Journal Title
XX(X):1–10

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Towards Reproducible Disease Models using the Systems Biology Markup Language

Leandro Watanabe[1], Jacob Barhak[2], and Chris Myers[1]

Abstract

Disease modelers have been modeling progression of diseases for several decades using tools such as Markov Models or microsimulation. However, they need to address a serious challenge; many models they create are not reproducible. Moreover, there is no proper practice that ensures reproducible models, since modelers rely on loose guidelines that change periodically, rather than well-defined machine-readable standards. The *Systems Biology Markup Language* (SBML) is one such standard that allows exchange of models amongst different software tools. Recently, the SBML Arrays package has been developed, which extends the standard to allow handling simulation of populations. This paper demonstrates through several abstract examples how microsimulation disease models can be encoded using the SBML Arrays package enabling reproducible disease modeling.
Received: September 25, 2017, **Revised:** March 25, 2018 and May 31,2018, **Accepted:** July 9, 2018

Keywords

Disease Modeling, Microsimulation, Reproducibility, SBML, Arrays Package

Introduction

Disease models attempt to explain phenomena observed by clinical trials and follow-up of patient populations through time. Such phenomena include complications of chronic diseases such as diabetes (1) and cardiovascular diseases (2), infectious diseases such as Ebola (3) and HIV (4), or even mental health conditions (5). Beyond complications, models can also include economic aspects, such as costs or quality of life related health utility scores. Models describe those phenomena using mathematical and statistical equations or other programmatic constructs.

In the past, differential equations have been used, which are still very dominant in the infectious disease domain (3). However, other disease models have used state transitions mechanisms. Markov cohort models have been prevalent in the past (5), but modern disease models tend to use *microsimulation* (1), where simulation considers each individual in the population separately. Some infectious disease models are also moving in the direction of individual-based simulation (6).

Individualization of computation makes models more flexible, but also more complex to understand. Therefore, clarity in model publication and transparency are essential. However, modeling practices in the field lack support for reproducibility. Publication of models' source code is rare (7; 8; 9; 10). The norm is still publication of descriptive-only models in papers in which they appear, and only rarely do authors attempt to publish in a way that their work can be reproduced. However, publishing models within a paper does not allow full reproducibility as numeric examples provided in papers have insufficient precision and are prone to misinterpretation (see, for example (1)).

The Mount Hood Diabetes Challenge highlighted this reproducibility problem (see `https://www.mthooddiabeteschallenge.com/`). The challenge revolved around reproducing models from two published papers. Multiple modeling teams around the world attempted to reproduce these published models, and they were unsuccessful. This is conclusive proof that a new method for model reproducibility is needed since models that cannot be reproduced are perceived to be non-credible.

To date, disease modelers have been trying to improve their model publication methods by publishing guidelines. Yet a better solution is to provide modeling tools that are reproducible to allow model exchange. This is exactly what the *Systems Biology Markup Language* (SBML) (11) and associated languages such as PharmML (12) and its counterpart Model Description Language (MDL) (13) are designed for. This paper continues the first attempt to reproduce disease models in such modeling languages that started with (14), which demonstrated how a disease model can be reproduced in three languages: SBML, PharmML, and MIcro Simulation Tool (MIST) (15). When the first paper was written, SBML capabilities lacked a definition of more complex models using microsimulation. SBML, as a community-driven standard, have biannual meetings about how SBML can be improved (16). Recently, the community talked about how to represent agent-based models in SBML at the COMBINE 2017 meeting, but such representation has not reached consensus. Nonetheless, SBML has evolved with

[1]Department of Electrical and Computer Engineering, University of Utah
[2]Jacob Barhak, Austin, TX

Corresponding author:
Leandro Watanabe, Dept. of Electrical and Computer Engineering, University of Utah, 50 S. Central Campus Drive, MEB Rm 2110, Salt Lake City, UT, 84112, Phone: (801) 581-6941
Email: l.watanabe@utah.edu

the recent introduction of the SBML Arrays draft package specification that can handle more complex models using microsimulation. This paper demonstrates this through a few abstract disease modeling examples that can now be implemented using SBML Arrays.

SBML Arrays

SBML is a standard representation that primarily targets chemical reaction networks. However, SBML has strong discrete event support in its core constructs, which allows the representation of a wide range of models other than chemical reaction networks in the form of Boolean networks (17), Petri nets (18), and Markov chains (19), among others. In addition, SBML has support for package extensions that enhances the standard even further with new constructs beyond the core constructs (20; 21; 22). In particular, the SBML Arrays package enables the instantiation of a specified number of identical model objects facilitating the representation of populations (23). SBML Arrays has been implemented within the C/C++ library of SBML called libSBML (24) and the Java library of SBML called JSBML (25). The JSBML library also supports validation and flattening of arrays.

Using SBML's discrete event support coupled with arrays, SBML can be used to represent microsimulation disease models. Disease models in SBML are probabilistic models that use arrays of parameters to encode each individual, where each index in the array represents a single person. Each possible state for an individual (e.g. Healthy, Sick, Dead, etc.) is created as an array of parameters, where each parameter is treated as a Boolean variable. SBML Events are used to transition states for each individual. Those events are not part of the model to be transported, they are used as an implementation mechanism to describe the desired model in SBML.

Disease Modeling Examples

To illustrate the requirements for disease models, the following sections presents several abstract examples that are successfully implemented using SBML coupled with the SBML Arrays package. We start with the same examples given in (14) and add microsimulation components to those that were not originally modeled by those discrete time Markov models. We then add two more examples that are impossible to model without SBML Arrays. Important nuances are discussed for each example.

Example 1: Simple Example

The first simple example (depicted in Figure 1) can be modeled as a cohort model as demonstrated before in (14), where the number of individuals in each state is counted for each time step. However, this paper implements it using microsimulation where each individual is processed through the model using Monte Carlo simulation with the probability defined. SBML Arrays defines an array of individuals, where each can be either *Alive* or *Dead*. Unlike cohort models where simulation continues for each time step until the end, microsimulation models can stop for individuals who reach a terminal state. In all simulations in this paper, this would be represented by the *Dead* state. This mechanism is used in disease models to shorten simulation time and to indicate non-existence of a record for a human in years after death, effectively diminishing cohort size.

Figure 1. State transition diagram of a simple Markov model. The model uses two disease states: Alive and Dead, where the Dead state terminates simulation. The yearly probability of transition between states Alive and Dead is: 0.05
Initial conditions: 100 people start in Alive and none in Dead.
Output: Number of people in each state for years 1-10.

This entire simulation is implemented as SBML events as can be seen in Table 1. SBML events are triggered if their trigger condition is previously false and then evaluated to true during simulation. This SBML mechanism is used to create a sequence of time steps that guides simulation. The *InstructionNumber* parameter helps SBML to control the firing sequence of events and specific events competing in time. This competition of events is an important SBML element and is not related to the model being implemented. Therefore the addition of the *InstructionNumber* parameter that forces discrete times for the sequence of occurrences. Also note that model time and SBML implementation time are different. In this example, there is a header of events enumerated as #0 ,#1,#2 that start each time step in the simulation. Event #0 advances the *Time* parameter. Event #1 provides a point where the user can take a snapshot of the data to represent the state of the system in the time step. Event #2 is used for termination. The last three events represent transitions. Namely, Event #3 generates a random number and stores it in the *Random* variable. Event #4 tests if the drawn random number matches the transition criteria and, if so, updates the states and increases the instruction count to progress the simulation. Event #5 is a counter event for event #4 that is triggered if event #4 does not happen. It is essential to advance the simulation by setting the *InstructionNumber*. Unless set, the simulation would not continue, since there would not be another event for the individual, which is how event #3 terminates simulation. Alternatively, termination can happen if the simulation time limit is reached. Future examples follow this structure. The models shown here are generated using the SBML Arrays package that automatically generates any arbitrary number of individual copies, which is 100 for the examples shown.

Example 2: Three State Markov Model

The next example (depicted in Figure 2) is a simple extension of the first one. This example demonstrates how new transitions are added by introducing more events. Table 2 represents the events implemented to run this simulation. Events #0-#2 form a simulation header. Events #3-#6 represent transitions originating from the *Healthy* state while #7-#10 represent transitions originating from the *Sick* state. Each state has three events since there are two transitions emanating from each event and therefore three competing options have to be checked: taking the first transition, taking

Table 1. SBML events for Example 1.

