2014-01

Data-driven simulation of complex multidimensional time series

Schruben, Lee W.; Singham, Dashi I.

Association of Computing Machinery

L.W. Schruben, D.I. Singham, "Data-driven simulation of complex multidimensional time series," ACM Transactions on Modeling and Computer Simulation, v.24, no.1 Article 5 (January 2014), 13 pages.
https://hdl.handle.net/10945/39559

Data-Driven Simulation of Complex Multidimensional Time Series

LEE W. SCHRUBEN, University of California, Berkeley
DASHI I. SINGHAM, Naval Postgraduate School

This article introduces a new framework for resampling general time series data. The approach, inspired by computer agent flocking algorithms, can be used to generate inputs to complex simulation models or for generating pseudo-replications of expensive simulation outputs. The method has the flexibility to enable replicated sensitivity analysis for trace-driven simulation, which is critical for risk assessment. The article includes two simple implementations to illustrate the approach. These implementations are applied to nonstationary and state-dependent multivariate time series. Examples using emergency department data are presented.

Categories and Subject Descriptors: General [**I.6.0**] Model Validation and Analysis [**I.6.4**]; Types of Simulation: Discrete Event [**I.6.8**]

General Terms: Data Modeling, Sensitivity Analysis

Additional Key Words and Phrases: Flocking algorithms

ACM Reference Format:
Lee W. Schruben and Dashi I. Singham. 2014. Data-driven simulation of complex multidimensional time series. ACM Trans. Model. Comput. Simul. 24, 1, Article 5 (January 2014), 13 pages.
DOI: http://dx.doi.org/10.1145/2553082

1. INTRODUCTION

Advantages of simulation include the ability to model complex systems with few unverifiable assumptions, assess decision robustness using sensitivity analysis, and quantify risk by running replications. Parametric statistical methods for generating multivariate time series (see Biller [2009]) include random noise components for simulating replications. Sensitivity analysis can be done by changing the values of the model parameters. These methods rely on statistical assumptions about the data to select and fit mathematical models. Parametric models cannot be used for some types of data because they may require unrealistic assumptions. Because the data is only one example of the past, this single sample of reality cannot be used to test the model without making additional assumptions. Data-driven methods include multivariate time series resampling such as block bootstrapping [Härdle et al. 2003]. These methods allow restricted replication and sensitivity analysis by changing the data sequence or adding noise, and rely on assumptions concerning dependency and stationarity. Time series bootstrapping can be combined with statistical models if one can justify making additional assumptions about the data.

Trace simulations are purely data driven. They simply use real data as the inputs for running a simulation. This is attractive because no assumptions about the data must be

Authors' addresses: L. W. Schruben, 4141 Etcheverry Hall, University of California, Berkeley, CA 94720-1777; email: lees@berkeely.edu; D. I. Singham, 1411 Cunningham Road, Naval Postgraduate School, Monterey, CA 93940; email: dsingham@nps.edu.

DOI: http://dx.doi.org/10.1145/2553082

made. However, trace simulation cannot easily be used for either sensitivity analysis or replication. Rare events that happened in the past are not rare in the simulated future; they will recur in every run. More importantly, it is impossible for any new important events to occur, potentially underestimating risk. All these methods assume in some sense that the future will tend to behave like the past. Trace simulations assume the future is *exactly* like the past (regardless of any changes in the system being simulated). Sensitivity analysis to the future being different from the past is important.

Our objective is to introduce a framework for general resampling from complex data where parametric models or bootstrapping methods are not appropriate. Purely data-driven trace simulation is a special case. This method enables replicated sensitivity analysis to explore how much the future will need to differ from the past before one might want to change his or her system.

The method in this article is inspired by flocking agents, called boids, popular in computer animations and the study of emergent social behaviors (Reynolds [1987] introduced the concept of boids). Although data resampling is completely different from previous flocking agent applications, we use some of the concepts and terminology to provide intuition. The basic notion is analogous to scripted agent flocking where a flock of boids follows the path of one or more leaders. Scripted boids are widely used in animations of flocks of birds, schools of fish, or crowds of people. Unless true replications of the data are available, there will only be a single leader boid whose path is the real data. We refer to the real data path as the *alpha boid*. Intuitively, imagine an n-dimensional time series of data as the coordinates of the flight path for a bird in n-dimensional space; this is the path of the alpha boid. If we wish to resample this data k times, we create a flock of k agent boids who swarm around this alpha boid. We then use the coordinates of the locations of each member in the flock as the values of k n-dimensional simulated times series.

To enable replication and sensitivity analysis, we need a way to control the degree that the simulated flock tends to swarm around the alpha boid. We do this by changing two classes of parameters. *Affinity* parameters control how closely members of the flock (different futures) want to swarm around the alpha boid (the past). We are quantifying the degree of affinity (or repulsion) the flock has to the alpha boid. There are also random *noise* parameters that model information or innovations in the future that cannot be fully determined from the past data.

In sensitivity analysis, we vary affinity parameters to control how much a future replicate tends to behave like the past. We vary noise parameters to model what cannot be known about a future replicate from its affinity to the past. We present two implementations of this method, using a single affinity parameter, λ, and additive white noise with variance σ^2. The affinity parameter in these algorithms is similar in spirit to the correlation between a future and the past. The closer the affinity is to one, the more a future tends to behave like the past. The lower the affinity, the more the future will behave differently from the past. When affinity is negative, the replicated futures are repulsed by the past.

