitigation such as school closures, antiviral stockpiles, and behavior modifications. For prior EpiSimS work, see [24, 22, 23, 36, 25]. The work in this paper uses the chikungunya disease model from [25], although we augment that system by extending the front end viewer and introducing the emulation workflow.

More generally, visual analytics plays an important tool in the study of disease spread. The time-dependent geographic nature of disease makes maps an effective method of showing overall diffusion [17]. A recent systematic review [6] discusses many applications that utilize map views (and other techniques) for disease spread. A single geospatial mapping may be insufficient, however, to communicate the overall spatial evolution of a disease. Animation and small multiples [38]

- *Chris Bryan is with ViDi @ U.C. Davis: cjbryan@ucdavis.edu.*
- *Xue Wu is with ViDi @ U.C. Davis: xewu@ucdavis.edu.*
- *Susan Mniszewski is with Los Alamos National Laboratory: smm@lanl.gov.*
- *Kwan-Liu Ma is with ViDi @ U.C. Davis: ma@cs.ucdavis.edu.*

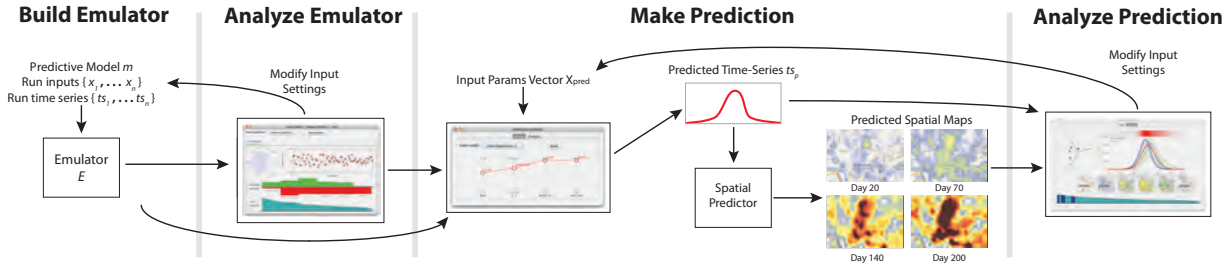


Fig. 1: The steps of our emulator and prediction workflow. First, build an emulator E by specificing input runs and parameters (x_i, ts_i) with a predictive model m. E can then be analyzed, or make predictions when given an input parameter vector X_{pred}. Predictions are a two-step process: time series first, then spatial snapshots for a set of days. The finished prediction can then be analyzed.

are two ways to show a disease's lifecycle. Each technique can offer advantages or drawbacks depending on the task: animation makes it easier to see overall trends [16], but small multiples may be better for focused analysis [35]. We use the small multiples technique as a component of our spatial prediction process (see Sections 4.2 and 5.2).

Maps can be linked with other views to form disease surveillance interfaces. Line charts, storyboard spreadsheets, heatmaps, flow maps, and even word clouds have been utilized recently [10, 1, 2, 28, 4]. Our interface combines map views, bar and line charts, and star coordinate plots to analyze the spatiotemporal disease predictions.

Visual Predictive and Parameter Space Analytics The second facet of our work concerns visual predictive and parameter space analytics. This includes support and validation of simulation models and analyzing behavior. A good example is [40], which decomposes high-dimensional input spaces via interpolation of parameter presets on a two-dimensional layout. [15] uses visual analytics to study the relationship between input parameters and simulation output time series. Like us, they generate a coarse sampling of input space, but they are focused on studying areas of output uncertainty using a heterogeneity metric. We are more interested in analyzing how a predicted run relates back to other runs in the dataset.

Many predictive applications are concerned with spatial predictions and their inherent uncertainty. Weather and climate models are notable [30, 32, 39], but other areas include topics like fishing grounds [31], bird populations [8], criminal activity [20], and even real-world disease trends [18].

Lastly, some works are concerned with building, validating and/or helping the user find optimum inputs for predictive models. Tuner [37] involves exploring the trade-offs of the high-dimensional input space to find "good" region segmentations in brain scans. [29] uses multiple linked views for the validation of regression models in the case of finding desired optimized parameters. Likewise, [3] is concerned with exploration of continuous parameter spaces for optimizing inputs. These papers differ from our work because we have no "optimum" disease setting, only finding what is realistic and possible (though optimizing mitigation strategies is another matter). Muhlbacher and Piringer introduce a framework for building and validating regression models in [26]. The focus of their work is on selecting subset features and then validating the quality of the built models. Our work is similar in that the user chooses input parameters to build on, but we focus more on the overall workflow of building an emulator, making predictions, and then analyzing those predictions.

Recently, [33] gave a survey of applications concerning visual parameter space analysis, and defines a conceptual framework for categorizing these types of applications. We formally discuss our place in this framework in Section 10 in the Appendix.

3 Our Approach and Workflow

Analyzing a scientific simulation like EpiSimS is challenging. One reason is scale: the data size is large, it takes a long time and lots of resources to run, and the complex raw output files must be post-processed before any analysis is attempted. Even for the relatively small population and location sets used in this study (simulating chikungunya on the Washington D.C. metro area, about 500,000 people), runs took up to an hour to complete. (Larger demographic sets can take hours and scale to hundreds of nodes.) Output files must then be parsed, indexed, and stored to a database.

One issue with emulating EpiSimS is its high input granularity. Individual geographic locations can have parameters such as "mosquito count," or various danger indexes. For example, locations classified as office buildings are usually set as safer from mosquitos than outdoor locations like parks. The parameter space quickly becomes overwhelming with so many dimensions and potential values.

What can be done is a discrete sampling of pertinent input parameters, using techniques such as Latin hypercube sampling [21]. From this set of input parameters, full runs can be done. After discussion with the EpiSimS team, we focused on a relatively small subset of the input parameter space and ran approximately 100 runs to generate a granular (if somewhat irregular) sampling space. Some parameters remained consistent for all people and locations (such as the *beta* value, or transmission index), some were location-specific (like *workIo*, which measures relative safety at work locations). This defined our input data and parameter space.