	Trigger	Assignments
0	InstructionNumber[d0] $= 0$	Time[d0] $=$ Time[d0] $+ 1$ InstructionNumber[d0] $= 0.1$
1	InstructionNumber[d0] $= 0.1$	InstructionNumber[d0] $= 0.2$
2	InstructionNumber[d0] $= 0.2 \wedge$ Dead[d0] $= 0 \wedge$ Time[d0] < 10	InstructionNumber[d0] $= 1$
3	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
4	InstructionNumber[d0] = 1.5 $\wedge$ Alive[d0] $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $<$ $(0 + 0.05)$	Alive[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 0$
5	InstructionNumber[d0] $= 1.5 \wedge$ Alive[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.05) \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$

Figure 2. State transition diagram of a three state Markov model. There are 3 disease states: Healthy, Sick, and Dead, where the Dead state is terminal The yearly transition probabilities are: Healthy to Dead: 0.01; Healthy to Sick: 0.2; Sick to Healthy: 0.1;Sick to Dead: 0.3.
Initial conditions: Healthy = 100, Sick = 0 and Dead = 0.
Output: Number of people in each state for years 1-10.

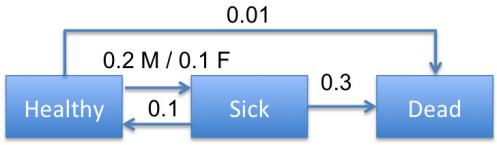


Figure 3. State transition diagram of a simple Markov model. There are 3 disease states: Healthy, Sick, and Dead, where the Dead state is terminal. The yearly transition probabilities are: Healthy to Dead: 0.01; Healthy to Sick: 0.2 for Male and 0.1 for Female; Sick to Healthy: 0.1; Sick to Dead: 0.3. The transition probability now depends on the cohort (Male or Female) and can be expressed as a function of a Boolean covariate Male.
Initial conditions: Healthy = (50 Male, 50 Female), Sick = (0,0) and Dead = (0,0).
Output: Number of men and women in each disease state for years 1-10.

the second transition, or not taking any transition. Our simulation generates a random variable and stores it in the *Random* parameter. The three events afterward check the three different possible options using the *Random* parameter and transition thresholds. This structure is used in all the remaining examples.

Example 3: Stratified Markov Model

This example (depicted in Figure 3) starts introducing microsimulation concepts since a parameter governs the transition probability. In this example, males become sick with higher probability than females and therefore simulation should show a higher sickness and death rate amongst males. This example is still simple enough to implement as two separate cohort models as can be seen in Table 3. The transition probabilities are controlled by the *Male* parameter, which is used in event #5 and in the counter event #6. Other than that, this example is similar to the previous example. Yet microsimulation becomes more significant and challenging when individuals have more characteristics. This is explored further in the next example.

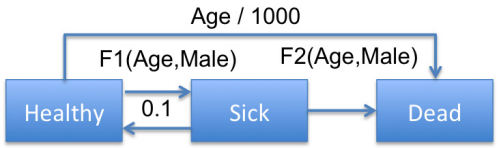
Example 4: State Transition Model Dependent on Changing Parameters

This example (depicted in Figure 4) can no longer be implemented using Markov cohort models due to the yearly change in *Age* and the stratification by *Male* and *Age* of the transition probabilities. This example captures the heterogeneity of the population by describing each individual's behavior. SBML arrays allows for the definition of distinct individuals. Table 4 presents the event sets for this example. SBML events plays a crucial role by increasing the *Age* every year before transition probabilities are calculated. The new element in this model is the change of *Age* before determining transitions in each simulation timestep. This can be seen in Instructions #3 to #5 that behave similar to transitions - note that some of the code is redundant and can be replaced by one event since event #5 never fires. However, this example maintains this code structure for compatibility and future extendibility. Once again, our method uses *InstructionNumber* to guide the model during simulation such that state transitions are considered at each simulation time step only after *InstructionNumber* reaches the value of 2.

Despite the complexity of this example, it is not yet representative of the full range of phenomena we wish to model that include treatment and cost. The next example shows how this is accomplished.

Table 2. SBML events for Example 2.

	Trigger	Assignments
0	InstructionNumber[d0] $= 0$	Time[d0] $=$ Time[d0] $+ 1$ InstructionNumber[d0] $= 0.1$
1	InstructionNumber[d0] $= 0.1$	InstructionNumber[d0] $= 0.2$
2	InstructionNumber[d0] $= 0.2 \wedge$ Dead[d0] $= 0 \wedge$ Time[d0] < 10	InstructionNumber[d0] $= 1$
3	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
4	InstructionNumber[d0] = 1.5 $\wedge$ Healthy[d0] = 1 $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + 0.01)$	Healthy[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 0$
5	InstructionNumber[d0] = 1.5 $\wedge$ Healthy[d0] = 1 $= 1 \wedge$ Random[d0] $\geq (0 + 0.01) \wedge$ Random[d0] $< (0 + 0.01) + 0.2$	Healthy[d0] $= 0$ Sick[d0] $= 1$ InstructionNumber[d0] $= 0$
6	InstructionNumber[d0] $= 1.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.01) + 0.2 \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$
7	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
8	InstructionNumber[d0] = 1.5 $\wedge$ Sick[d0] = 1 $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + 0.1)$	Sick[d0] $= 0$ Healthy[d0] $= 1$ InstructionNumber[d0] $= 0$
9	InstructionNumber[d0] = 1.5 $\wedge$ Sick[d0] = 1 $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) \wedge$ Random[d0] $< (0 + 0.1) + 0.3$	Sick[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 0$
10	InstructionNumber[d0] $= 1.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) + 0.3 \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$



$$\mathrm{F1}(\mathrm{Age}, \mathrm{Male}) = \mathrm{Min}(0.8, 0.1 \cdot (1 + \mathrm{Male}) + 0.01 \cdot \mathrm{Age})$$
$$\mathrm{F2}(\mathrm{Age}, \mathrm{Male}) = \mathrm{Min}(0.9, 0.01 \cdot \mathrm{Age} + 0.2 \cdot \mathrm{Male})$$

Figure 4. State transition model dependent on changing parameters. There are 3 disease states: Healthy, Sick, and Dead, where the Dead state is terminal. The yearly transition probabilities are: Healthy to Dead: Age/1000; Healthy to Sick: according to function F1 depending on Age and Male parameters; Sick to Healthy: 0.1; Sick to Dead: according to function F2 depending on Age and Male parameters. Pre-Transition Rules: Age increased by 1 each cycle. **Initial conditions:** Healthy = (50 Male, 50 Female with Age =1,2,,50 for each individual), Sick = (0,0) and Dead = (0,0). **Output:** Number of men and women in each disease state for years 1-10 and their ages in each state.

Example 5: State Transition Model with Treatment and Costs

This example (depicted in Figure 5 adds *Blood Pressure* (*BP*) as another parameter that increases yearly at different rates. Once *BP* is above a threshold, treatment is administered that drops it back closer to previous values. Moreover, costs include elements of *Age* and *Treatment*. Even this relatively simple example is complex enough to show why individual modeling is needed, and hence making SBML Arrays essential. Table 5 shows the event scheme implementation in SBML. Notice that in this example there are multiple post transition rules implemented as event triplets: #3-#5 handle *Age* increment in pre-transition, #6-#8 handle *BP* pre-transition update, #17-#19 determine whether treatment is administered post-transition, #20-#22 adjust *BP* according to treatment for next timestep post-treatment calculation, #23-#25 calculate yearly cost that includes treatment cost, and finally #26-#28 accumulate total cost. The important elements of this simulation are the pre-transition rules and post-transition rules. Each of those rule sets needs to be executed in sequential order during simulation. SBML events allow for timing these using the *InstructionCounter*.

Results

The examples described above are implemented as a combination of the Python programming language and SBML files to define the models and then simulated using iBioSim, which supports SBML Arrays. Since these examples are not intuitive, a reference is needed to provide some comparison of results. The MIcro Simulation Tool (MIST) is used to implement the same examples.

Table 3. SBML events for Example 3.