Schruben and Singham [2010] proposed the idea of using flocking algorithms to simulate data that has similar properties to a stream of multivariate-dependent data. This concept has been used for modeling simulation input and for generating pseudo-replications of expensive simulation output [Schruben and Singham 2011]. This article presents easily implementable algorithms as examples and demonstrates how these can be used for replication and sensitivity analysis of multidimensional data for complex systems. Input to a trace simulation can be replicated to enable sensitivity analysis for a trace-driven simulation. Output of a simulation can be replicated to test the sensitivity of a decision to the simulation output if the cost of running the simulation is high.

First, we present an overview of current methodology and a motivation for the approach using emergency department data. The general framework is defined in

Section 2 and two specific implementations are presented in Section 3. In Section 4, the two algorithms are applied to generate arrival processes from the emergency department data and the results are compared to a simple nonhomogeneous Poisson process simulation. We conclude the article and provide some suggestions for future research in Section 5.

2. EXAMPLES OF FLOCKING FOR DATA MODELING

Hospital admission processes provide the context for this article. Patient arrivals to hospitals come in two general categories, scheduled and unscheduled. It is the unscheduled demand processes that need to be generated to drive a simulation run. Modeling admissions is a major problem, but not the only one, in creating a credible hospital simulation [Gunal and Pidd 2007]. By far the largest source of unscheduled patient admissions to many hospitals is the Emergency Department (ED), even when there is no precipitating event such as an epidemic or terrorist attack. Patients present with a variety of symptoms to an ED and are assigned a classification level (called an "acuity level") based on the severity of their symptoms. Even if two individual patients did arrive independently of one another (say, heart attacks, not traffic accidents), the overall demand process is observably nonstationary (Saturday nights are usually worse than Tuesday mornings) and state dependent (waiting room queues can be discouragingly large, even for critically ill patients).

Emergency departments in the United States and elsewhere are increasingly dysfunctional and risky to patient health and welfare [Johnson 2008; Eitel and Samuelson 2011]. There is a systemic operational issue that has put EDs in crisis. A major ED problem is patient "boarding," when patients who are ready for hospital admission are bedded in the hallways of the ED waiting for an empty bed in the appropriate ward. The movement from the ED to the hospital floors is, in the vernacular of industrial engineers, a "pull" system. This lack of buffering between the hospital floor and the ED makes unscheduled floor admissions state dependent. This problem has been addressed in desperate and creative ways. A particularly successful method is simply using a "push" approach for patient transfer, where patients are moved to the hallways of the hospital wards, not left in the ED hallway [Viccellio et al. 2009]. Unscheduled hospital admissions from the ED can be high dimensional, nonstationary, censored, and state dependent. Additionally, it can be hard to distinguish between nonstationarity and state dependency in the observed data.

2.1. Nonstationary Example

As an example, we consider the ED data collected as part of a research experiment in a large urban hospital. Patients arriving to the ED are evaluated by a triage nurse and are assigned a number to represent their acuity level. We consider three aggregated classification levels, with category 1 reserved for patients with the most severe symptoms. The likelihood of hospitalization of a patient in each category appears to be quite stable, but the populations in each vary in a complex manner.

One way to look at the data is by looking at the number of patients who are in the ED in each category at discrete time points. We define the patients in the system as those who have arrived to the ED and either are awaiting a bed or have been assigned a bed in the ED and are being treated before being discharged or sent to the hospital ward. Although we have a three-dimensional time series (one for each category), we show an example using two categories (1 and 2) to visually explain the method. Figure 1 shows the progression of the number of category 1 and 2 patients in the system throughout 1 day, from midnight to midnight, plotted against each other. Each arrow shows the transition over one half-hour increment of time. The arrows reveal the dependence by

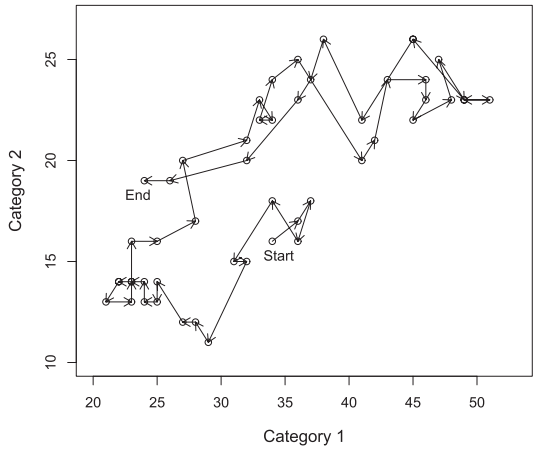

Fig. 1. The number of category 1 and 2 patients in the ED system, collected every half hour for 1 day. The arrows show the transitions over each increment of time.

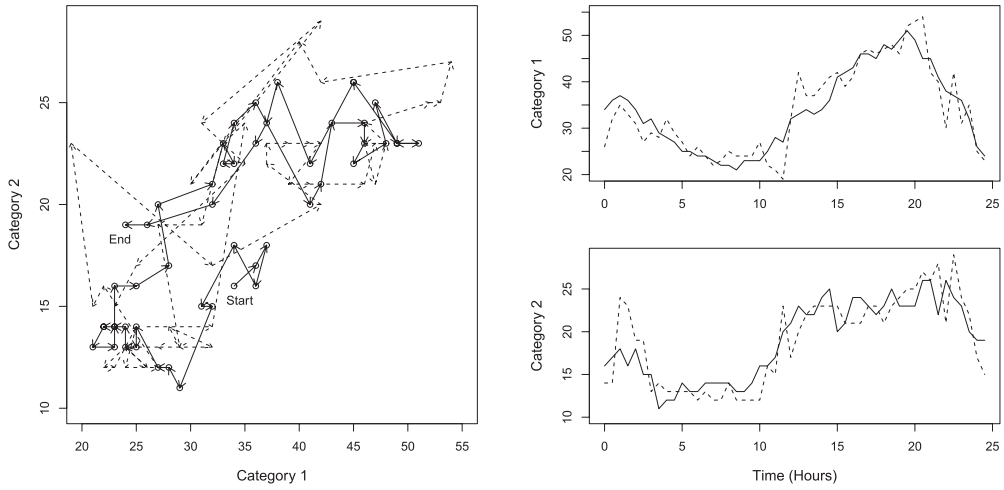

Fig. 2. Left plot: Data boid (solid line) with follower simulated boid (dashed line). Right plots: Original data (solid line) with simulated boid data (dashed line).

showing the state transitions over time. For an n-dimensional time series, this path would travel through n-dimensional space.

