Simulation

Applications

Simulation: Transactions of the Society for Modeling and Simulation International
1–36
© The Author(s) 2018
DOI: 10.1177/0037549718793214
journals.sagepub.com/home/sim
SAGE SCS

Toward 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 such tools 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 between 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.

Keywords
Disease modeling, microsimulation, reproducibility, SBML, arrays package

1 Introduction

Disease models attempt to explain phenomena observed by clinical trials and follow-up of patient populations through time. These 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 these phenomena using mathematical and statistical equations or other programmatic constructs.

In the past, differential equations have been used, which are still 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 codes is rare.[7–10] The norm is still publication of descriptive-only models in papers in which they appear, and only rarely do authors attempt to publish in such 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, Hayes et al.[1]).

The Mount Hood Diabetes Challenge highlighted this reproducibility problem.[11] The challenge in 2016 revolved around reproducing models from two published papers. A number of modeling teams around the world attempted to reproduce these published models, and 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)[12]

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

Corresponding author:
Leandro Watanabe, Department, of Electrical and Computer Engineering, University of Utah, 50 S. Central Campus Drive, MEB Rm 2110, Salt Lake City, UT, 84112, USA.
Email: l.watanabe@utah.edu

Table 1. SBML events for Example 1.

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

and associated languages, such as PharmML[13] and its counterpart Model Description Language (MDL),[14] are designed for. This paper continues the first attempt to reproduce disease models in such modeling languages that started with Smith et al.,[15] which demonstrated how a disease model can be reproduced in three languages: SBML, PharmML, and MIcro Simulation Tool (MIST).[16] When the first paper was written, SBML capabilities lacked a definition of more complex models using microsimulation. As SBML is a community-driven standard, there are biannual meetings about how SBML can be improved.[17] Recently, the community talked about how to represent agent-based models in SBML at the COMBINE 2017 meeting, but such representation has not reached a consensus. Nonetheless, SBML has evolved, with the recent introduction of the SBML Arrays draft package specification, which 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.

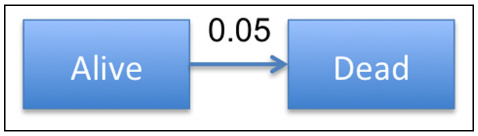


Figure 1. State transition diagram of a simple Markov model. The model uses two disease states, *alive* and *dead*, where the dead state terminates simulation. The yearly probability of transition between *alive* and *dead* states is 0.05.
Initial conditions: 100 people start in *alive* and none in *dead*.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

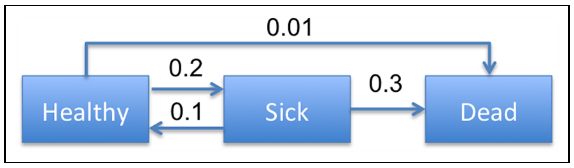


Figure 2. State transition diagram of a three-state Markov model. There are three disease states: healthy, sick, and dead, where the dead state is terminal. The yearly transition probabilities are: healthy to dead, 0.01; healthy to sick, 0.2; sick to healthy, 0.1; sick to dead, 0.3.
Initial conditions: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

2 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,[18] Petri nets,[19] and Markov chains,[20] among others. In addition, SBML has support for package extensions that enhance the standard even further with new constructs beyond the core constructs.[21–23] In particular, the SBML Arrays package enables the instantiation of a specified number of identical model objects, facilitating the representation of populations.[24] SBML Arrays has been implemented within the C/C ++ library of SBML, called libSBML,[25] and the Java library of SBML, called JSBML.[26] 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

Table 2. SBML events for Example 2.

	Trigger	Assignments
0	Instruction Number[d0] = 0	Time[d0] = Time[d0] + 1 Instruction Number[d0] = 0.1
1	Instruction Number[d0] = 0.1	Instruction Number[d0] = 0.2
2	Instruction Number[d0] = 0.2 ∧ Dead[d0] = 0 ∧ Time[d0] < 10	Instruction Number[d0] = 1
3	Instruction Number[d0] = 1	Random[d0] = uniform(0,1) Instruction Number[d0] = 1.5

4	Instruction Number[d0] = 1.5 ∧ Healthy[d0] = 1 = 1 ∧ Random[d0] ≥ 0 ∧ Random[d0] < (0 + 0.01)	Healthy[d0] = 0
		Dead[d0] = 1
		Instruction Number[d0] = 0

5	Instruction Number[d0] = 1.5 ∧ Healthy[d0] = 1 = 1 ∧ Random[d0] ≥ (0 + 0.01) ∧ Random[d0] < (0 + 0.01) + 0.2	Healthy[d0] = 0
		Sick[d0] = 1
		Instruction Number[d0] = 0

6	Instruction Number[d0] = 1.5 ∧ Healthy[d0] = 1 ∧ Random[d0] ≥ (0 + 0.01) + 0.2 ∧ Random[d0] < 1	Instruction Number[d0] = 0
7	Instruction Number[d0] = 1	Random[d0] = uniform(0,1) Instruction Number[d0] = 1.5

8	Instruction Number[d0] = 1.5 ∧ Sick[d0] = 1 = 1 ∧ Random[d0] ≥ 0 ∧ Random[d0] < (0 + 0.1)	Sick[d0] = 0
		Healthy[d0] = 1
		Instruction Number[d0] = 0

9	Instruction Number[d0] = 1.5 ∧ Sick[d0] = 1 = 1 ∧ Random[d0] ≥ (0 + 0.1) ∧ Random[d0] < (0 + 0.1) + 0.3	Sick[d0] = 0
		Dead[d0] = 1
		Instruction Number[d0] = 0

10	Instruction Number[d0] = 1.5 ∧ Sick[d0] = 1 ∧ Random[d0] ≥ (0 + 0.1) + 0.3 ∧ Random[d0] < 1	Instruction Number[d0] = 0

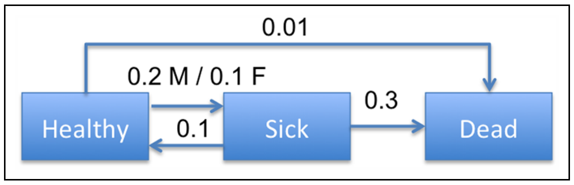


Figure 3. State transition diagram of a simple Markov model. There are three disease states: healthy, sick, and dead, where the dead state is terminal. The yearly transition probabilities are: healthy to dead, 0.01; healthy to sick, 0.2 for male and 0.1 for female; sick to healthy, 0.1; sick to dead,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

individual, where each index in the array represents a single person. Each possible state for an individual (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.