	Trigger	Assignments
0	InstructionNumber[d0] $= 0$	Time[d0] $=$ Time[d0] $+ 1$ InstructionNumber[d0] $= 0.1$
1	InstructionNumber[d0] $= 0.1$	InstructionNumber[d0] $= 0.2$
2	InstructionNumber[d0] $= 0.2 \wedge$ Dead[d0] $= 0 \wedge$ Time[d0] < 10	InstructionNumber[d0] $= 1$
3	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
4	InstructionNumber[d0] = 1.5 $\wedge$ Healthy[d0] = 1 $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + 0.01)$	Healthy[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 0$
5	InstructionNumber[d0] = 1.5 $\wedge$ Healthy[d0] = 1 $= 1 \wedge$ Random[d0] $\geq (0 + 0.01) \wedge$ Random[d0] $< (0 + 0.01) + 0.1 * (1 +$ Male[d0]$)$	Healthy[d0] $= 0$ Sick[d0] $= 1$ InstructionNumber[d0] $= 0$
6	InstructionNumber[d0] $= 1.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.01) + 0.1 * (1 +$ Male[d0]$) \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$
7	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
8	InstructionNumber[d0] = 1.5 $\wedge$ Sick[d0] = 1 $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + 0.1)$	Sick[d0] $= 0$ Healthy[d0] $= 1$ InstructionNumber[d0] $= 0$
9	InstructionNumber[d0] = 1.5 $\wedge$ Sick[d0] = 1 $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) \wedge$ Random[d0] $< (0 + 0.1) + 0.3$	Sick[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 0$
10	InstructionNumber[d0] $= 1.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) + 0.3 \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$

Since MIST is particularly designed for disease modeling, comparable results provide sufficient support to the claim that SBML Arrays is suitable to create reproducible disease models. Figures 6, 7, 8 presents the result of simulation using MIST compared to SBML Arrays implemented in iBioSim. Since this is random simulation, results should not match exactly using a single run of the simulation, Figure 6 shows this case, yet they are comparable enough to indicate a similar simulation. To verify that the models indeed are identical, the models are executed 10 times and results are averaged as shown in Figure 7. The average results of 100 repetitions are shown in Figure 8. The plots show clear convergence as more repetitions are added. To provide additional support, we conducted statistical analysis of results for each example in a similar way. The models were executed with MIST, which includes utilities to run simulations multiple times and extract statistics from multiple simulations. These statistics include mean, standard deviation, minimum, and maximum of results reported by different repetitions. The same models were executed on iBioSim and statistics were extracted using a dedicated script that was written for this paper. Finally a python script collected the results and generated the graphics presented in the figures in this paper. All scripts are provided in the Github Repository. Statistical analysis results are provided for each example: Example 1 (Figure 9), Example 2 (Figure 10), Example 3 (Figures 11,12), Example 4 (Figures 13,14),and Example 5 (Figures 15,16 17,18). In these analyses, the vertical axis represents the absolute difference between MIST and iBioSim results. The left column shows the mean as circles and standard deviation as squares for each year in simulation. The right column shows the average difference of all years and the horizontal axis represents repetitions. The convergence of the model results is clearly seen from those statistics for most plots where both the mean and standard deviation difference is reduced by adding iterations. There are several outliers where mean does not follow this trend, such as in example 3 healthy male 10 repetitions mean, yet even in those cases standard deviation statistic improves or stays similar implying convergence. In Example 5 there are three cases where standard deviation does not improve for 100 repetitions: dead young male and age old female, and cost old female. However, in those examples the mean statistic improves and considering that example 5 is highly stratified, has some relatively rare events and has somewhat volatile changes in age, so it is quite reasonable and expected. Therefore, we conclude that the examples are reproduced properly between tools as clearly seen in 8.

To support this reproducibility, the models, example code, and results for both implementations are available at: `https://github.com/Jacob-Barhak/DiseaseModelsSBML`. This repository includes detailed instructions to replicate the results in this paper in both MIST and iBioSim as well as python code to assist SBML creation

Table 4. SBML events for Example 4.

	Trigger	Assignments
0	InstructionNumber[d0] $= 0$	Time[d0] $=$ Time[d0] $+ 1$ InstructionNumber[d0] $= 0.1$
1	InstructionNumber[d0] $= 0.1$	InstructionNumber[d0] $= 0.2$
2	InstructionNumber[d0] $= 0.2 \wedge$ Dead[d0] $= 0 \wedge$ Time[d0] < 10	InstructionNumber[d0] $= 1$
3	InstructionNumber[d0] $= 1$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
4	InstructionNumber[d0] = 1.5 $\wedge$ 1	Age[d0] $=$ Age[d0] $+ 1$ InstructionNumber[d0] $= 2$
5	InstructionNumber[d0] = 1.5 $\wedge$ 0	InstructionNumber[d0] $= 2$
6	InstructionNumber[d0] $= 2$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 2.5$
7	InstructionNumber[d0] $= 2.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $<$ $\left(0 + \frac{\text{Age[d0]}}{1000}\right)$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
8	InstructionNumber[d0] = 2.5 $\wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq \left(0 + \frac{\text{Age[d0]}}{1000}\right) \wedge$ Random[d0] $< \left(0 + \frac{\text{Age[d0]}}{1000}\right) +$ min$(0.8, 0.1 * (1 +$ Male[d0]$) + 0.01 *$ Age[d0]$)$	Sick[d0] $= 1$ Healthy[d0] $= 0$ InstructionNumber[d0] $= 0$
9	InstructionNumber[d0] = 2.5 $\wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq \left(0 + \frac{\text{Age[d0]}}{1000}\right) +$ min$(0.8, 0.1 * (1 +$ Male[d0]$) + 0.01 *$ Age[d0]$) \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$
10	InstructionNumber[d0] $= 2$	Random[d0] $=$ uniform$(0, 1)$ InstructionNumber[d0] $= 2.5$
11	InstructionNumber[d0] $= 2.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $<$ $(0 + 0.1)$	Sick[d0] $= 0$ Healthy[d0] $= 1$ InstructionNumber[d0] $= 0$
12	InstructionNumber[d0] = 2.5 $\wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) \wedge$ Random[d0] $<$ $(0 + 0.1) +$ min$(0.9, 0.2 *$ Male[d0]$ + 0.01 *$ Age[d0]$)$	Sick[d0] $= 1$ Healthy[d0] $= 0$ InstructionNumber[d0] $= 0$
13	InstructionNumber[d0] = 2.5 $\wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) +$ min$(0.9, 0.2 *$ Male[d0]$ + 0.01 *$ Age[d0]$) \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 0$

and additional statistical analysis. Although the results were generated with MIST and iBioSim. It is important to remember that this paper promotes SBML with arrays as a transfer mechanism between systems rather than focusing on a specific system. In addition, all of the examples have been uploaded to the BioModels database (26). The models used in this paper have been assigned the following identifiers: MODEL1803120002, MODEL1803120003, MODEL1803120004, MODEL1803120005, and MODEL1803120006 for Example 1, Example 2, Example 3, Example 4, and Example 5 respectively.

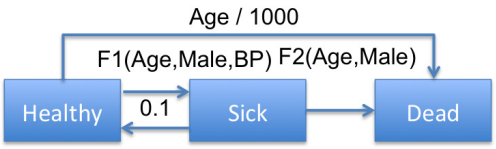
Discussion

The long term goal of this effort of implementing disease modeling examples in SBML is to eventually allow converting MIST examples to SBML using the SBML Arrays package. The provided examples pave the way in this direction.

Those examples do not cover all possible modeling elements used in epidemiological modeling, such as infectious disease modeling, discrete event simulation, or population generation. Only the very basic essential building block elements, that are regularly used to model chronic disease progression at the individual level, are presented here. Those examples are sufficient to support tasks such as life expectancy estimation and cost effectiveness analysis, which are core uses of disease models. Future work will include adding more elements such as handling event states, splitting and joining disease processes and other elements supported by MIST with the intention to promote SBML Arrays to be part of the SBML standard. In this work, the model transport is partially manual since MIST did not write the SBML code that was transported to iBioSim. Future work will address automation of this mechanism. This work is intended to establish feasibility of SBML Arrays as a model transport mechanism. It is only a step towards adoption by SBML editors into the SBML standard, which already provides SBML Arrays specification and tools towards such a goal.

Table 5. SBML events for Example 5.