The scientists involved with EpiSimS are disease experts (sociologists and biologists) and simulation engineers. Despite some knowledge of statistical modeling, none were experts in regression model building and validation. This was an important factor to consider; we didn't want them to blindly trust any built emulator. To assist users, we designed a workflow that focused around straightforward steps: build an emulator, analyze the emulator, make a prediction, evaluate the prediction. These are noted in the headers of Figure 1.

More formally, the user first selects their desired input specifications and builds an emulator. These include input parameters to use, runs in the database, and a predictive model. The user can analyze the built emulator to see if particular bits are detrimental, such as existence of outlier runs, and rebuild the emulator if desired.

With a built emulator, the user makes predictions. This is a two-step process: first predicting the disease temporally, then predicting a set of spatial maps. At this point, we hit a potential roadblock: if there is no ground truth to compare the output to (i.e., the simulation hasn't been run yet), how does the user know if the prediction should be trusted? We use visual analytics to review how the predicted run compares to other existing runs. The user explores can explore a prediction's place in both the input parameter space and the output results.

4 Predictive Models and Methods

This section discusses the predictive models and methods we have implemented. To build an emulator, the user first loads a set of runs and parameters. These define the input data. EpiSimS runs are stored in two ways: as grids of density points that denote daily disease values, and as aggregated sets of time series that denote things like "infections per day" or "attack rate per day." We define an emulator E as:

$$E = (m, \{x_1, ...x_n\}, \{ts_1, ...ts_n\})$$

m is a temporal prediction model, each x_i is a run's input parameters, and each ts_i is the run's selected time series. To make a prediction P, a vector of input parameters (termed X_{pred}) is supplied. E first uses the model m to generate a predicted output time series, ts_p. This is used as input to the spatial predictor, which generates predicted disease map views, $\{map_1, ...map_z\}$. This is formalized as:

$$P = (E, X_{pred}, ts_p, \{map_1, ...map_z\})$$

E is the selected emulator used to build the prediction, X_{pred} the parameters to predict on, ts_p the predicted output time series, and $\{map_1, ...map_z\}$ are the z predicted spatial maps generated by our system. (z is chosen by the user, default is $z = 6$.) Figure 1 show's the steps of building an E and using it to create a P.

4.1 Temporal Predictive Models

We have implemented three initial predictive models (the m variable) to predict output time series (ts_p). Each is implemented as a set of R scripts, called by the main application. Our system is easily extensible; new models can be created and added with no front-end application changes. Currently, we have two regression-based models, and one nonparametric model. We give a brief overview here of each. To see the mathematical formalism of each, see Section 11 in the Appendix.

4.1.1 Simple Linear Regression

Linear regression [14] is based around the potential linear relationship between prediction parameters and run time series points, and is one of the most widely-used regression techniques. Because this model can only predict on a single point, we build a regression model for each day k of ts_p. If each day's model is defined as Y_k, then $m = \{Y_1, ...Y_n\}$ for n timesteps.

4.1.2 Stepwise Selection Model with Interactive Terms

Based on linear regression, this technique uses two- and three-way interaction of parameters to build an improved model m. The added terms drastically change the interpretation of all the coefficients in the model by including their relationships among the variables. Comparing all variable relationships produces an exponential number of interaction terms, though not all of these are necessary for construction. The process may even drop single terms if they are deemed insignificant for that timestep. Stepwise selection can go both forwards and backwards along the timesteps. By being based on the AIC score, the selection optimizes itself to pick the set of interaction terms to give the best model it can [12, 5]. Like linear regression, a different model is computed for each day, and m is the set of daily models.

4.1.3 Nonparametric Model

The two above models build ts_p by predicting individual days. This may lead to inadequate fitting, say if one or more days are high outliers. This could skew that part of the curve towards unrealistic behavior. A nonparametric model is an alternative to this, where the overall curve's function is predicted, as opposed to being pieced together by a set of predicted time steps. Our model defines three features to generate the curve: $\{f_1, f_2, f_3\}$. $f_1 = x(max(y_i))$ is the day that the run's peak occurs on. The peak's value is the second feature, $f_2 = max(y_i)$. f_3 determines the curve width, by using a single σ point (the normal practice). When a prediction is done, it returns the three curve features; the time series ts_p is then created based on the features.

4.2 Derived Spatial Prediction

EpiSimS simulates disease spread at individual locations for each time step, so at the most discrete level each geographic place could be thought of as a full time series. In this view, we could take the time series for each location and build a regression model. This would quickly grow unwieldy, as EpiSimS can scale to the entire United States (hundreds of millions of discrete locations). Even for the Washington D.C. runs in this paper, there are approximately 35,000 locations. Building a regression model with 100 runs for 300 timesteps already has over a billion data points if we regress on individual locations. Even using aggregated locations (where the data is stored at more granular resolutions for fast level-of-zoom queries), this is still unfeasible. We have therefore implemented a derived spatial predictor based on the predicted time series output. The pseudocode for this is in Section 12 in the Appendix. We give an overview of the spatial heuristic here.

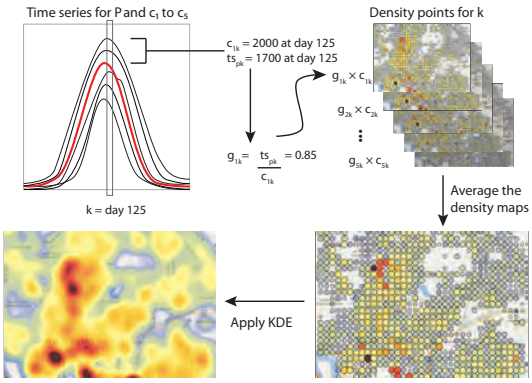


Fig. 2: To build a spatial prediction for a particular day k (in this case, $k = 125$) with $c = 5$, the time series of the five closest runs (denoted c_1 to c_5) to the prediction's time series (ts_p) are retrieved. The fractional value of each c_{ik} is $g_{ik} = ts_{pk}/c_{ik}$. This is applied to each c_{ik}'s density points, and all five of the scaled c_{ik} point maps are averaged. A KDE function [34] is applied to this averaged map to generate day's predicted density map.