We next simulate a boid that follows the same general direction of the data boid in Figure 2. Of course, any number of such boids in a flock can be generated for multiple replications. Depending on how closely we want the simulated data to follow the real data, we can adjust this simulated boid path. The left plot of Figure 2 shows one such simulated boid, and the right plots show the simulated boid data along with the original time series. We round the values of the simulated boid to integers in order to provide meaningful values for an ED simulation. Qualitatively, we see that the simulated data captures the nonstationary properties of the original time series. Because the simulated boid follows the joint motion of both time series, dependence is captured as well.

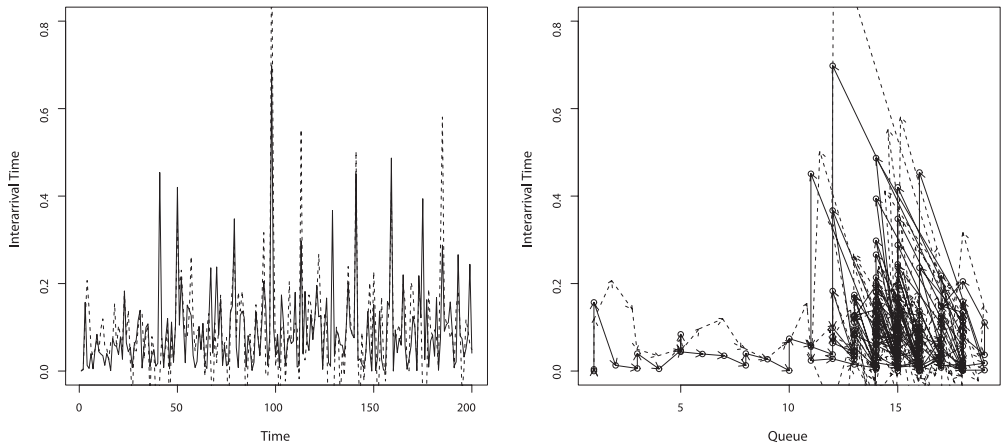

Fig. 3. Left: Interarrival times for a state-dependent arrival process with observed data (solid line) and simulated boid data (dashed line). Right: A plot of the simulated boid (dashed line) generated by following the data boid (solid line) that maps the interarrival time against the queue. Initialization bias and state dependency can be observed.

2.2. State-Dependent Example

Most service systems experience demands that behave in a nonstationary or state-dependent manner, and it might be difficult to distinguish these behaviors. For example, people might join long queues at a lower rate (state dependency) and/or long queues also occur during rush hour (nonstationarity). In either case, realistic modeling of the demand process can be difficult. In the simplest nonstationary case, modeling a nonhomogeneous Poisson process simply by estimating the arrival rate is difficult and certainly challenging to account for the uncertainty because of the fact that this estimated rate is itself a random process. For references on estimating parameters of nonhomogenous arrival processes, see Arkin and Leemis [2000], Gerhardt and Nelson [2009], and Kuhl [2011]. If the arrivals are state dependent, say, in reluctance to join long queues, then there may be censoring (latent demand is higher than observed demand). Regardless of the reasons, it is important to be able to model complex service demand processes. Our approach holds promise in addressing these issues because we follow the data without fitting a parametric model.

For illustration, consider a simple capacitated queue where the unobservable latent arrivals are according to a stationary Poisson process, but the number of actual arrivals who enter the system is inversely proportional to the queue length and the rate is unknown. A single path of the interarrival times (all you see in the real world) of this process looks like the left plot of Figure 3. This plot shows the real interarrival times along with those from a simulated boid. It is tempting to model this process as a cyclic nonstationary process, but the pseudo-cycles occur because of the underdamped feedback control loop due to the state dependency. The latent demand process actually had a constant rate. If this system were to be replicated with a different seed, the same pseudo-cycle would appear at the same frequency but perhaps with a phase shift. Phase shifts in apparent cyclic behavior can be introduced into the simulated boids by simply delaying their input to the model. Distinguishing actual cyclic nonstationarity from state dependency using only data in a single short real sample path may be impossible; however, one can simulate for sensitivity analysis using random phase shifts in a flock of simulated boids to see if this actually matters in the simulation performance.

The right plot of Figure 3 shows the interarrival time plotted against the queue size. Here we observe the initialization bias in the system, which starts empty and idle. We also see the state dependence, in that when the queue gets large, arrivals to the system are less frequent until the queue size drops. By using the interarrival times and the queue sizes as dimensions of a flight path, a flock of replications can be created without having to estimate the state dependency or nonstationarity. One issue with the simulated boid is that it produced interarrival times that are less than zero (we can take the positive parts of each dimension, which qualitatively works well). Our basic flocking algorithm follows the data without adding any constraints on the output space, but constraints can be added by incorporating them directly into a flocking algorithm implementation. Wall-following algorithms are used in robotics models to address boundary or region avoidance problems [Borenstein and Koren 1989].

