9 Appendix

This supplemental document contains discussion and figures that due to space constraints, could not fit into the main paper. This is intended to assist with reader understanding by providing more formal discussion of topics, and to give images that either could not fit into the main paper, or were possibly too small to easily see.

10 Categorizing our Approach in the Visual Parameter Space Analysis Framework

In 33, Sedlamair et al. define a conceptual framework for analysis of visual parameter space. This framework is for pertinent applications to describe, discuss, and, evaluate visual parameter space applications (including predictive applications). They define three main components to their framework: 1) a data flow model for describing data generation, derivation, and prediction, 2) a set of navigation strategies, and 3) typical analysis tasks for validating parameter analytics. We now summarize our application in the context this framework and its key terms.

1) Data flow model For generating our input space, we use a set of EpiSimS runs as our *sampling* space. This input is very coarse, as it only consists of approximately 100 runs. The runs direct outputs are multi-variate and highly dimensional, so we use *derivation* to generate aggregated statistical measures for each run. *Prediction*, as can be guessed by the title and theme of this paper, is also implemented.

2) Navigation strategies We employ a *global-to-local* navigation strategy. Taking our large number of sample points, we build a model that serves as an overview of our dataset. Performing predictions investigates individual parameter configurations, to see how the output time series and spatial mapping. Although *simulation steering* might seem a natural fit for our simulation, this is unfeasible for EpiSimS, due to the long run- and post-processing time required for each run. Instead, we allow an author to export the particular input parameter configurations as a script for running EpiSimS.

3) Analysis Tasks Sedlamair et al. also describe a number of potential analysis tasks that can be explored, and we use some of them for our use case examples in Section 6. Specifically, we are concerned with the ideas of *sensitivity* in the built models and predictions, as well as *outliers* of various runs and input parameters in EpiSimS.

11 Formalism for the Predictive Models

This section mathematically defines the predictive model from Section 4.1.

11.1 Simple Linear Regression

Formally, we define a linear regression model as:

$$Y = X\beta$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,p} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,p} \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

where n is the number of runs of input data set, p is the number of parameters, and β is estimated by $\hat{\beta}$, given by:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

If we define X_{pred} as a vector of input parameters to predict on, and we want to get the fitted value $\hat{Y}$ (the output time series value), we use the following formula:

$$\hat{Y} = X_{pred}\hat{\beta}$$

Note that this $\hat{Y}$ only gives the predicted output for a single timestep. This means we need to calculate the regression for each timestep independently. Therefore, m, which represents the overall predictive model, can be represented by Y_i for each i day.

11.2 Stepwise Selection Model with Interactive Terms

Formally, the stepwise selection linear model still has the same expression as the linear regression model, but with different X, defined as:

$$y_i = \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \beta_{p+1} x_{i,j} x_{i,k}$$

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} & x_{1,j}x_{1,k} & \cdots \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} & x_{2,j}x_{2,k} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} & x_{n,j}x_{n,k} & \cdots \end{pmatrix}$$

where $j \neq k$, j and k are between 1 to n. The model requires increasing the number of β_is the same as with the number of columns in the X matrix. $\beta_{p+1} x_{ia} x_{ib}$ are interaction terms.

11.3 Nonparametric Model

For a nonparametric model, the overall curve's shape is generated; each day's individual value is derived from the curve's structure. This is in contrast to the regression models above, which predict individual timesteps, and then the output curve is pieced together. For our model, we fit the curve as a generated bell curve. We define three features of the curve: f_1, f_2, and f_3, whose value can be derived from each input time series points y.

$$y = exp(-\frac{(x-f_1)^2}{2f_3^2}) * f_2$$

$f_1 = x(max(y_i))$ is the day that the run's peak occurs on. The peak's value is the second feature, $f_2 = max(y_i)$. This means the peak point for a time series y has the coordinates $(f_1(y), f_2(y))$. f_3 determines how wide the bell curve is. This is usually done using a single σ point, which is what we do as well.

Each input run can be described by its three features. We take this feature-described run set and then build a single linear regression model using the features and their corresponding input parameter vectors (β in the linear regression model). When given an input parameter vector (X_{pred}), the model can simply output the predicted f_1, f_2, and f_3 and generate the curve. To see what the outputted value is for a particular day, simply find that x-spot on the curve and find it's y-value.