First, the system must choose a set of z days on which to build predicted maps. These and are meant to be snapshots of the disease evolution, showing its lifecycle over time. To generate these days, we do the following segmentation process: First, we take the predicted time series ts_p and find the delta between each day's value. We perform a hierarchical clustering on this list of deltas, and then do a cut for the desired z number of clusters/segments (the default is six). These clusters have a start and end day, and define the predicted stages of the disease. For each stage, a single day is chosen as the "representative.". This is done by finding the day with a delta value that is closest-to-the-mean of all the deltas in its particular segment. We now have z snapshot days, and build a spatial prediction for each.

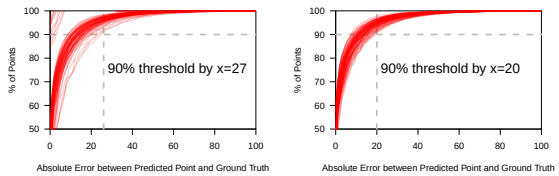
Figure 2 denotes the steps taken to generate a spatial prediction for an single day. First, the system finds a set of c runs with the closest overall time series to the predicted one (ts_p), by finding the Euclidean distance between ts_p and each run's time series. The default is $c = 5$ runs, but it's user editable.

For a single day k, the factional multiplier g_{ik} corresponds to each c_i run for that day. Dividing the ts_{pk} value by the c_{ik}'s time series value gives g_{ik}. Each g_{ik} applies to one c_i only, and a different g_{ik} is computed for each day k of c_i. We retrieve each c_{ik}'s spatial density (a grid of points stored in the database) and multiply each point by g_{ik} to get a scaled density map for the k day. The scaled maps are averaged together to generate map_k, which is the predicted set of disease points for that particular day. To get the spatial heat map for the day, we apply a KDE (with normal distribution) [34].

4.3 Analysis of the Spatial Predictor

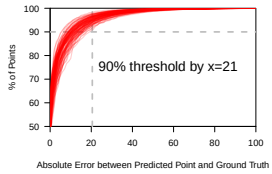
Our spatial prediction process works due to a tight interplay between the temporal time series of runs and their resultant spatial distributions. To validate our spatial predictor, we analyzed how accurately our process predicted existing runs in our dataset. If the existing runs could be predicted accurately, it would mean two things. 1) There is a tight correlation between temporal disease time series and spatial mappings. 2) If a temporal ts_p prediction is accurate, the predicted spatial maps $\{map_1, ...map_z\}$ would also be accurate.

We tested with $c = 1, 5, 10$, and 20. For each existing run in the

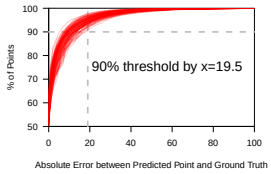


(a) c = 1

(b) c=5.



(c) c = 10.



(d) c = 20

Fig. 3: Accuracy of the spatial prediction process with $c = 1, 5, 10$, and 20. We note on each figure the 90% threshold, where 90% of predicted points fall within the **x** absolute difference from their real value. The system defaults to $c = 5$, a good balance of accuracy and speed.

database, we retrieved its "became infected per day" time series and produced the spatial predictions according to the above process (with $z = 6$). When finding the c closest runs to an existing run, we had to exclude the run itself from comparison (as the Euclidean distance of two identical time series would be zero). For each run, we now had a set of z predicted spatial maps, and could compare them to the actual (ground truth) maps for each day. For each density point, we computed the absolute error between prediction and real value. We combined the errors for all points into a single plot for each run and c value, mapping the overall percent of points by their error bound. For an individual run, we could then state a sentence like, "90% of all density points in run 15 with c = 5 are within an absolute error of 20." (Maximum disease point values can go above 500.) The runs were overlaid to a single plot for each c value, generating four aggregated plots in Figure 3.

Our system defaults to $c = 5$, which we feel is a good balance for accuracy and performance. At $c = 1$, some runs display very high accuracy (the upper left lines in Figure Figure 3a), this is because some runs in our dataset have similar positions in input parameter space and almost identical output time series. Since only one run is used to build the predicted snapshots, close similar runs with close spatial mappings lead to very high predicted accuracy. This goes away when increasing c to 5 because more runs are averaged in, but the overall 90% accuracy threshold is lowered. When c is too high, the time required to retrieve the density points for many runs causes the application to slow, without increasing (or decreasing) the overall accuracy. When performing this analysis, we created a spatial analysis panel so users could compare spatial predictions to other existing runs, see Section 5.3.

5 Visual Design

The existing EpiSimS viewer was originally designed for [25] as a browser-based tool, and afterwards updated to a Java application. Users create panels to analyze post-processed data. The three main views in this system are (1) parallel coordinates to show run input parameters, (2) line charts to show temporal outputs, and (3) maps to show disease spread for a particular day (using a KDE-based function [34]). Figure 11 in the Appendix shows these components.