3. IMPLEMENTATION

Previous work in Schruben and Singham [2010] and Singham et al. [2011] introduced flocking algorithms to replicate data. In this article, we develop the statistical properties of two general improved flocking algorithms. Different flocking algorithms may be appropriate depending on the data and the requirements of the modeler. The point here is to show that simple geometric models can be used to replicate data in a way that systematically produces output that exhibits qualitatively similar properties, like an observable trend or a level of variation in the data. As in Figure 1, we take each time series of the data as the coordinates of one dimension of a path in space. We simulate paths that follow the path formed by the alpha boid according to the two algorithms described next.

3.1. Algorithm 1

Suppose the real multivariate data (n time series) is indexed by time as $\mathbf{x}_1, \ldots, \mathbf{x}_t$, where each vector $\mathbf{x}_i$ has length n. We wish to simulate a multivariate time series $\mathbf{y}_1, \ldots, \mathbf{y}_t$, which follows the real data series $\mathbf{x}$ with affinity λ, which can take any real value. A value of $\lambda = 1$ means the simulated data follows the real data exactly, and $\lambda = 0$ means the simulated data is random and has no relation to $\mathbf{x}$ (except for initializing the algorithm). Negative values of λ mean the simulated data moves in the opposite direction of the real data and $\lambda > 1$ means the simulation data overshoots in the direction of the real data.

Assume that we have simulated a boid that follows the real data up to time $t-1$ as $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}$, and we wish to generate $\mathbf{y}_t$ from these previous values and $\mathbf{x}_t$. Because this algorithm is recursive, generated values of $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}$ will have taken into account $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$. The value of $\mathbf{y}_1$ can be initialized as $\mathcal{N}(\mathbf{x}_1, \sigma^2\mathbf{I}_n)$, where $\mathbf{I}_n$ is the identity matrix with dimension n. Define R_t as the distance between $\mathbf{y}_{t-1}$ and $\mathbf{x}_t$ in Euclidean space. Let $\boldsymbol{\theta}_t$ be the normalized direction vector $\mathbf{x}_t - \mathbf{y}_{t-1}$. This means that $\mathbf{y}_{t-1} + R_t\boldsymbol{\theta}_t = \mathbf{x}_t$. Next, let $\boldsymbol{\phi}_t$ be the direction vector formed by the origin and a point that is normally distributed in $\mathbf{R}_n$ with a mean vector of zeros and a covariance that is the identity matrix with dimension n. In other words, $\boldsymbol{\phi}_t$ is a random vector whose values are distributed as $\mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$.

The simulated flocked data is generated according to the following recursion:

$$\mathbf{y}_t = \mathbf{y}_{t-1} + R_t[\lambda\boldsymbol{\theta}_t + (1-\lambda)\boldsymbol{\phi}_t]. \tag{1}$$

We derive the conditional distribution of $\mathbf{y}_t$ given $\mathbf{y}_{t-1}$ as:

$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}\big(\mathbf{y}_{t-1} + R_t\lambda\boldsymbol{\theta}_t, R_t^2(1-\lambda)^2\mathbf{I}_n\big).$$

The unconditional distribution of $\mathbf{y}_t$ could be found by recursively integrating over possible values of $\mathbf{y}_{t-1}, \mathbf{y}_{t-2} \ldots$, which are normally distributed, so we leave it in

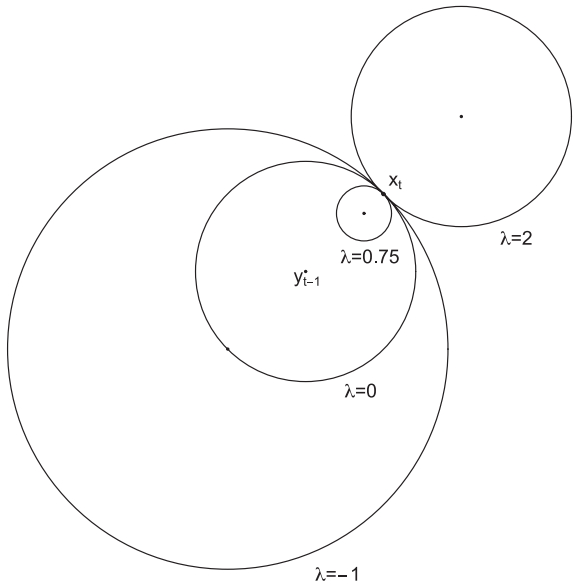

Fig. 4. Distribution of data generated for Algorithm 1 using $\sigma^2 = 0$, for different values of λ and $n = 2$. The simulated values of $\mathbf{y}_t|\mathbf{y}_{t-1}$ are normally distributed with means at the circle centers and standard deviations shown by the radius of each circle.

conditional form for simplicity. By conditioning on $\mathbf{y}_{t-1}$, the values of R_t and $\boldsymbol{\theta}_t$ become known for the purposes of calculating the distribution and $\boldsymbol{\phi}$ is the only random term. We show how this distribution can be simplified for key choices of λ. For $\lambda = 0$, we have $\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(\mathbf{y}_{t-1}, R_t^2\mathbf{I}_n)$. This means the real data does not determine the direction of the simulated data, but the next point is normally distributed around $\mathbf{y}_{t-1}$ with a standard deviation equal to the distance to the next point $\mathbf{x}_t$. For $\lambda = 1$, we have $\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(\mathbf{y}_{t-1} + R_t\boldsymbol{\theta}_t, \mathbf{0}_{nn})$, where $\mathbf{0}_{nn}$ is an $n \times n$ matrix of zeros, which is equivalent to $\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(\mathbf{x}_t, \mathbf{0}_{nn})$, so $\mathbf{y}_t = \mathbf{x}_t$ as in a trace-driven simulation. Figure 4 shows the distribution of $\mathbf{y}_t|\mathbf{y}_{t-1}$ for different values of λ using circles centered at the mean of the distribution and the standard deviation as the radius.