Figure 10 references one issue with nonparametric models, as they pertain to regression. The confidence and prediction intervals are not temporally bounded to the predicted day. Instead, they are defined as two trace curves around the predicted curve. The peaks of these two curves form the opposite edges of a bounding box that contains the actual predicted peak. To get a usable bounding curve for our visualization, we freeze the f_2 bound and slide the f_1 bounds together until they orient over the top of the predicted trace curve.

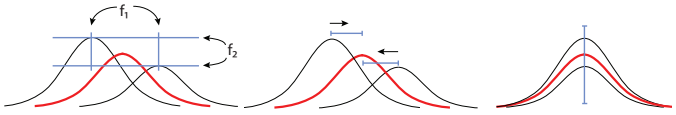


Fig. 10: The confidence and prediction intervals for the nonparametric model define a bounding box that the predicted curve's peak will be within (i.e, the f_1 and f_2 will be within these bounds). To get a suitable interval bounding for our viewer, we freeze the f_2 feature, and slide the f_1 feature along the x-axis until it is centered at the peak day of the predicted time series.

12 Formalism of the Spatial Prediction Algorithm

This section presents the pseudocode process for performing a spatial prediction from Section 4.2. There are three main steps in this: 1) Find the z representative snapshot days to use (the default is $z = 6$). 2) Find the c closest runs (the default is $c = 5$). 3) For each representative

day, build the predictive spatial map as a set of geospatial points. The points are returned, and a KDE converts them into a heat map view for the front end viewer.

```
ts_p     ← prediction's time series
deltas   ← empty list with length = length(ts_p)
z        ← nubmer of segments desired

# build deltas list
for timestep k from 1 to length(ts_p) {
    deltas(k) ← |ts_p(k) − ts_p(k+1)|
    # deltas(k) ← √deltas(k) if desired
}

# initialize segments
segments ← list of size length(ts_p) with
                segments(k).delta ←delta(k),
                segments(k).start ←k,
                segments(k).end ←k

# hierarchically cluster segments to size = z
while(length(segments)>z) {
    k ← position in segments with smallest delta
    segment(k).end ←segments(k+1).end
    segment(k).delta ←segments(k).delta +
                      segments(k+1).delta
    remove segments(k+1) from list
}

# find the day to use for each segment
for timestep k from 1 to z
    s ← segments(k)
    s.dayToUse ←day of the delta that's closest
                to the mean of all deltas from
                deltas(s.start) to deltas(s.end)

# generate the spatial prediction maps
c_runs ←list of c closest runs to ts_p
        based on Euclidean distance
maps ←empty list with length = z
for each day k in segments.dayToUse
    val_pred ←ts_p(k)
    for each c_i in c_runs
        val_c ←c_i time series value at day k
        g ←val_pred/val_c
        c_map(k) ←density points for c_i at day k
        for each point pt in c_map(k)
            pt ←pt∗g
    maps_k ←average of each c_i's c_map(k)

return maps
# apply KDE to each map_k to convert to heat map
```

13 Additional Figures

This section displays enlarged and additional figures from the paper.

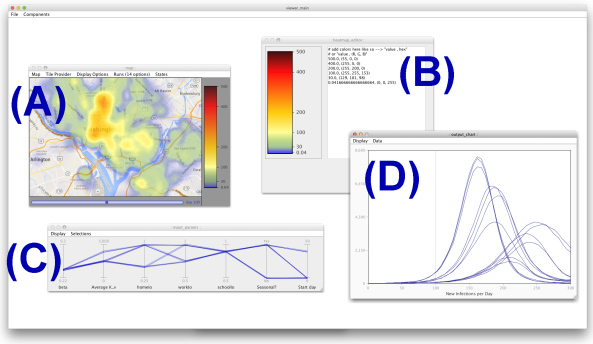


Fig. 11: This shows standard (pre-existing) EpiSimS components. (A) daily map view, (B) editor for setting for map layer colors, (C) input parameters panel, and (D) output time series line chart.