We explicitly mention these because we utilize their ideas in the new workflow panels: the emulator panel, the prediction panel, and a third panel for analyzing spatial maps. For example, to build an emulator, the user first loads a set of runs into the input parameters. The set of runs comprises the input data and the user selects axes here

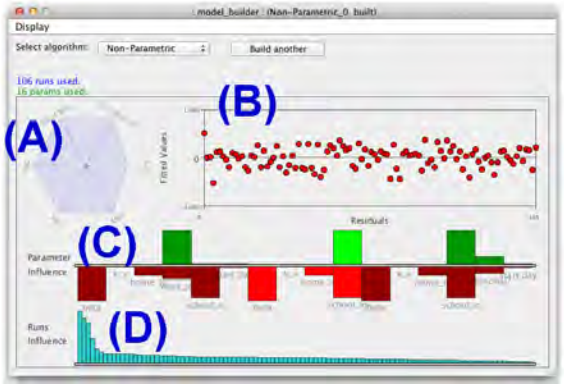


Fig. 4: The emulator panel includes (A) a radar plot for important numeric values, (B) error residuals, (C), parameter influence chart (showing importance to the model and potential error), and (D) run influences to the model.

to define the input parameters to use. We now describe the main new panels and their components.

5.1 Emulator Panel

This is where a user builds and analyzes emulators. First, the user selects the predictive model m and makes sure to have the desired runs loaded and input parameters selected. An emulator E is built, and its analytics are loaded. Figure 4 shows a built emulator. We visually display a number of traditional statistical metrics to inform the user about the state of the emulator's predictive model.

Radar Plot The radar plot (Figure 4A) contains a number of numeric calculations for showing the state of the emulator's predictive model: R^2, MSE, AIC and BIC, and the number of runs and parameters used by the model.

R^2, also known as the coefficient of determination, is a key output in predictive model analysis. It measures how close the fitted data is to the model. The closer the R^2 is to one, the better the model fits the data, so a higher value is desired. Relatedly, mean squared error (MSE) measures total averages of the squares of the errors. While R^2 is a standardized measure of degree of fit in the sample, MSE is an unbiased estimate of error variance. A lower MSE is desired.

AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are measures of the relative quality of a statistical model for its given set of data [5]. More parameters can improve a model's accuracy but also result in overfitting. This creates a tradeoff between goodness-of-fit and the model's complexity. BIC and AIC resolve this conflict by penalizing the number of parameters in the model. Lower BIC and AIC indicate the parameters had a better fit. For our purposes then, smaller AIC and BIC mean a better emulator. Usually, the AIC and BIC correspond to each other.

For linear regression, the number of parameters is the same as the selected input parameters. Stepwise selection includes interaction terms, and the nonparametric model includes all terms used to build the three features (features are differentiated by color hues).

The values in the radar plot scale relative to other built emulators. For example, a user can easily compare two emulators built from the same set of input data to see which has a higher R^2, or which uses more parameters.

Error Residuals The error residuals show the average error for each run in the built emulator (Figure 4B). Each run is assigned to a spot on the x-axis, and its corresponding residual value (its average error over all timesteps) is the y-value. Residuals should be closely banded around $y = 0$ for a tighter predicting emulator. If the residuals are distributed widely along the y-domain, or there are has many outliers, there is larger potential for error within the predictive model.

Parameter Metrics We show the parameter metrics with a two-sided bar chart (Figure 4C), where each bar represents one parameter. The top side, with green bars, denotes the influence of each parameter in building the model. We calculate this by taking the square of the inverse of the p-value (which helps show if the result is likely significant, i.e., caused by the independent value). The larger the bar's height, the more importance the parameter has when making predictions.

The red-colored bars on the bottom denote the corresponding standard error of coefficients for each parameter. This is used to measure the precision of the estimate of the coefficient, so the smaller the standard error, the more precise the estimate will be. A smaller value is desired here. Both bars are normalized over the set of parameters. The best case for a variable is high influence (large green bar) with corresponding low error potential (red bar). If all bars are the same (for either the top or bottom), it means that all parameters have equal weight for that metric. In cases where a single variable has both a higher green and red bar compared to other parameters, this means that the parameter is very influential in the resulting run, but has higher potential to cause imprecise predictions. Users should be cautious about this "outlier" parameter, and consider running full simulations in that part of input parameter space.

Run Influence We show the influence of each run in the bottom bar chart (Figure 4D), where the height is determined by calculating the square of its Cook's Distance [7]. This estimates the influence of each observation (each run) by measuring the effect of deleting it. A higher value indicates the runs plays a more influential role in the emulator. This can be good or bad: if only a few runs have high scores, they can overly distort the emulator's predictions, and may indicate the parameter space needs better sampling relative to those runs (i.e., performing full EpiSimS runs with input configurations close to this to better sample these areas of the parameter space). We normalize the runs and order them in descending value. This chart is copied in the prediction analysis panel where it is used as a navigation tool to help users explore runs in the dataset (see Figure 5E).

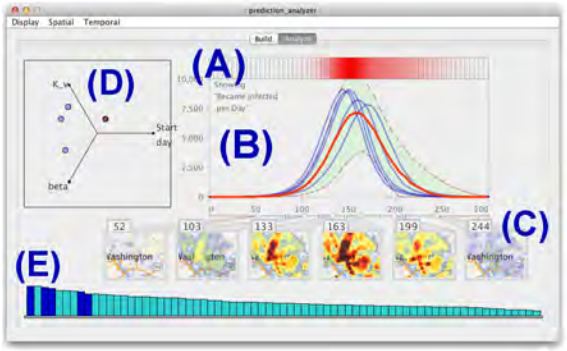


Fig. 5: The prediction panel allows analysis of a single prediction. It's main components are (A) temporal mean squared error bars, (B) time series lines for the prediction (red), other runs (blue), and confidence and prediction intervals (dotted lines and green bounds) (C) map snapshots based on time series segmentation, (D) a star coordinates plot for dimensional investigation of runs in the input parameter space, and (E) the run influence bar with closest runs highlighted dark blue.

5.2 Prediction Panel

Once an emulator E is built, the user loads a prediction panel and makes predictions. The prediction panel loads with a parallel coordinate tab showing available parameters for the selected emulator. The user sets the value of each parameter to use (this is the X_{pred} input vector for the model m), and the emulator generates a prediction P. Because the predictive model m is pre-built, the temporal prediction (ts_p) is very fast. The majority of the time is spent doing the spatial predictions ($\{map_1, ...map_z\}$). When the prediction is triggered, the panel switches to the analysis view (Figure 5). We implement a number of components here to assist analysis.