An additional noise term can be added so that even for $\lambda = 1$, there will be randomness in the flocked data. Suppose we let $\boldsymbol{\epsilon}_t$ be a random vector with distribution $\mathcal{N}(\mathbf{0}_n, \sigma^2\mathbf{I}_n)$. Consider the modified algorithm:

$$\mathbf{y}_t = \mathbf{y}_{t-1} + R_t[\lambda\boldsymbol{\theta}_t + (1-\lambda)\boldsymbol{\phi}_t] + \boldsymbol{\epsilon}_t. \tag{2}$$

The conditional distribution for $\mathbf{y}_t$ in (2) is:

$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}\big(\mathbf{y}_{t-1} + R_t\lambda\boldsymbol{\theta}_t, \big[R_t^2(1-\lambda)^2 + \sigma^2\big]\mathbf{I}_n\big).$$

In this version, there is an additive noise component that is independent and normally distributed along each dimension with mean $\mathbf{0}$ and variance $\sigma^2\mathbf{I}_n$. If $\lambda = 1$, the method would simply result in $\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(\mathbf{x}_t, \sigma^2\mathbf{I}_n)$, which may be desired if multiple perturbed data boids are to be generated for sensitivity analysis. For any λ, if $\sigma^2 > 0$, the circles in Figure 4 would be centered at the same location but would be larger to reflect the added variance.

3.2. Algorithm 2

We again generate the next data point $\mathbf{y}_t$ based on $\mathbf{y}_{t-1}$ and information from $\mathbf{x}$. Here, instead of a random direction component, we move away from $\mathbf{y}_{t-1}$ in the same direction

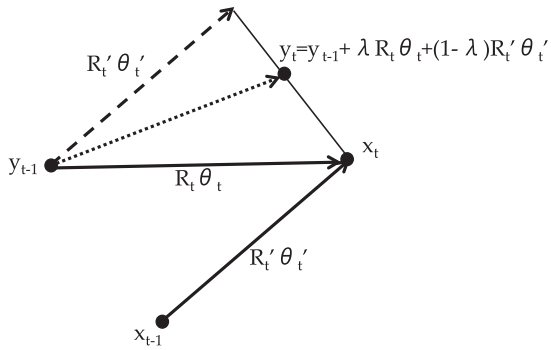

Fig. 5. Pictorial representation of Algorithm 2. The dashed line shows how we would move from $\boldsymbol{y}_{t-1}$ in the same way that $\boldsymbol{x}_{t-1}$ moved to $\boldsymbol{x}_t$. The dotted line shows the weighted average of the two directions considered.

that $\mathbf{x}_t$ moved away from $\mathbf{x}_{t-1}$. Figure 5 shows the two directions that determine the movement from $\mathbf{y}_{t-1}$ to $\mathbf{y}_t$. We have the direction $R_t\boldsymbol{\theta}_t$ as before, which is $\mathbf{x}_t - \mathbf{y}_{t-1}$, but now we also consider $R_t'\boldsymbol{\theta}_t' = \mathbf{x}_t - \mathbf{x}_{t-1}$.

Using λ as the weighting for the linear combination of the two direction vectors, construct the recursive algorithm:

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \lambda R_t\boldsymbol{\theta}_t + (1-\lambda)R_t'\boldsymbol{\theta}_t' + \boldsymbol{\epsilon}_t. \tag{3}$$

For this algorithm, the only random component is the $\boldsymbol{\epsilon}_t$ term, which must be included in order to generate replications. Incorporating the behavior of $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ helps the simulated boid maintain the directional momentum and trends of the data. Because there is no random direction component, there are fewer random jumps in the output time series. This may be beneficial if the user is looking for output time series with similar levels of variability as the data.

If we start the algorithm using $\mathbf{y}_1 \sim \mathcal{N}(\mathbf{x}_1, \sigma^2\mathbf{I}_n)$, then the conditional distribution of $\mathbf{y}_t$ using (3) is:

$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}\big(\mathbf{y}_{t-1} + \lambda R_t\boldsymbol{\theta}_t + (1-\lambda)R_t'\boldsymbol{\theta}_t', \sigma^2\mathbf{I}_n\big).$$

Let $\lambda = 1$, and the algorithm reduces to $\mathbf{y}_t \sim \mathcal{N}(\mathbf{x}_t, \sigma^2\mathbf{I}_n)$. As in Algorithm 1, if $\sigma^2 = 0$, then $\mathbf{y}_t = \mathbf{x}_t$. If $\lambda = 0$, then we have $\mathbf{y}_t|\mathbf{y}_{t-1} = \mathcal{N}(\mathbf{y}_{t-1} + R_t'\boldsymbol{\theta}_t', \sigma^2\mathbf{I}_n)$. By recursively breaking down $\mathbf{y}_{t-1}$ and using the fact that $\mathbf{x}_{t-1} + R_t'\theta_t' = \mathbf{x}_t$, we derive the distribution of $\mathbf{y}_t$ as $\mathcal{N}(\mathbf{x}_t, t\sigma^2\mathbf{I}_n)$. If there is no noise term (or if $\sigma^2 = 0$) and $0 < \lambda < 2$, then $\mathbf{y}_t$ converges to $\mathbf{x}_t$ as $t \to \infty$. The noise term is needed not just to provide replications, but to prevent this convergence.