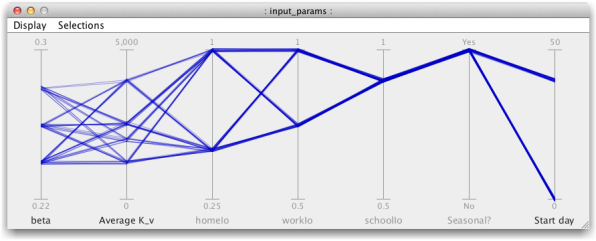


Fig. 12: This is the input parameters selection for Use Case One. The user enables the *beta*, *K_v*, and *Start day* axes.

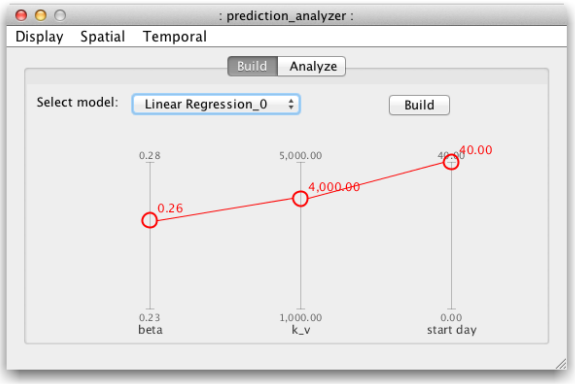
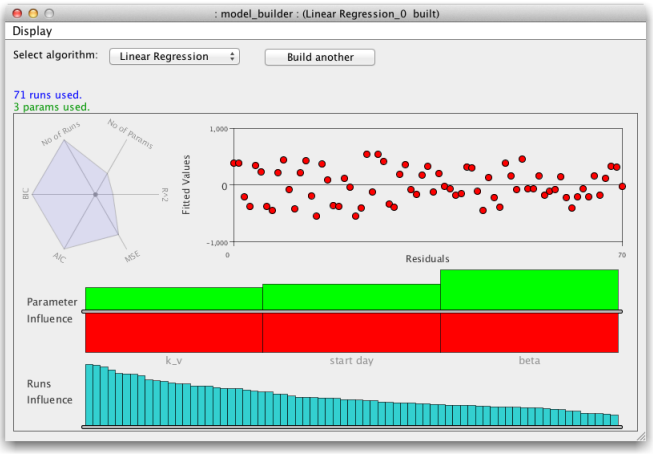
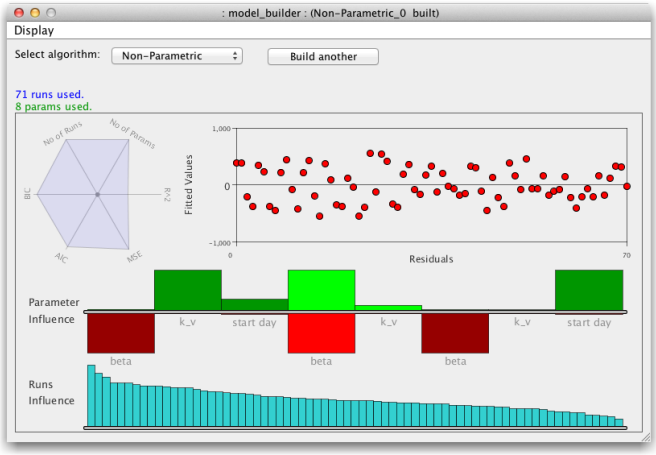


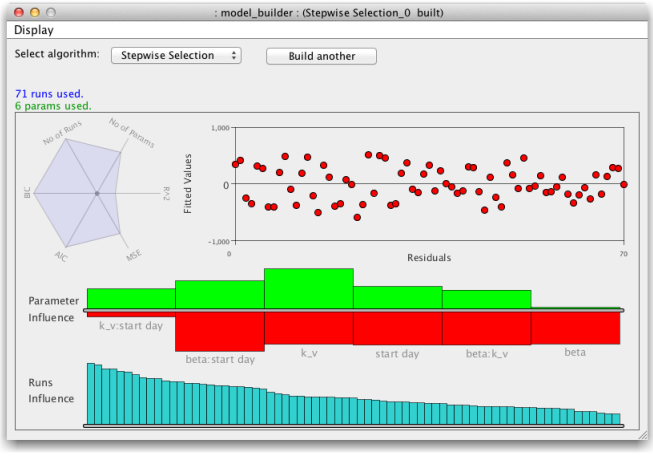
Fig. 13: When a user makes a prediction using a built emulator, the prediction panel initially opens with a parallel coordinates view. The desired emulator is selected from the drop down, and the user chooses predicted values by dragging the axes.