Time Series Plot and Mean Squared Error Band Figure 5A and B show the predicted output time series (ts_p) in a line chart, and above that a color band showing the daily mean squared error (MSE). The line chart initially loads both the ts_p (red color) and the time series of each closest c_i run (dark blue), showing the user how the closest runs temporally compare to the prediction. If desired, more runs can be loaded into the line chart for comparison. ts_p is bound with its 95% confidence intervals [27] and 95% prediction intervals [9], denoted as red dotted lines and a green bounding box, respectively. In the MSE band above, red represents a higher value, so these timesteps had more error in the emulator's training data, which indicates more potential for uncertainty on these days.

Map Snapshots To summarize the spatial evolution of the predicted run, we display the predicted spatial snapshots and segments as generated in Section 4.2 (Figure 5C). The segments are denoted on a track below the time series plot, and map snapshots and their respective days are below this. Maps can be dragged along their segment track to explore the spatial prediction at different days, but the idea is that the originally set snapshot day for each segment represents the general disease state within that time frame.

Star Coordinates Star coordinates [13] are a way to lay out multidimensional data points on a 2D plane (Figure 5D). Each input parameter is mapped to an axis, which can be dragged by the user. The prediction and any loaded runs are mapped as single points on the plane. Although a coarse layout for multi-dimensional data, star coordinates allow a user to quickly explore the relationships of points in dimensional space by interactively rearranging the axes. Using this with the time series view lets a user quickly analyze if runs with similar temporal outputs are related in input parameter space.

Run Influence Bar Chart We copy the run influences bar chart from the emulator panel, and place it to the bottom of the prediction panel (Figure 5E), as a navigation tool. Each c_i closest run is denoted with a darker blue color, and the user can toggle runs here to load their time series lines and star coordinate points for further analysis.

5.3 Spatial Analysis Panel

To assist with the analyzing spatial predictions (Section 4.3), we designed a map comparison panel (Figure 6). A user loads a base run into the top row, generates its z map snapshots, and compares it to a second run's map snapshots at the same days. To the left of the map snapshots, a line chart shows the time series of the two runs. A set of scatterplots on the bottom row show comparative statistical differences about the corresponding daily maps. A scatterplot to the bottom left aggregates the daily scatterplots into a single view.

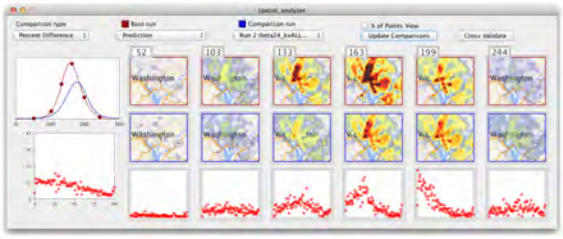
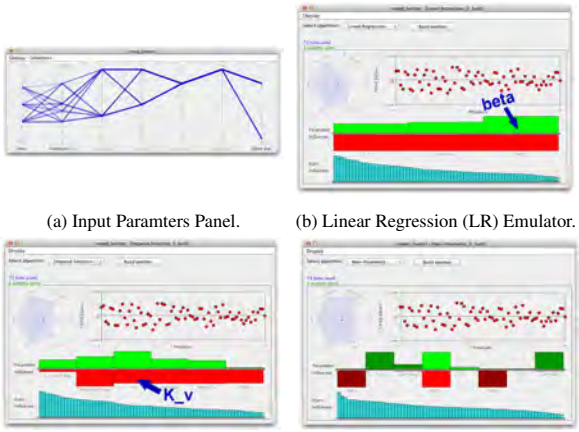


Fig. 6: The spatial analysis panel can be used to compare two runs (included predicted runs) to each other, both spatially and via scatterplots. The line chart at left shows the time series of the two runs and the left scatterplot aggregates the daily statistical views together.

6 Use Cases

We now walk through two scenarios as a potential user. We demoed to and then had the EpiSimS team walk through these to help familiarize them with the system. We discuss their initial feedback from this session and follow-up evaluations in Section 7.



(a) Input Paramters Panel. (b) Linear Regression (LR) Emulator.

(c) Stepwise Selection (SS) Emulator. (d) Nonparametric (NP) Emulator.

Fig. 7: The input parameters panel and the three built emulators for the initial use case. The runs are filtered to have a *Seasonal*=YES value, and other axes have been manually disabled. For the built emulators, the error residuals and run influences charts are similar, but the difference in the parameter influences is notable. For the LR, they have relatively smooth influences. The SS model uses interaction terms, and the NP model defines parameters across its three features.

6.1 Validating Good Emulators and Making Predictions

The user wants to build an emulator for predicting on three input parameters: *beta*, *K_v*, and a positive *(Seasonal) Start Day*. The user first loads all runs into the input parameters panel with a *Seasonal?*=YES value, and disables all input parameters besides the desired three (Figure 7a). Predictions made using only three parameters will still have a measure of uncertainty because the disabled parameters have variability, but the user can take this into account.

The user then builds three emulators, one for each predictive model from Section 4.1, termed LR (linear regression), SS (stepwise selection), and NP (nonparametric), see Figures 7b-d. Comparing the emulators, the user notes their similarities. The run influence and error residuals charts show no runs are skewed in either influence or error. Each emulator should give stable predictions.

The parameter influence bar charts of the runs are interesting. The LR model (Figure 7b) shows all three parameters as having equivalent potential error (red bar chart), meaning they each can equally introduce error into predictions. The top (green) bar chart shows *beta* is the most important parameter, *K_v* the least. This is different in the SS model (Figure 7c), which shows *K_v* has the highest parameter influence. The NP model has very different parameter influence bars (Figure 7d). For each feature, a different parameter is the most influential. *beta*, the most influential parameter for f_2, has red bar values for each of the three features, so it is the most widepsread potential source for error.