3.3. Choosing Implementations

While we have suggested two specific algorithms that can be used to replicate multidimensional time series, much research remains to explore what methods are best applied to different types of data. We have proposed the parameters λ and σ to control the types of output that are produced, but there may be entirely different ways of approaching the problem. Variations on these algorithms can be made on an *ad hoc* basis in order to meet the needs of the user (e.g., if nonnegativity or integer values are required). It is not necessary to use normally distributed random errors, and the choice of distribution may depend on the specific dataset being analyzed. Shifts in the data values, or along the time axis, can be simple ways of testing model sensitivity and can be used in conjunction with the algorithms provided.

When choosing values of λ and σ, there are two possible approaches. One approach is to try multiple combinations of λ and σ and plot the output time series against the real data. Some parameter values may replicate the properties of the data well; others may provide very different properties. To test the model's sensitivity to input data, we suggest using replications that appear different from the data in order to see how the model performs under extreme or nonusual circumstances. A second approach is to use multiple data series to calibrate the choices of λ and σ and use these values to replicate the data. While it is certainly useful to be able to replicate the properties of the data, we emphasize the importance of sensitivity analysis using different parameter choices.

Schruben and Singham [2010] recommended rescaling the data by mapping it to the $[0, 1]^n$ hypercube. This rescaling is not necessary to implement the previous algorithms but is helpful when the relative scale of each time series component is different. The choices of λ and σ then affect each series equally, and the simulated flock of boids can be mapped back to their appropriate values using an inverse transformation.

4. EMERGENCY DEPARTMENT PATIENT ARRIVAL SIMULATION

Using the algorithms described in Section 3, we simulate boids that model the arrival of patients in the ED for each acuity level. The boid that models the state of the system (the number of patients present in the ED) could also be replicated to provide input to another hospital simulation or inform a staffing decision, but in this section we focus on patient arrivals to compare the results to simulation using a nonhomogenous Poisson process.

We take the alpha boid as the number of patients arriving each hour in each category over 1 week of time. Because the data in category 3 has a much smaller scale than the values for categories 1 and 2, we map the alpha boid to $[0, 1]^3$ before running the flocking algorithms by taking each time series and rescaling it linearly to values between [0, 1]. The simulated boid values are rounded to the nearest integer, and values less than zero are changed to zero to provide realistic values.

We run Algorithm 1 on the hospital data using different sets of parameters. One simulated boid is plotted in Figure 6 using $\lambda = 0.55$ and $\sigma = 0.01$, along with the original time series for the three categories. We see that the time series for the ED data are complex cyclical, with peaks occurring in the afternoon and early evening. The boid follows these cycles seen in the actual data, but with some variation. There are a few spikes in the simulated data that do not appear in the original data. These can be induced for sensitivity analysis to test the simulation model's behavior when there are high-impact rare events. In Singham et al. [2011], waypoints of hostile agents in a border-crossing scenario were varied in extreme ways using boids (sometimes the agents were sent in the opposite direction of the border). This revealed situations in which the simulation was not properly accounting for agent movement. Using "unrealistic" input was useful in detecting when the logic of the simulation code, based on implicit assumptions, might be broken.

Figure 7 shows the results for Algorithm 2 using parameters $\lambda = 0.08$ and $\sigma = 0.12$. The affinity parameter has a different interpretation here, and we find that using a small value of λ produces time series that might be appropriate for an ED simulation. Here, the boid produces similar types of peaks and valleys as the original data, appearing to maintain its directional momentum.

The methods presented can also be used in conjunction with parametric models. If the arrival of patients to a hospital is thought to be a nonhomogenous Poisson process, the rate function can be estimated (according to a method such as that in Kuhl and Wilson [2000]). The estimated rate function for each of the three time series can be

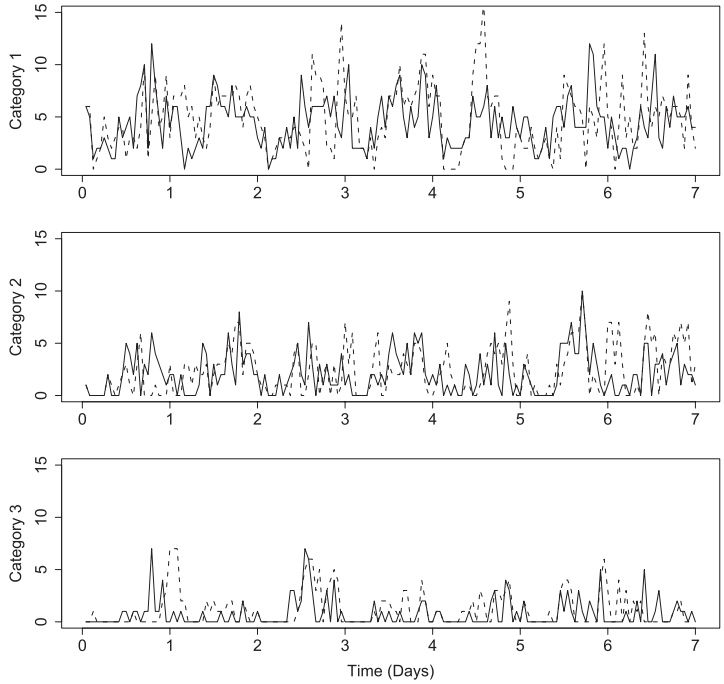

Fig. 6. Algorithm 1: Number of patients in the ED for 10 days of data for each category. Parameters used: $\lambda = 0.55, \sigma = 0.01$. Solid line is the real data; dashed line is simulated data.