(a) Linear regression (LR) emulator

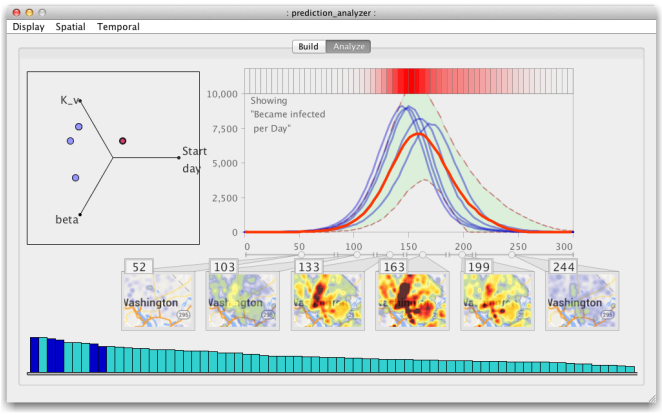


(b) Nonparametric (NP) emulator

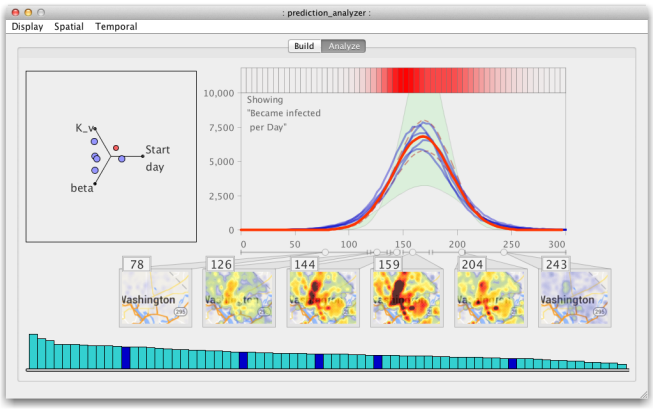


(c) Stepwise Selection (SS) emulator

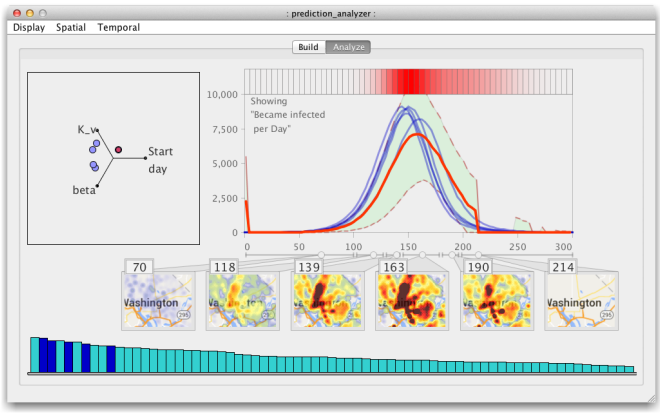
Fig. 14: These are the three built emulators for Use Case One.