Finally, comparing the radar plots, the user sees that the LR model is built using of a smaller number of parameters. For this reason, and because the parameter influence chart is more stable than the other emulators, the user selects the LR emulator.

The user then creates a prediction using the LR model with the following input settings: *beta* = 0.26, *K_v* = 4000, and *Start Day* = 40. The resultant prediction panel is shown in Figure 5. The user notes that the closest runs in the time series chart (dark blue lines) were all ones that had values near the top in the run influence bar chart. This means that these runs that were close to the prediction played a relatively more important role in the emulator's model. Potentially, this means that the runs can cause overfitting, so perhaps more full EpiSimS runs near this point in the input parameter space should be run. The runs are also plotted in the star coordinates. Examining it, one run maps to the same three-dimensional mapping as the prediction (i.e., it has the same *K_v*, *beta*, and *Start Day* values). To see if any other runs map to this same position, the user loads all the runs in the panel, populating

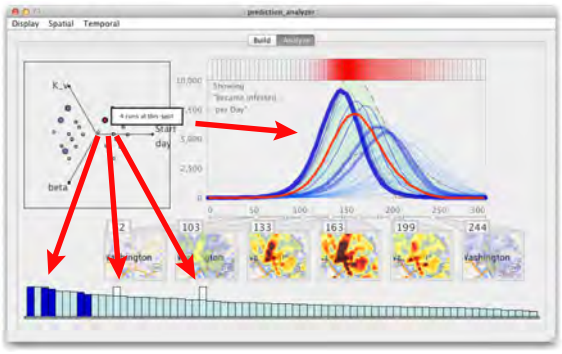


Fig. 8: Hovering over a set of points in the star coordinates plot highlights the runs in the corresponding bar chart and time series. We can see that the highlighted runs have different time series from the predicted run even though they match on the three input parameters that were used for the prediction.

the star coordinates and time series plots. Hovering over the prediction point on the star coordinates plot now (Figure 8) shows that four runs are at this point. These four points have the same *beta*, *K_v*, and *start day* values. To see the runs' full parameters, the user can load them into the input parameters panel and view them (not shown).

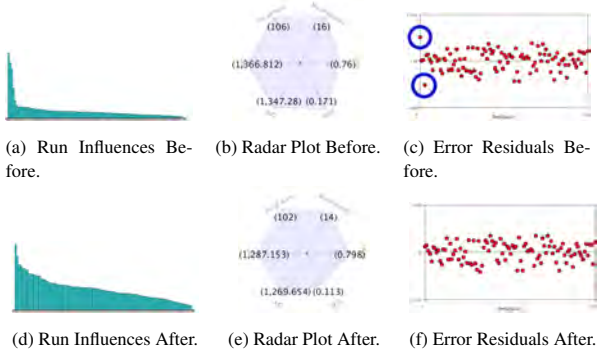
At this point the user is satisfied. The emulator will predict with good behavior based on the input data, and the prediction the user made was validated by analyzing it in relation to other runs in the prediction panel.

6.2 Analyzing Emulators in Detail

The user next wants to analyze emulator panels to determine if a set are worth using for predictions. For space, we only show analytic components for the nonparametric (NP) model, see the Appendix for all three emulators (they show similar results). The user first builds the emulators using all 106 runs in the dataset with all seven parameters (Figure 4 is the NP emulator). The pertinent analytic components of this NP panel are shown in Figure 9a-c. The run influences chart shows there is a small subset of runs (left side of Figure 9a) that have a large importance relative to the other runs in the model. There are usually two reasons for this:

1. These are outlier runs. A single outlier can cause misprediction, and potentially skew the model. Removing these runs, if they are indeed outliers, will improve the model.

2. These runs play a much more important role due to linearity of the parameters. This can happen if the input data is not fully



(a) Run Influences Before. (b) Radar Plot Before. (c) Error Residuals Before.

(d) Run Influences After. (e) Radar Plot After. (f) Error Residuals After.

Fig. 9: This shows how dropping four overly influential runs helps to improve an emulator: (a, d) The run influences are more stable, (b, e) MSE decreases and R^2 increases slightly, and (c, f) two outlier points (highlighted in the top plot) are removed.

ranked, which means there is insufficient data information to estimate the model. In this case, the first several runs play the important role, and the others are approximate linear combinations of these. Even with over 100 runs, most of the runs could not provide much information towards building the model.

To figure out which is the case, the user rebuilds the emulator after removing four high-influence runs and compares how the analytics have changed (Figure 9d-f). In the radar plot (Figure 9e), the R^2 value has increased slightly and the MSE value has decreased, both of which indicate an improved model. In other words, the removed runs' error residuals were outliers (the first reason in the list), and removing them made the emulator better.

The error residuals plot and the run chart also confirm this (Figure 9f and d). Two of the removed runs were outliers, and the run chart displays a much smooth distribution of influences.

Since the runs were outliers, it suggests two things. First, removing them improved the emulators. Second, these outlier runs represent areas of the parameter space that need to be simulated more. If the user ran full EpiSimS simulations close to the outlier runs in the parameter space, then future emulators could include these runs with less issue.

If the runs were linear combinations of each other (the second possibility), that would mean that despite the large number of runs there was little variability in parameter space. The information needed to build the model was insufficient with this dataset, so more varied EpiSimS runs should be done over the whole of parameter space. In either case, an ideal model building analysis should have a smooth run bar without any dominating runs. If this is the case, the R^2 and MSE values would not change too much when removing a single run from the input data.