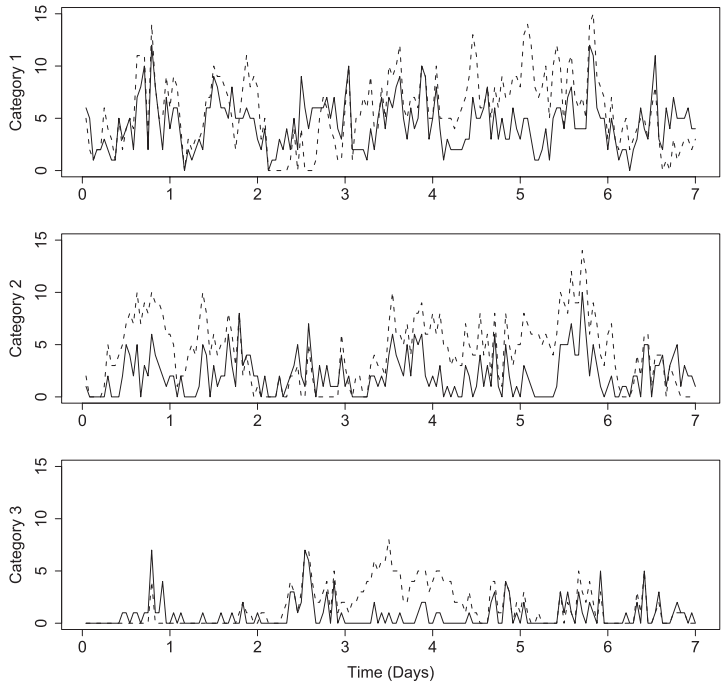

Fig. 7. Algorithm 2: Number of patients in the ED for 10 days of data for each category. Parameters used: $\lambda = 0.08, \sigma = 0.12$. Solid line is the real data; dashed line is simulated data.

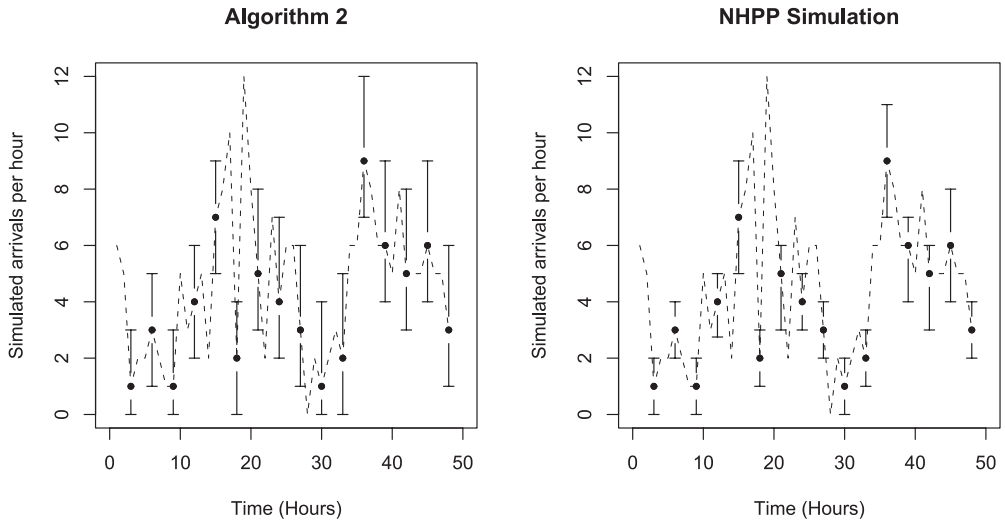

Fig. 8. Medians (dots) and interquartile ranges (whiskers) for simulated arrival data generated using Algorithm 2 and simulated nonhomogeneous Poisson processes. The dashed line shows the real number of arrivals.

mapped as a data boid, and corresponding simulated boids generated. The user could then check how sensitive the model is to variations in this rate function using flocking algorithms to generate the time series of rate functions having similar dependency structures. These simulated rate functions could be used to generate replications. If the user could make the assumption that all weeks were driven by the same process, the data could be divided into weeks, and an ensemble average taken to derive a trend. While this would result in an alpha boid smoothed from averaging over the variation from different weeks, the noise could be inserted back in using λ and σ.

Assessment of these algorithms is mainly qualitative at this point. In practice, Turing tests can be used to see whether the simulated boid data appears realistic to domain experts [Schruben 1980]. Visually observing the simulated time series allows one to see whether the algorithms are generating values that appear appropriate. The flocking methods presented here appear most appropriate in trying to replicate data with trends or cycles. The authors tested a previous version of the first algorithm in Schruben and Singham [2010] on vector autoregressive data. Although the algorithm could mimic the behavior of this data by simulating boids that had similar best-fit autoregressive parameters, it followed the specific peaks and valleys given by a particular realization of the process. Clearly, in this case one would prefer to fit the correct parametric model to the data and simply generate new vector autoregressive series as input to the simulation runs.

We compare Algorithm 2 to a simple heuristic for fitting and simulating a nonhomogenous Poisson process (NHPP). We estimate a sample rate function for the NHPP by taking the number of arrivals in each 1-hour interval over 2 days' worth of our data. We simulate 1,000 replications from this estimated rate function using a thinning process and then calculate the number of arrivals in each hour for each replication. Additionally, we simulate 1,000 replications of the same rate function using Algorithm 2 to generate the number of arrivals in each hour. Figure 8 shows the medians and interquartile ranges for the number of simulated arrivals during various hours. For clarity, we only show the results from every third hour. The dashed line shows the data that was used as the real rate function. The figure reveals that for our particular

choice of parameters ($\lambda = 0.08$ and $\sigma = 0.12$), the variability is greater than that of the NHPP simulation. While statistical properties of the NHPP simulation can be calculated analytically, we can control the desired level of variability in Algorithm 2 by changing the parameters used. The assumption of an NHPP is valid in this case, but the algorithm can deliver useful inputs to a simulation without making that assumption.