(a) Linear regression (LR) prediction



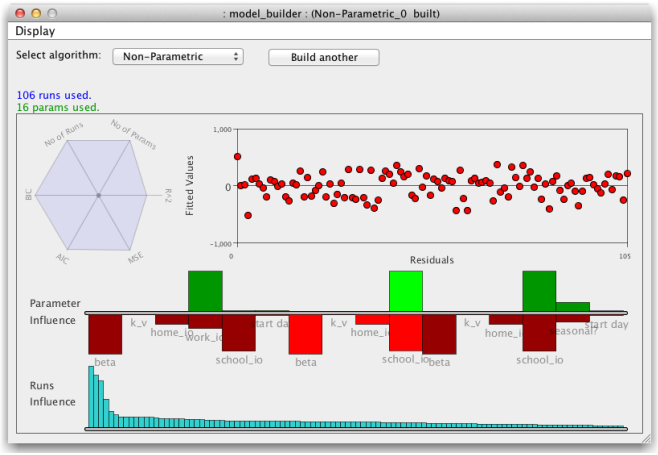
(b) Nonparametric (NP) prediction



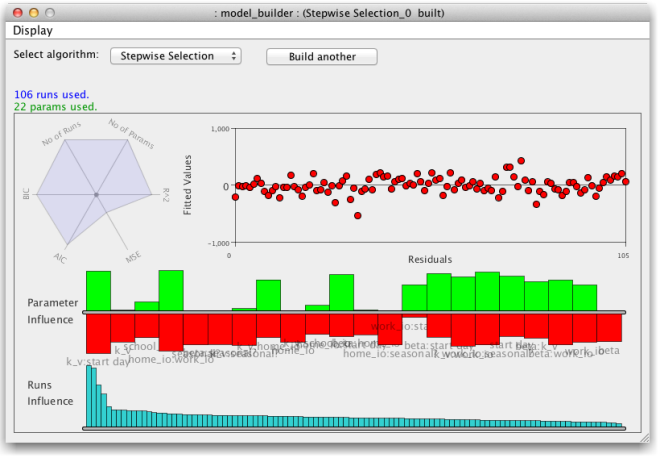
(c) Stepwise Selection (SS) prediction

Fig. 15: These are the predictions from the three built emulators in Use Case One, with input parameters set according to Figure 13. Notice that because their predicted output time series are not the same, the sets of closest runs are also different. This affects the spatial predictions (the map views).

(a) Linear regression (LR) emulator with 106 runs

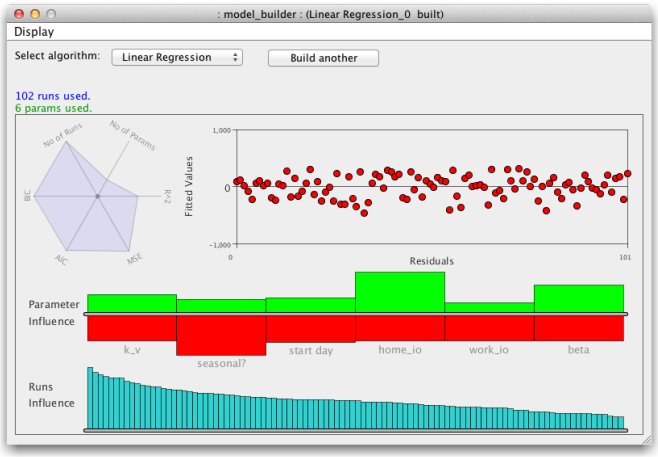


(b) Nonparametric (NP) emulator with 106 runs



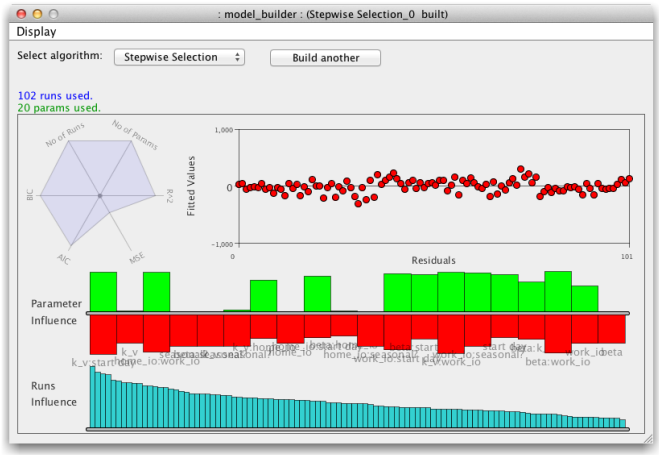
(c) Stepwise Selection (SS) emulator with 106 runs

Fig. 16: These are the three built emulators for Use Case Two, which were built from 106 runs and seven input parameters. Note that each model has error residual outliers and a set of runs that dominate the run influences bar chart.



(a) Linear regression (LR) emulator with 102 runs

(b) Nonparametric (NP) emulator with 102 runs



(c) Stepwise Selection (SS) emulator with 102 runs

Fig. 17: Removing the four highest ranked runs (in the run influences chart) improved each model considerably (and dropped the schoolIo parameter due to linearity). Error residual outliers are removed, the run influences bar charts are more stablized. In the NP model, the MSE and R^2 values are both improved considerably (in the LR and SS emulators these also improve, but the change is small).