7 User Feedback and Discussion

Ensuring that our workflow and components are intuitive and usable by the non-statistical experts of the EpiSimS team is a critical task. When we introduced the new components to the EpiSimS team, we did an initial demo of the new functionality and workflow using the two use cases, then supervised the users as they walked through the scenarios and played around with the system. This led to a first round of feedback during which many suggested changes to the system were made. We recommended that in the following weeks the scientists play with and use the new panels and workflow while we implemented some of the initially suggested changes. Both supervised (where we observed and helped users) and unsupervised usage took place. Also during this time, the spatial analysis panel (Section 5.3) was developed and added to the application codebase.

Feedback we received from users through discussion and email was (aside from bug submissions) mostly qualitative. In our initial demo, we had to define terms like R^2, so a Help overlay option was suggested. Users were initially concerned about the subtle interplay and understanding required of some metrics (such as analyzing the run influence chart in the second use case), and there was a small learning curve for users as they became accustomed to the system. However, users noted in follow-up evaluations that they were better able to understand the interface components as they became familiarized with the workflow. One noted that the visual display of statistical metrics (instead of raw numbers) helped simplify things. While our system did not completely alleviate the need to learn about the metrics, visualization eased the process. Users became proficient at being able to build and edit emulators, analyze for outliers and recognize parameter space gaps, and make and analyze predictions. As a general feature in EpiSims development, users stated the new emulation capabilities would be beneficial for future studies. Normally, only a small number of input parameters are tested, and only one parameter is changed and analyzed for each run. By being able to load a granular parameter sampling and use emulation in between, users told us that in future studies they will be able to test over a higher-dimensional parameter space.

Specifically regarding the emulator panel, one user remarked that it's good to easily see if runs are skewed to high influence, as they can quickly be verified as outliers. This helps determine if more varied full runs are needed to "fill in the gaps." Although they could see outlier runs for an emulator, and areas where predictions would probably perform poorly, one request was for an automated way to show and analyze parameter space gaps, such as in [15]. We are considering this a future work.

On the prediction panel, team members liked that they could analyze the predicted run both temporally, spatially, and in relation to other existing runs. One user noted that "comparing a potential run to existing similar runs for the time series is useful, helps to understand the effect of the parameter settings." The predicted map views were said to be beneficial on an intuitive level, a sort of "sneak peak" into a prospective run's spread. When the spatial analysis panel was made available, one user said the ability to spatially compare an existing run to the prediction was a "very cool analysis tool, being able to see how they will look visually as compared to a real run." Users also liked that they could compare only existing runs if desired, omitting the prediction altogether. "This is something we could really leverage for future analysis," and has led to discussions about expanding the spatial analysis panel's capabilities.

Regarding current system limitations and drawbacks, evaluations have touched on a number of areas. One thing users want to do is build emulators iteratively while assessing variable influences, like as is done in [26]. This would help users better validate that their chosen predictive model was more precise, and alleviate some parts of the subjective trial-and-error process currently performed when building emulators. Users also want a better system for highlighting correlations and significant results when they happen between panels.

The three predictive models we have implemented could also be more refined, and studied quantitatively to see how well suited they are for EpiSimS data. For example, in the nonparametric model's predicted time series, our curve function is equal for both sides. Although this works well for our particular EpiSimS chikungunya dataset, this is not always the case (see [23], where the time series can have multiple humps). We are considering more refined models (such as [11]), which can be easily integrated into our system and workflow, but this is an ongoing research topic. Similarly, our spatial predictor may not perform as well if the geographic disease spread is more stochastic and/or subtle. Future EpiSimS datasets will have to do accuracy analysis to ensure the spatial predictor is feasible. Finally, we note that our panels could have scalability issues on larger datasets containing thousands of runs over hundreds of parameters. For this dataset of seven parameters and 100 runs, it's not an issue, but too many runs and/or parameters would make the bar charts insufficient visualization techniques.

8 Conclusions

In this paper we describe a novel emulator workflow and visual analytics interface. We combine a number of statistical metrics, interactions, predictive models, a novel spatial predictor, and visualization techniques for emulator building, usage, and analysis.

Our approach utilizes main two panels: one for emulators and one for predictions. The emulator panel is useful for showing the overall state of a built emulator. The prediction panel is used for making and analyzing potential simulation runs, letting a user explore both output and input space. We also provide a spatial analytics view, to let a user geographically compare the predicted run to other runs in the dataset.

We validate our design through two use case scenarios and feedback from users. We also discuss and justify our spatial predictor.

Possible future work includes integrating feedback obtained from users, mostly concerned with helping users understand the large number of statistical metrics and their sometimes subtle relationships with each other, as well as giving users better analysis of input parameter space. We are also considering adding more predictive models to our system, and performing a quantitative study of their predictive ability.

Acknowledgments

This research has been sponsored in part by the U.S. National Science Foundation via grant NSF IIS-1320229, the U.S. Department of Energy through grants DE-FC02-12ER26072, and also LANL INGVA.

REFERENCES

[1] S. Afzal, R. Maciejewski, and D. Ebert. Visual analytics decision support environment for epidemic modeling and response evaluation. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 191–200, Oct 2011.

[2] L. ap Cenydd, R. Walker, S. Pop, H. Miles, C. Hughes, W. Teahan, and J. Roberts. epSpread - Storyboarding for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 311–312, Oct 2011.

[3] W. Berger, H. Piringer, P. Filzmoser, and E. Grller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, June 2011.

[4] W. V. Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC infectious diseases*, 11(1):37, 2011.

[5] K. P. Burnham and D. R. Anderson. Multimodel inference understanding AIC and BIC in model selection. *Sociological methods & research*, 33(2):261–304, 2004.

[6] L. N. Carroll, A. P. Au, L. T. Detwiler, T. chieh Fu, I. S. Painter, and N. F. Abernethy. Visualization and analytics tools for infectious disease epidemiology: A systematic review. *Journal of Biomedical Informatics*, 51:287–298, Oct 2014.

[7] R. D. Cook. Detection of influential observation in linear regression. *Technometrics*, pages 15–18, 1977.