	Trigger	Assignments
0	InstructionNumber[d0] $= 0$	Time[d0] = Time[d0] $+ 1$ InstructionNumber[d0] $= 0.1$
1	InstructionNumber[d0] $= 0.1$	InstructionNumber[d0] $= 0.2$
2	InstructionNumber[d0] $= 0.2 \wedge$ Dead[d0] $= 0 \wedge$ Time[d0] < 10	InstructionNumber[d0] $= 1$
3	InstructionNumber[d0] $= 1$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 1.5$
4	InstructionNumber[d0] = 1.5 $\wedge$ 1	Age[d0] = Age[d0] $+ 1$ InstructionNumber[d0] $= 2$
5	InstructionNumber[d0] = 1.5 $\wedge$ 0	InstructionNumber[d0] $= 2$
6	InstructionNumber[d0] $= 2$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 2.5$
7	InstructionNumber[d0] $= 2.5 \wedge 1$	InstructionNumber[d0] $= 3$ BP[d0] = BP[d0] $+ \frac{\text{Age[d0]}}{10}$
8	InstructionNumber[d0] = 2.5 $\wedge$ 0	InstructionNumber[d0] $= 3$
9	InstructionNumber[d0] $= 3$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 3.5$
10	InstructionNumber[d0] $= 3.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + \frac{\text{Age[d0]}}{1000})$	Healthy[d0] $= 0$ Dead[d0] $= 1$ InstructionNumber[d0] $= 4$
11	InstructionNumber[d0] $= 3.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq (0 + \frac{\text{Age[d0]}}{1000}) \wedge$ Random[d0] $< (0 + \frac{\text{Age[d0]}}{1000}) + \min(0.8, 0.1 * (1 + \text{Male[d0]}) + 0.01 * \text{Age[d0]} + (\frac{\text{BP[d0]}-120}{100})^2)$	Sick[d0] $= 1$ Healthy[d0]=0 InstructionNumber[d0] $= 4$
12	InstructionNumber[d0] $= 3.5 \wedge$ Healthy[d0] $= 1 \wedge$ Random[d0] $\geq (0 + \frac{\text{Age[d0]}}{1000}) + \min(0.8, 0.1 * (1 + \text{Male[d0]}) + 0.01 * \text{Age[d0]} + (\frac{\text{BP[d0]}-120}{100})^2) \wedge$ Random[d0] < 1	InstructionNumber[d0] $= 4$
13	InstructionNumber[d0] $= 3$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 3.5$
14	InstructionNumber[d0] $= 3.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq 0 \wedge$ Random[d0] $< (0 + 0.1)$	Sick[d0] $= 0$ Healthy[d0] $= 1$ InstructionNumber[d0] $= 4$
15	InstructionNumber[d0] $= 3.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $\geq (0 + 0.1) \wedge$ Random[d0] $< (0 + 0.1) + \min(0.9, 0.2 * \text{Male[d0]} + 0.01 * \text{Age[d0]})$	Sick[d0] $= 0$ Dead[d0]=1 InstructionNumber[d0] $= 4$
16	InstructionNumber[d0] $= 3.5 \wedge$ Sick[d0] $= 1 \wedge$ Random[d0] $< (0 + 0.1) + \min(0.9, 0.2 *$ Male[d0] $+ 0.01 *$ Age[d0] $\wedge$ Random[d0] < 1	InstructionNumber[d0] $= 4$
17	InstructionNumber[d0] $= 4$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 4.5$
18	InstructionNumber[d0] = 4.5 $\wedge$ 1	Treatment[d0] = Age[d0] $+ 1$ InstructionNumber[d0] $= 5$
19	InstructionNumber[d0] $= 4.5 \wedge 0$	InstructionNumber[d0] $= 5$
20	InstructionNumber[d0] $= 5$	Random[d0] = uniform$(0, 1)$ InstructionNumber[d0] $= 5.5$
21	InstructionNumber[d0] $= 5.5 \wedge 1$	BP[d0] = BP[d0] $-$ Treatment[d0]*10 InstructionNumber[d0] $= 6$
22	InstructionNumber[d0] $= 5.5 \wedge 0$	InstructionNumber[d0] $= 6$
23	InstructionNumber[d0] $= 6$	InstructionNumber[d0] $= 6$
24	InstructionNumber[d0] $= 6.5 \wedge 1$	CostThisYear[d0] = Age[d0] $+$ $Treatment[d0] * 10$ InstructionNumber[d0] $= 7$
25	InstructionNumber[d0] $= 6.5 \wedge 0$	InstructionNumber[d0] $= 7$
26	InstructionNumber[d0] $= 7$	InstructionNumber[d0] $= 7.5$
27	InstructionNumber[d0] $= 7.5 \wedge 1$	Cost[d0] = Cost[d0] $+$ CostThisYear[d0] InstructionNumber[d0] $= 0$
28	InstructionNumber[d0] $= 7.5 \wedge 0$	InstructionNumber[d0] $= 0$



$$\mathrm{F1}(\mathrm{Age},\mathrm{Male},\mathrm{BP}) = \mathrm{Min}(0.8, 0.1 \cdot (1+\mathrm{Male}) + 0.01 \cdot \mathrm{Age} + (\tfrac{BP-120}{100})^2)$$
$$\mathrm{F2}(\mathrm{Age},\mathrm{Male}) = \mathrm{Min}(0.9, 0.01 \cdot \mathrm{Age} + 0.2 \cdot \mathrm{Male})$$

Figure 5. State transition diagram with functions depending on Age, Male, BP (Blood Pressure). There are 3 disease states: Healthy, Sick, and Dead, where the Dead state is terminal. The yearly transition probabilities are: Healthy to Dead: Age/1000; Healthy to Sick: According to function F1 depending on Age and Male and BP; Sick to Healthy: 0.1; Sick to Dead: according to function F2 depending on Age and Male. Pre-Transition Rules: Age increased by 1 and BP by Age/10 each simulation cycle. Post-Transition Rules: Treatment $=$ BP > 140 , becomes 1 when BP crosses 140 threshold; BP $=$ BP $-$ Treatment $* 10$, meaning a drop of 10 once treatment is applied; CostThisYear $=$ Age $+$ Treatment $* 10$, cost depends on age and if treatment was taken; Cost $=$ Cost $+$ CostThisYear , it accumulates cost over time.
Initial conditions: Healthy = (50 Male, 50 Female with Age =1,2,. . . ,50 for each individual), BP =120, Sick = (0,0) and Dead = (0,0).
Output: Number of men and women in each disease state for years 1-10 and their ages and costs in each state. A stratified report by male and female and young up to age 30 and old above age 30 is produced.

The SBML standard is widely adopted and it is very active within the Multi-Scale Modeling community (`https://www.imagwiki.nibib.nih.gov/content/frequently-asked-questions-faq`) that tries to address different types of modeling that traverse scales, from cells to organs to populations. When modeling many types of systems in different scales, it is essential to have many modeling capabilities. SBML has 280 systems reported that support it as well as an established development process, specifications, and 2 annual meetings. This makes it an established infrastructure for modeling transport mechanism. Therefore enriching SBML to support microsimulation may make it an attractive candidate for adoption for modelers that need to support many modeling systems.

There are many modeling systems that support microsimulation as mentioned by a recent review in (27). However, we are aware of very few other similar approaches to allow model representation towards transport between systems. PharmML (12) is a close candidate that was already addressed in (14) and since it works together with the human readable MDL (13), we consider them together. And although PharmML has elements that address individual modeling and shows promise, it is far from the SBML level of adoption as easily demonstrated by the large variety and number of SBML models in `biomodels.net`. Another effort worth mentioning towards supporting communicating models is the ODD protocol (28) used to describe agent based models. However, this protocol, although helpful to convey models to humans, is not a model transport mechanism that allows transporting models between computing systems. If we move beyond biomedical models, another known modeling standard is Discrete Event System Specification (DEVS) that was introduced about four decades ago. DEVS allows formalism of a model as a set of states and transitions and just like SBML it has many extensions to the basic formulation and many implementations in different computing languages. DEVS basic formulation is very simple as can be seen in (29) and the authors see potential in this formulation that supports parallelism (30) that invites future exploration, yet despite its popularity, it is not widely used by biomedical communities.However, recent work connecting between DEVS and SBML (26) shows promise and contributes to the approach presented in this paper.

SBML will not replace existing modeling systems that model in a large variety of tools and languages, instead it presents a common reproducible standard that communities may choose to adopt. This paper may influence such adoption by presenting reproducible examples that others can follow.

Note that reproducibility has many facets. Different implementations of the models with different tools may generate different results. Even if two different tools are provided the same random numbers sequence to drive the stochastic model may generate different outcomes using different tools. Therefore, asking for the exact same output files generated by two systems is not practical. However, we do expect that the same tools after receiving the SBML file will be able to internally be able to repeat the same results given the same random seed and therefore be deterministic. We also expect that repetition of the same model simulations on different systems have to converge towards the same statistical solution as was demonstrated. In this paper we made an effort to include all possible information to allow reproducing our results including attaching code and describing computing system environment as well as archiving the output of the systems. However, the core idea is that we can represent the same model in SBML, which will allow future exchange between systems.

The ability to transport models between systems may encourage other modelers to adopt the ideas and follow these examples to create other microsimulation models in SBML. We aim toward chronic disease modelers, yet hope that this will eventually impact infectious disease modelers who many times use variations of SIR (Susceptible, Infectious, Recovered) models and their extensions (3). An example of SIR model implemented in SBML can be found in: `http://www.ebi.ac.uk/biomodels-main/MODEL1009230000`. However, it does not use utilize SBML Arrays which may open opportunities in the future to model interactions at the individual level.