5. CONCLUSIONS

This article is intended to introduce the fundamental concepts of this new approach to simulation of multivariate time series and replicating complex trace data. This proposed method is different from current methods in many ways. The approach has many of the positive aspects of trace-driven simulations but eliminates two of its main shortcomings, allowing replication and sensitivity analysis. The two example algorithms presented simulate data directly from the real data, so simplifying statistical assumptions or resource-intensive data fitting algorithms are not required. The affinity parameter to model the sensitivity to the data is directly included in the model and the user can easily vary it. The method is simple and flexible enough that it can be applied to complex, high-dimensional data.

Like any new methodology, there are many open questions for future research. Among these are including both simulation input and output time series as coordinates of the flight paths with reward or penalty fields for performance measurement, and extending the affinity parameter to an affinity matrix to model more complex serial and cross-dependencies among the components of multivariate time series. Multivariate initialization truncation to warm up simulation runs is another promising area of research. In order to be effective for practitioners, methods should be able to adapt to changing conditions and generate input that can go directly into a simulation model for decision making. As automatic data collection techniques become more widespread, modeling techniques that can take data directly as input without making additional assumptions are uniquely flexible. Decision makers can easily adjust the affinity and noise parameters to see how their proposed system might react in different future scenarios.

Although these flocking algorithms are general and can be applied to any time series, they cannot guarantee that the simulated data will meet the needs of the modeler. The theory for this method has not yet been fully developed, but its anticipated flexibility in handling high-dimensional data suggests that it may be useful for modelers who do not have statistical requirements for their simulation input but simply want qualitatively similar time series with which to stress test their model. The methods introduced here provide an alternative way of thinking about the input modeling process and an easy way to generate simulation replications without fitting a model. These simulated boids can be generated in a systematic way and used for sensitivity analysis when only one set of trace data is available.

ACKNOWLEDGMENTS

The authors would like to acknowledge the associate editor and two anonymous referees for many helpful comments that improved the substance and presentation of this article.

REFERENCES

B. L. Arkin and L. M. Leemis. 2000. Nonparametric estimation of the cumulative intensity function for a nonhomogeneous Poisson process from overlapping realizations. *Management Science* (2000), 989–998.

B. Biller. 2009. Copula-based multivariate input models for stochastic simulation. *Operations Research* 57, 4 (2009), 878–892.

J. Borenstein and Y. Koren. 1989. Real-time obstacle avoidance for fact mobile robots. *IEEE Transactions on Systems, Man and Cybernetics* 19, 5 (1989), 1179–1187.

D. Eitel and D. A. Samuelson. 2011. OR in the ER. *ORMS Today* 38, 4 (2011).

I. Gerhardt and B. L. Nelson. 2009. Transforming renewal processes for simulation of nonstationary arrival processes. *INFORMS Journal on Computing* 21, 4 (2009), 630–640.

M. M. Gunal and M. Pidd. 2007. Interconnected DES models of emergency, outpatient, and inpatient departments of a hospital. In *Proceedings of the 39th Conference on Winter Simulation: 40 years! The Best Is Yet to Come*, S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton (Eds.). Institute of Electrical and Electronics Engineers, Inc., Piscataway, New Jersey, 1461–1466.

W. Härdle, J. Horowitz, and J. P. Kreiss. 2003. Bootstrap methods for time series. *International Statistical Review / Revue Internationale de Statistique* 71, 2 (2003), 435–459.

C. K. Johnson. 2008. Hospitals ease ER crowding with ward beds in halls. Online at USAToday.com. (October 2008). Retrieved from http://www.usatoday.com/news/health/2008-10-27-188502636_x.htm.

M. E. Kuhl. 2011. Nonstationary Input Processes. *Wiley Encyclopedia of Operations Research and Management Science* (2011).

M. E. Kuhl and J. R. Wilson. 2000. Least squares estimation of nonhomogeneous Poisson processes. *Journal of Statistical Computation and Simulation* 67, 1 (2000), 75.

C. W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 25–34.

L. W. Schruben. 1980. Establishing the credibility of simulations. *Simulation* 34, 3 (1980), 101.

L. W. Schruben and D. I. Singham. 2010. Simulating multivariate time series using flocking. In *Proceedings of the 2010 Winter Simulation Conference*, B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan (Eds.). Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, 1048–1054.

L. W. Schruben and D. I. Singham. 2011. Agent Based Output Analysis. In *Proceedings of the 2011 Winter Simulation Conference*, S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu (Eds.). Institute of Electrical and Electronics Engineers, Inc., Piscataway, New Jersey.

D. I. Singham, M. A. Therkildsen, and L. W. Schruben. 2011. Applications of flocking algorithms to input modeling for agent movement. In *Proceedings of the 2011 Winter Simulation Conference*, S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu (Eds.). Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ.

A. Viccellio, C. Santora, A. J. Singer, H. C. Thode Jr., and M. C. Henry. 2009. The association between transfer of emergency department boarders to inpatient hallways and mortality: a 4-year experience. *Annals of Emergency Medicine* 54, 4 (2009), 487–491.

Received September 2011; revised April 2012; accepted June 2012