[8] N. Ferreira, L. D. Lins, D. Fink, S. Kelling, C. Wood, J. Freire, and C. T. Silva. BirdVis: Visualizing and understanding bird populations. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2374–2383, 2011.

[9] S. Geisser. *Predictive Inference*, volume 55 of *Chapman & Hall/CRC Monographs on Statistics & Applied Probability*. Taylor & Francis, 1993.

[10] D. Guo. Visual analytics of spatial interaction patterns for pandemic decision support. *International Journal of Geographical Information Science*, 21(8):859–877, 2007.

[11] H. W. Hethcote, W. Wang, L. Han, and Z. Ma. A predator–prey model with infected prey. *Theoretical Population Biology*, 66(3):259–268, 2004.

[12] C. M. Hurvich and C.-L. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

[13] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, volume 650, pages 9–12, 2000.

[14] M. Kutner, C. Nachtsheim, and J. Neter. *Applied Linear Models- 4th Edition*. McGraw-Hill/Irwin, 2004.

[15] M. Luboschik, S. Rybacki, F. Haack, and H.-J. Schulz. Special section on uncertainty and parameter space analysis in visualization: Supporting the integrated visual analysis of input parameters and simulation trajectories. *Computers and Graphics*, 39:37–47, Apr. 2014.

[16] A. M. Maceachren, F. P. Boscoe, D. Haug, and L. W. Pickle. Geographic visualization: designing manipulable maps for exploring temporally varying georeferenced statistics. In *Proceedings of IEEE Information Visualization Symposium, Research Triangle*, pages 87–94, 1998.

[17] R. Maciejewski, P. Livengood, S. Rudolph, T. F. Collins, D. S. Ebert, R. T. Brigantic, C. D. Corley, G. A. Muller, and S. W. Sanders. A pandemic influenza modeling and visualization tool. *Journal of Visual Languages for Computing*, 22(4):268–278, 2011.

[18] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, and D. S. Ebert. A visual analytics approach to understanding spatiotemporal hotspots. *Visualization and Computer Graphics, IEEE Transactions on*, 16(2):205–220, Mar. 2010.

[19] D. Mackenzie. Threatwatch: chikungunya virus spreads in the americas. *New Scientist*, 220(2948), 2013.

[20] A. Malik, R. Maciejewski, S. Towers, S. McCullough, and D. Ebert. Proactive spatiotemporal resource allocation and predictive visual analytics for community policing and law enforcement. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1863–1872, Dec 2014.

[21] M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[22] S. Mniszewski, S. Y. DelValle, P. Stroud, J. Riese, and S. Sydoriak. Pandemic simulation of antivirals + school closures: buying time until strain-specific vaccine is available. *Computational and Mathematical Organization Theory*, 14(3):209–221, 2008.

[23] S. Mniszewski, S. Y. D. Valle, R. Priedhorsky, J. Hyman, and K. Hickman. Understanding the impact of face mask usage through epidemic simulation of large social networks. In *Theories and Simulations of Complex Social Systems*, pages 97–115. Springer, 2014.

[24] S. Mniszewski, S. Y. Del Valle, P. Stroud, J. Riese, and S. Sydoriak. Episims simulation of a multi-component strategy for pandemic influenza. In *Proceedings of the 2008 Spring simulation multiconference*, pages 556–563, 2008.

[25] S. M. Mniszewski, C. A. Manore, C. Bryan, S. Y. Del Valle, and D. Roberts. Towards a hybrid agent-based model for mosquito borne disease. In *Proceedings of the 2014 Summer Simulation Multiconference*, pages 10:1–10:8. Society for Computer Simulation International, 2014.

[26] T. Muhlbacher and H. Piringer. A partition-based framework for building and validating regression models. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):1962–1971, Dec 2013.

[27] J. Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937.

[28] J. Paparian, S. Brown, D. Burke, and J. Grefenstette. Fred navigator: An interactive system for visualizing results from large-scale epidemic simulations. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–5, Oct 2012.

[29] H. Piringer, W. Berger, and J. Krasser. Hypermoval: Interactive visual validation of regression models for real-time simulation. In *Computer Graphics Forum*, volume 29, pages 983–992, 2010.

[30] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *Data Mining Workshops, 2009. IEEE International Conference on*, pages 233–240, Dec 2009.

[31] N. Sakamoto, T. Uenaka, and K. Koyamada. Visual analysis of habitat suitability index model for predicting the locations of fishing grounds. In *Pacific Visualization Symposium, 2014 IEEE*, pages 306–310, Los Alamitos, CA, USA, 2014. IEEE Computer Society.

[32] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1421–1430, Nov 2010.

[33] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Moller. Visual parameter space analysis: A conceptual framework. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2161–2170, Dec 2014.

[34] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[35] T. Slocum, R. Sluter, F. Kessler, and S. Yoder. A qualitative evaluation of MapTime, a program for exploring spatiotemporal point data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 39:43–68, 2004.

[36] P. Stroud, S. Del Valle, S. Sydoriak, J. Riese, and S. Mniszewski. Spatial dynamics of pandemic influenza in a massive artificial society. *Journal of Artificial Societies and Social Simulation*, 10(4):9, 2007.

[37] T. Torsney-Weir, A. Saad, T. Moller, H.-C. Hege, B. Weber, J. Verbavatz, and S. Bergner. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):1892–1901, Dec 2011.

[38] E. Tufte. *The visual display of quantitative information*. Number v. 914 in The Visual Display of Quantitative Information. Graphics Press, 1983.

[39] A. Unger, S. Schulte, V. Klemann, and D. Dransch. A visual analysis concept for the validation of geoscientific simulation models. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2216–2225, Dec 2012.

[40] J. J. van Wijk and C. W. A. M. van Overveld. Preset based interaction with high dimensional parameter spaces. In *Data Visualization: The State of the Art*, pages 391–406, 2003.