Once disease models are implemented in SBML, it opens a multitude of software tool options for disease modelers and may have considerable impact on the field (in particular, a significant impact on model reproducibility). When model reproducibility is no longer an issue, model credibility will certainly increase.

Reproducibility Information

The following tools were used to generate the results in this paper: MIST Version 0.92.2.0 using Python 2.7.14 , Anaconda2-5.0.1 (64-bit) running on a Windows 10 (64 bit) machine; SBML files were generated with libsbml experimental version 5.16.0 for Python 2.7 64 bit running on a Windows 10 (64 bit) machine; SBML files were imported to iBioSim Version 3.0.0 – freely available for download at: `http://www.async.ece.utah.edu/ibiosim` – running on macOS Sierra. Model files and results can be obtained from the following GitHub repository: `https://github.com/Jacob-Barhak/DiseaseModelsSBML`

Acknowledgements

The authors from the University of Utah are supported by the National Science Foundation under Grants CCF-1218095 and CCF-1748200. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The arrays package support and its development within the JSBML API was supported by Google Summer of Code 2014. Many thanks to the SBML discussion groups and their users that helped with this work by participating in the discussion or providing solutions: Lucian Smith, Brett Olivier, Nicolas Le Novere, Fengkai Zhang, and Sarah Keating.

References

[1] Hayes AJ, Leal J, Gray AM, Holman RR, Clarke PM. UKPDS Outcomes Model 2: a new version of a model to simulate lifetime health outcomes of patients with type 2 diabetes mellitus using data from the 30year United Kingdom Prospective Diabetes Study: UKPDS 82. Diabetologia. 2013 Sep;56(9):1925–1933. Available from: `https://doi.org/10.1007/s00125-013-2940-y`.

[2] D'Agostino RB, Vasan RS, Pencina MJ, Wolf PA, Cobain M, Massaro JM, et al. General Cardiovascular Risk Profile for Use in Primary Care. Circulation. 2008;117(6):743–753. Available from: `http://circ.ahajournals.org/content/117/6/743`.

[3] Salem D, Smith R. A Mathematical Model of Ebola Virus Disease: Using Sensitivity Analysis to Determine Effective Intervention Targets. In: Proceedings of the Summer Computer Simulation Conference. SCSC '16. San Diego, CA, USA: Society for Computer Simulation International; 2016. p. 3:1–3:8. Available from: `http://dl.acm.org/citation.cfm?id=3015574.3015577`.

[4] Lasry A, Zaric GS, Carter MW. Multi-level resource allocation for HIV prevention: A model for developing countries. European Journal of Operational Research. 2007;180(2):786 – 799. Available from: `http://www.sciencedirect.com/science/article/pii/S0377221706002578`.

[5] Leff HS, Dada M, Graves SC. An LP Planning Model for a Mental Health Community Support System. Management Science. 1986;32(2):139–155. Available from: `https://doi.org/10.1287/mnsc.32.2.139`.

[6] Scholz S, Batram M, Greiner W. The SILAS Model: Sexual Infections As Large-scale Agent-based Simulation. In: Proceedings of the Conference on Summer Computer Simulation. SummerSim '15. San Diego, CA, USA: Society for Computer Simulation International; 2015. p. 1–6. Available from: `http://dl.acm.org/citation.cfm?id=2874916.2874970`.

[7] Hippisley-Cox J, Coupland C, Vinogradova Y, Robson J, May M, Brindle P. Derivation and validation of QRISK, a new cardiovascular disease risk score for the United Kingdom: prospective open cohort study. BMJ. 2007;335(7611):136. Available from: `http://www.bmj.com/content/335/7611/136`.

[8] Hippisley-Cox J, Coupland C, Vinogradova Y, Robson J, Minhas R, Sheikh A, et al. Predicting cardiovascular risk in England and Wales: prospective derivation and validation of QRISK2. BMJ. 2008;336(7659):1475–1482. Available from: `http://www.bmj.com/content/336/7659/1475`.

[9] Isaman DJ, Barhak J, Ye W. Indirect estimation of a discrete-state discrete-time model using secondary data analysis of regression data. Statistics in medicine. 2009;28(16):2095–2115.

[10] Barhak J, Isaman DJ, Ye W, Lee D. Chronic disease modeling and simulation software. Journal of biomedical informatics. 2010;43(5):791–799.

[11] Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics. 2003 Mar;19(4):524–531.

[12] Swat M, Moodie S, Wimalaratne S, Kristensen N, Lavielle M, Mari A, et al. Pharmacometrics Markup Language (PharmML): opening new perspectives for model exchange in drug development. CPT: pharmacometrics & systems pharmacology. 2015;4(6):316–319.

[13] Smith MK, Moodie SL, Bizzotto R, Blaudez E, Borella E, Carrara L, et al. Model Description Language (MDL): A Standard for Modeling and Simulation. CPT: Pharmacometrics and Systems Pharmacology. 2017;6(10):647–650. Available from: `http://dx.doi.org/10.1002/psp4.12222`.

[14] Smith L, Swat MJ, Barhak J. Sharing formats for disease models. In: Proceedings of the Summer Computer Simulation Conference. Society for Computer Simulation International; 2016. p. 5.

[15] Barhak J. MIST: Micro-Simulation Tool to Support Disease Modeling. SciPy. 2013;.

[16] Hucka M, Nickerson DP, Bader GD, Bergmann FT, Cooper J, Demir E, et al. Promoting Coordinated Development of Community-Based Information Standards for Modeling in Biology: The COMBINE Initiative. Frontiers in Bioengineering and Biotechnology. 2015;3:19. Available from: `https://www.frontiersin.org/article/10.3389/fbioe.2015.00019`.

[17] Davidich MI, Bornholdt S. Boolean network model predicts cell cycle sequence of fission yeast. PloS one. 2008;3(2):e1672.

[18] Goss PJ, Peccoud J. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. Proceedings of the National Academy of Sciences. 1998;95(12):6750–6755.

[19] Madsen C, Zhang Z, Roehner N, Winstead C, Myers C. Stochastic Model Checking of Genetic Circuits. J Emerg Technol Comput Syst. 2014 Dec;11(3):23:1–23:21. Available

from: http://doi.acm.org/10.1145/2644817.

[20] Smith LP, Hucka M, Hoops S, Finney A, Ginkel M, Myers CJ, et al. SBML level 3 package: Hierarchical model composition, version 1 release 3. Journal of Integrative Bioinformatics (JIB). 2015;12(2):603–659.

[21] Olivier BG, Bergmann FT. The systems biology markup language (SBML) level 3 package: Flux balance constraints. Journal of Integrative Bioinformatics (JIB). 2015;12(2):660–690.

[22] Gauges R, Rost U, Sahle S, Wengler K, Bergmann FT. The Systems Biology Markup Language (SBML) Level 3 Package: Layout, Version 1 Core. Journal of Integrative Bioinformatics (JIB). 2015;12(2):550–602.

[23] Watanabe L, Myers CJ. Efficient Analysis of Systems Biology Markup Language Models of Cellular Populations Using Arrays. ACS Synthetic Biology. 2016;5(8):835–841. PMID: 26912276. Available from: http://dx.doi.org/10.1021/acssynbio.5b00242.

[24] Bornstein BJ, Keating SM, Jouraku A, Hucka M. LibSBML: an API Library for SBML. Bioinformatics. 2008;24(6):880–881. Available from: +http://dx.doi.org/10.1093/bioinformatics/btn051.

[25] Rodriguez N, Thomas A, Watanabe L, Vazirabad IY, Kofia V, Gómez HF, et al. JSBML 1.0: providing a smorgasbord of options to encode systems biology models. Bioinformatics. 2015;31(20):3383–3386. Available from: +http://dx.doi.org/10.1093/bioinformatics/btv341.

[26] Belloli L, Wainer G, Najmanovich R. Parsing and Model Generation for Biological Processes. In: Proceedings of the Symposium on Theory of Modeling & Simulation. TMS-DEVS '16. San Diego, CA, USA: Society for Computer Simulation International; 2016. p. 21:1–21:6. Available from: http://dl.acm.org/citation.cfm?id=2975389.2975410.

[27] Sorensen RJD, Flaxman AD, Deason A, Mumford JE, Eldrenkamp E, Moses M, et al. Microsimulation Models for Cost-effectiveness Analysis: A Review and Introduction to CEAM. In: Proceedings of the Summer Simulation Multi-Conference. SummerSim '17. San Diego, CA, USA: Society for Computer Simulation International; 2017. p. 32:1–32:11. Available from: http://dl.acm.org/citation.cfm?id=3140065.3140097.

[28] Grimm V, Berger U, Bastiansen F, Eliassen S, Ginot V, Giske J, et al. A standard protocol for describing individual-based and agent-based models. Ecological Modelling. 2006;198(1):115 – 126. Available from: http://www.sciencedirect.com/science/article/pii/S0304380006002043.

[29] Tendeloo YV, Vangheluwe H. Introduction to Parallel DEVS Modelling and Simulation; 2017. Accessed: 2018-02-12. http://msdl.cs.mcgill.ca/people/yentl/papers/2017-DEVSTutorial.pdf.

[30] Zeigler BP. Using the Parallel DEVS Protocol for General Robust Simulation with Near Optimal Performance. Computing in Science Engineering. 2017 May;19(3):68–77.

Author Biographies

Leandro Watanabe is a PhD candidate at the University of Utah, Department of Electrical and Computer Engineering, USA.

Jacob Barhak has diverse international background in engineering, computing science, and disease modeling. He was educated at the Technion Israel Institute of Technology and worked at the University of Michigan from 2003 to 2012. The Reference Model for disease progression is his main project since 2012 and he currently pursues this development effort as an independent researcher. He is the developer of the Micro Simulation Tool (MIST). Dr. Barhak is an advocate of non-blind public scientific review processes and active in several forums, including SummerSim and the Population Modeling working group. He is active within the python community and runs the Austin Evening of Python Coding meetup.

Chris Myers is a full professor at the University of Utah, Department of Electrical and Computer Engineering, USA, and he has adjunct appointments with the School of Computing and the Department of Biomedical Engineering.

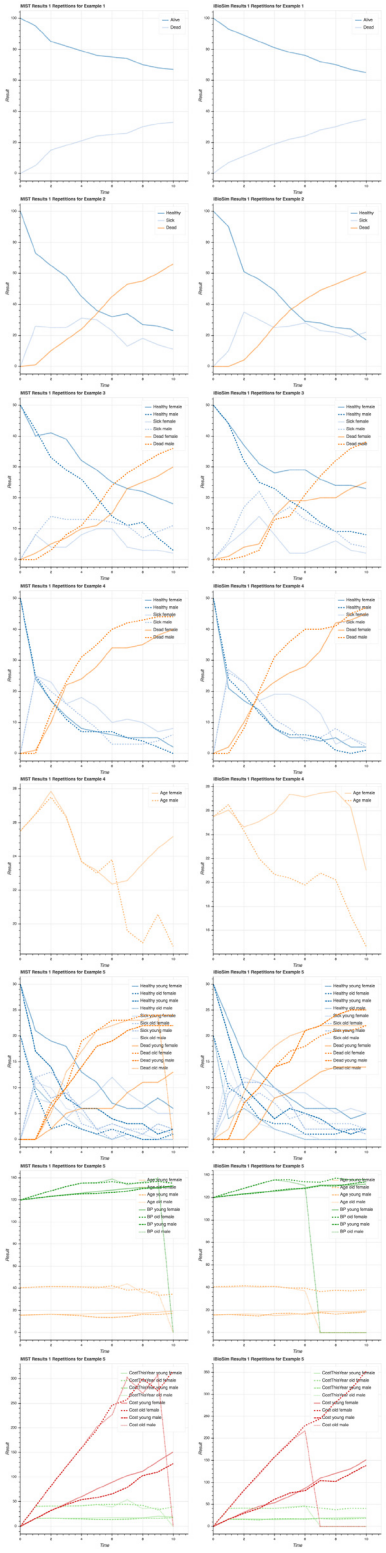


Figure 6. This figure shows the results comparison between MIST and iBioSim for one run.

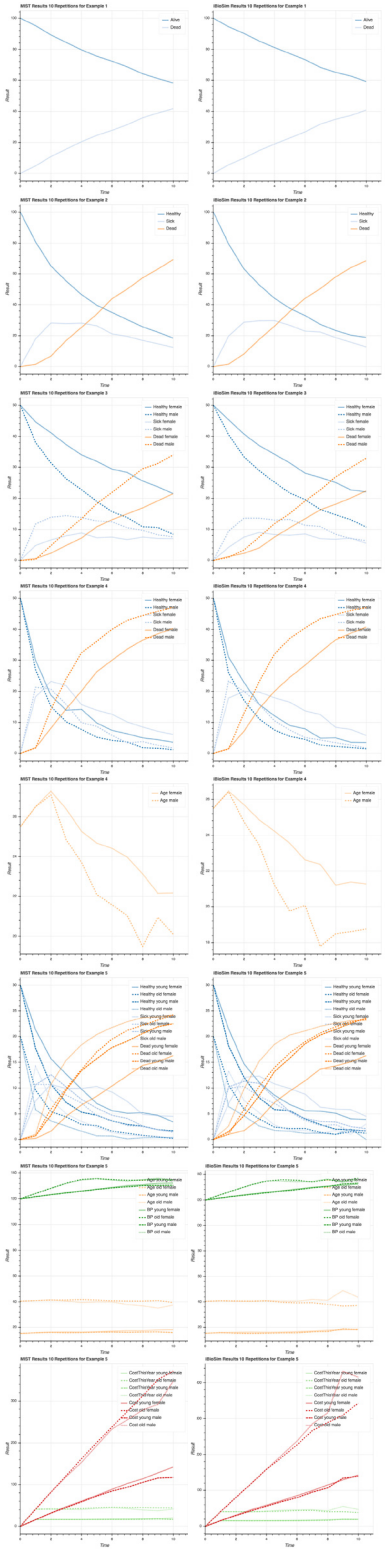


Figure 7. This figure shows the results comparison between MIST and iBioSim for ten runs.

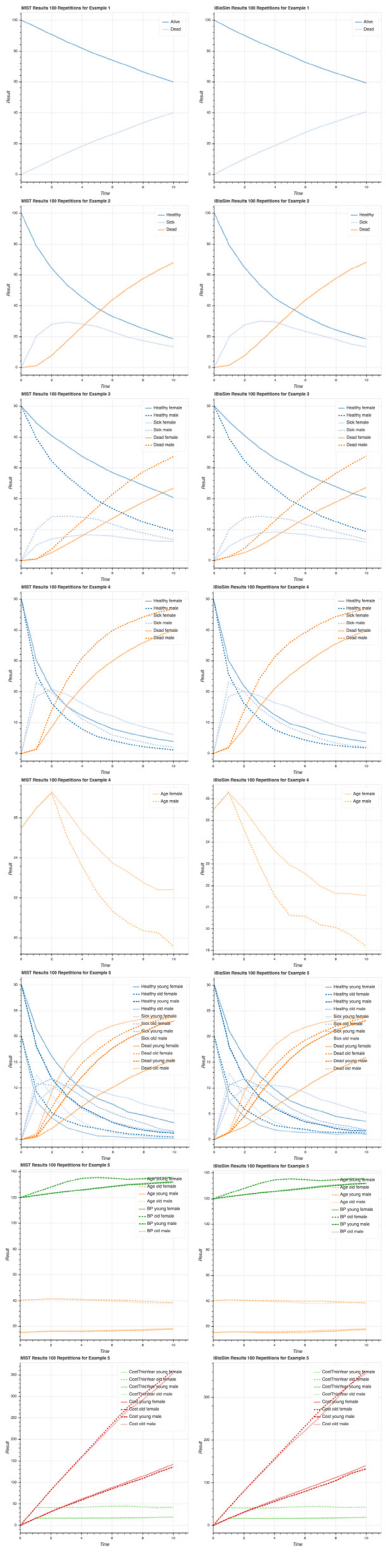


Figure 8. This figure shows the results comparison between MIST and iBioSim for 100 runs.

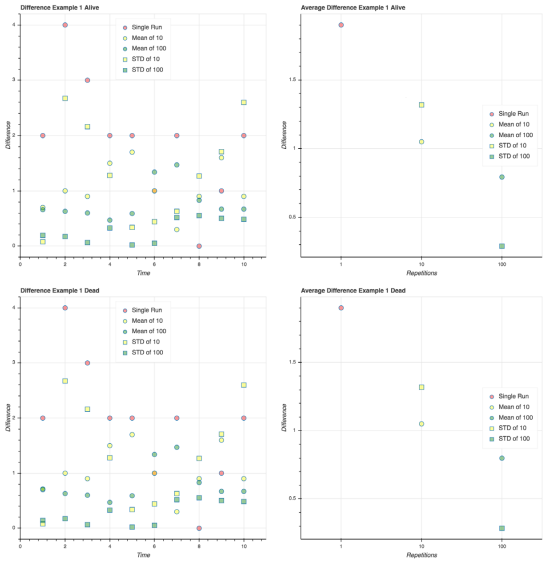


Figure 9. Statistical analysis for example 1.

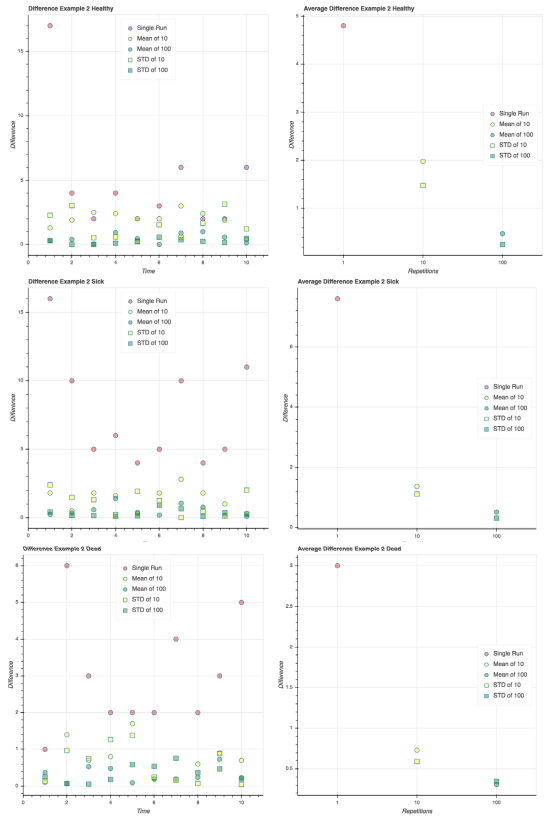


Figure 10. Statistical analysis for example 2.

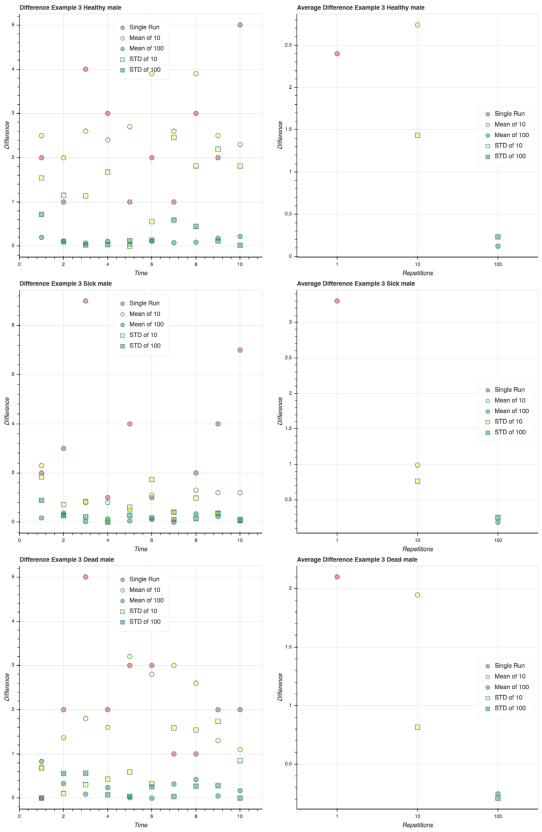


Figure 11. Statistical analysis for males in example 3.

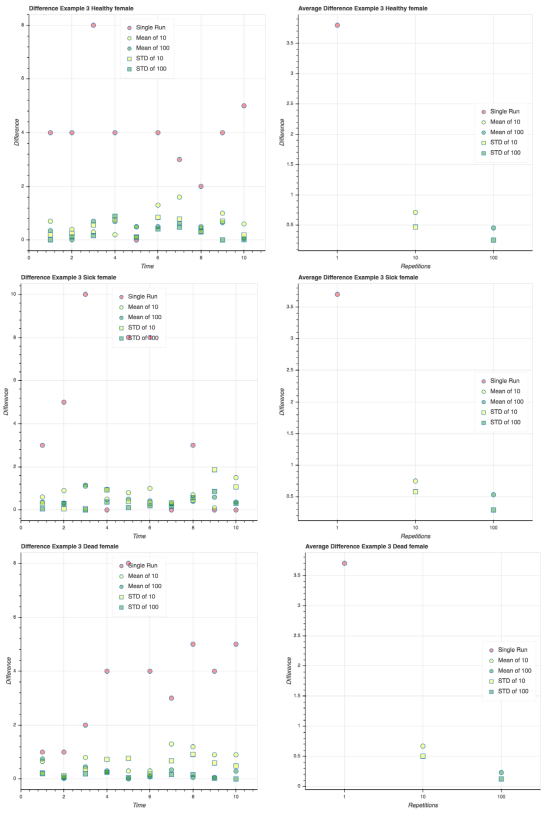


Figure 12. Statistical analysis for females in example 3.

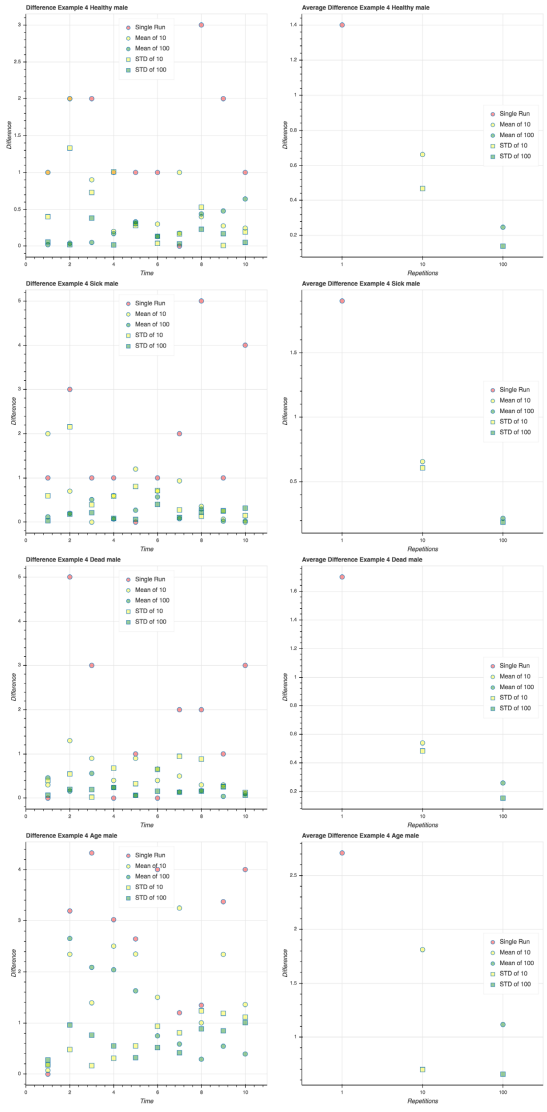


Figure 13. Statistical analysis for males in example 4.

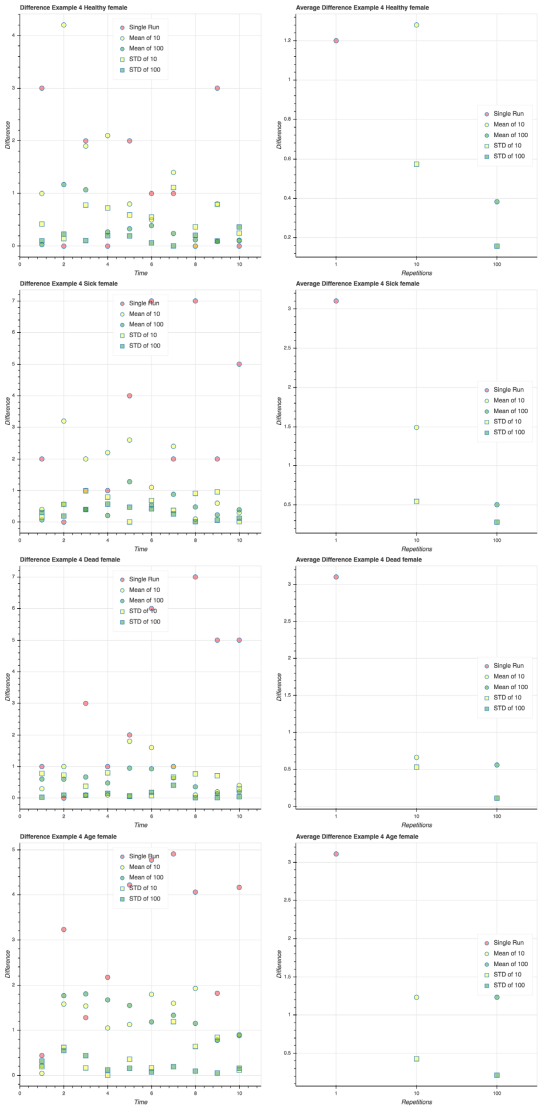


Figure 14. Statistical analysis for females in example 4.

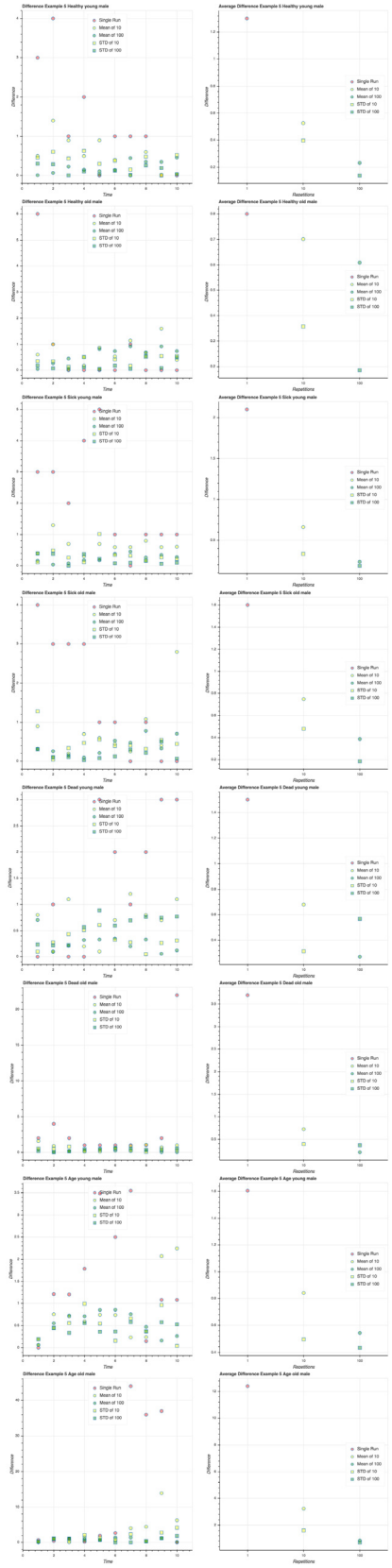


Figure 15. Statistical analysis for the average age of males alive and the number of healthy, dead, and sick males in example 5.

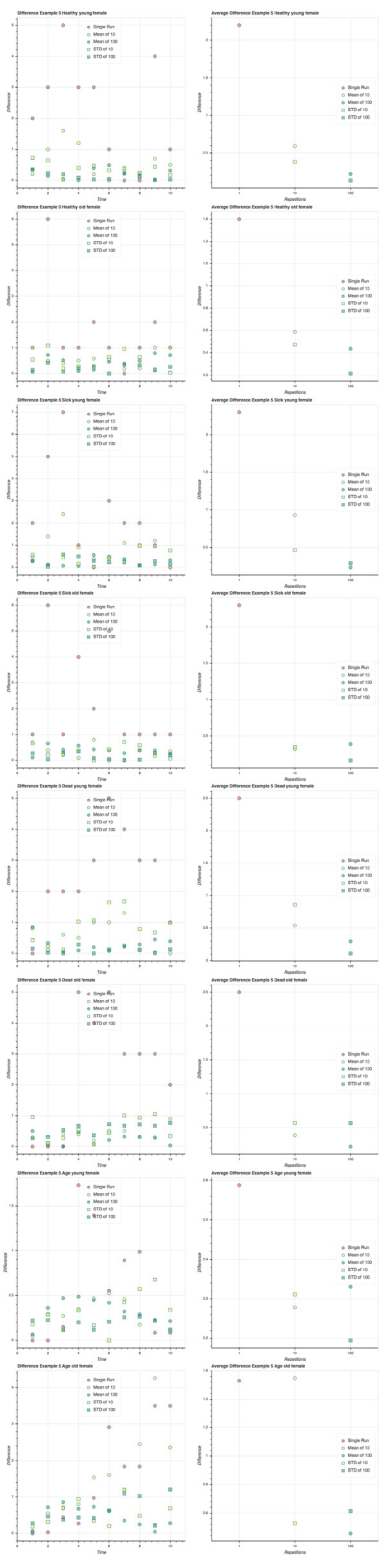


Figure 16. Statistical analysis for the average age of males alive and the number of healthy, dead, and sick females in example 5.

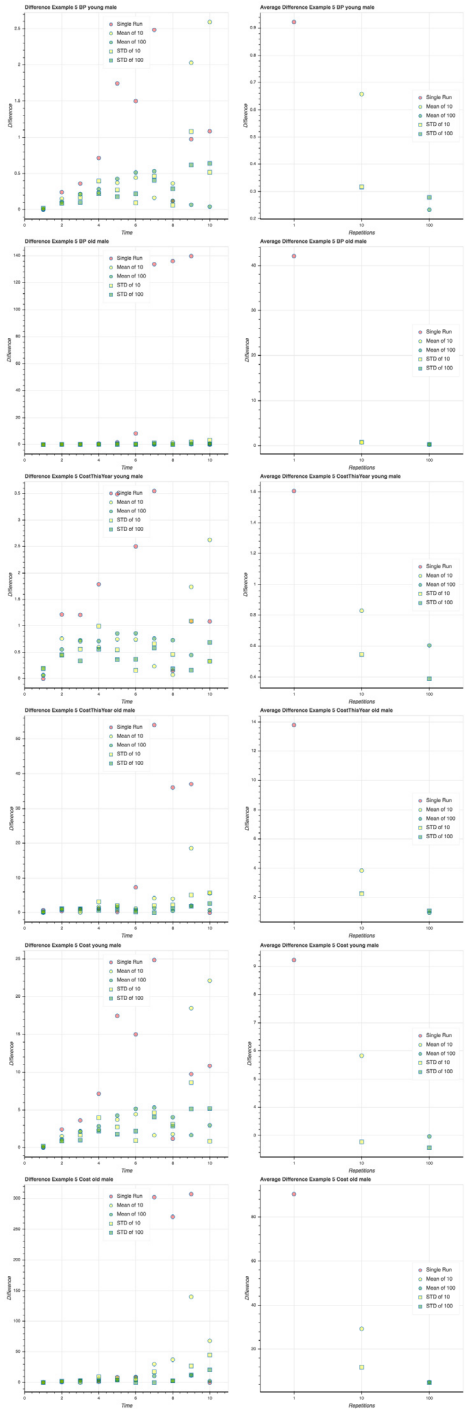


Figure 17. Statistical analysis for blood pressure, cost, and cost this year for males in example 5.

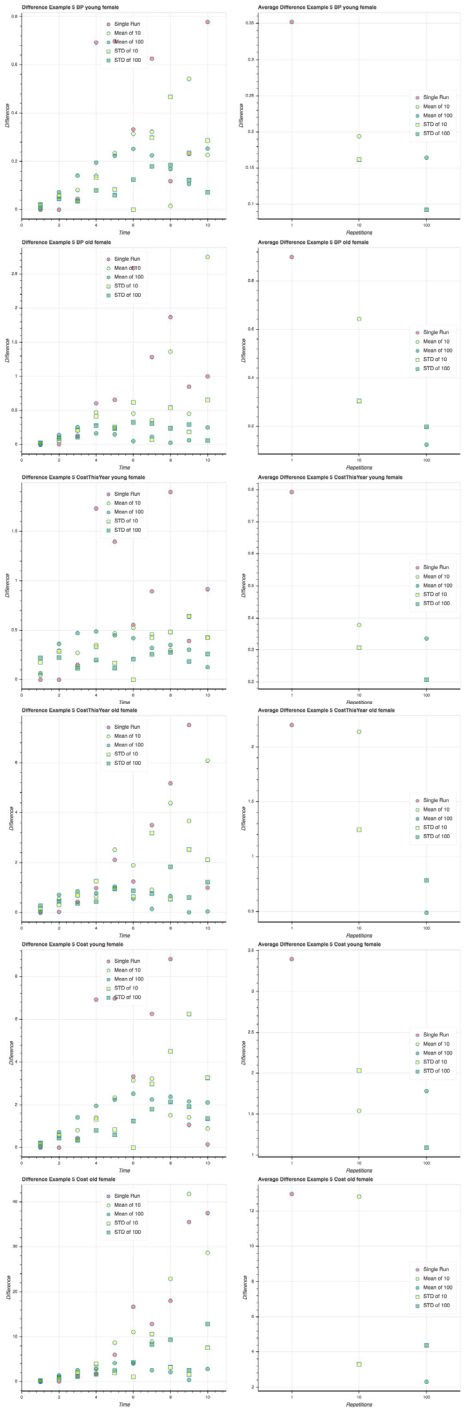


Figure 18. Statistical analysis for blood pressure, cost, and cost this year for females in example 5.