3 Disease modeling examples

To illustrate the requirements for disease models, the following sections present several abstract examples that are successfully implemented using SBML coupled with the SBML Arrays package. We start with the same examples given in Smith et al.[15] 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.

Table 3. SBML events for Example 3.

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

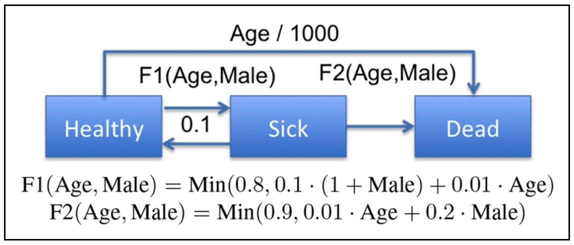

Age / 1000
F1(Age,Male)
F2(Age,Male)
Healthy
0.1
Sick
Dead
F1(Age, Male) = Min(0.8, 0.1 · (1 + Male) + 0.01 · Age)
F2(Age, Male) = Min(0.9, 0.01 · Age + 0.2 · Male)

Figure 4. State transition model dependent on changing parameters. There are three 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 men, 50 women with age =1, 2, …, 50 for each individual); sick = (0, 0); dead = (0, 0).
Output: Number of men and women in each disease state for years 1–10 and their ages in each state.

3.1 *Example 1: Simple example*

The first simple example (depicted in Figure 1) can be modeled as a cohort model, as demonstrated before in Smith et al.,[15] 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 being in years after death, effectively diminishing cohort size.

This entire simulation is implemented as SBML events, as can be seen in Table 1. SBML events are triggered if

Table 4. SBML events for Example 4.

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

their trigger condition is previously false and then evaluate 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 forces discrete times for the sequence of occurrences. Also note that the model time and SBML implementation time are different. In this example, there is a header of events enumerated #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 whether 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*.

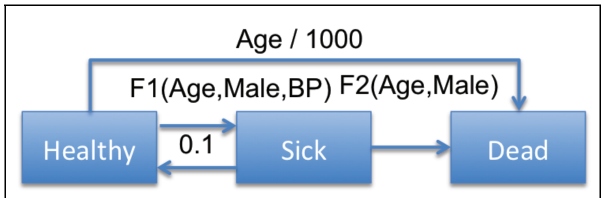
Table 5. SBML events for Example 5.

	Trigger	Assignments
0	$\text{Instruction Number}[d0] = 0$	$\text{Time}[d0] = \text{Time}[d0] + 1$ $\text{Instruction Number}[d0] = 0.1$
1	$\text{Instruction Number}[d0] = 0.1$	$\text{Instruction Number}[d0] = 0.2$
2	$\text{Instruction Number}[d0] = 0.2 \wedge \text{Dead}[d0] = 0 \wedge \text{Time}[d0] < 10$	$\text{Instruction Number}[d0] = 1$
3	$\text{Instruction Number}[d0] = 1$	$\text{Random}[d0] = \text{uniform}(0,1)$ $\text{Instruction Number}[d0] = 1.5$
4	$\text{Instruction Number}[d0] = 1.5 \wedge 1$	$\text{Age}[d0] = \text{Age}[d0] + 1$ $\text{Instruction Number}[d0] = 2$
5	$\text{Instruction Number}[d0] = 1.5 \wedge 0$	$\text{Instruction Number}[d0] = 2$
6	$\text{Instruction Number}[d0] = 2$	$\text{Random}[d0] = \text{uniform}(0,1)$ $\text{Instruction Number}[d0] = 2.5$
7	$\text{Instruction Number}[d0] = 2.5 \wedge 1$	$\text{Instruction Number}[d0] = 3$ $\text{BP}[d0] = \text{BP}[d0] + \frac{\text{Age}[d0]}{10}$
8	$\text{Instruction Number}[d0] = 2.5 \wedge 0$	$\text{Instruction Number}[d0] = 3$
9	$\text{Instruction Number}[d0] = 3$	$\text{Random}[d0] = \text{uniform}(0,1)$ $\text{Instruction Number}[d0] = 3.5$
10	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Healthy}[d0] = 1 \wedge \text{Random}[d0] \geq 0 \wedge \text{Random}[d0] < (0 + \frac{\text{Age}[d0]}{1000})$	$\text{Healthy}[d0] = 0$ $\text{Dead}[d0] = 1$ $\text{Instruction Number}[d0] = 4$
11	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Healthy}[d0] = 1 \wedge \text{Random}[d0] \geq (0 + \frac{\text{Age}[d0]}{1000}) \wedge \text{Random}[d0] < (0 + \frac{\text{Age}[d0]}{1000}) + \min(0.8, 0.1 * (1 + \text{Male}[d0]) + 0.01 * \text{Age}[d0] + (\frac{\text{BP}[d0] - 120}{100})^2)$	$\text{Sick}[d0] = 1$ $\text{Healthy}[d0] = 0$ $\text{Instruction Number}[d0] = 4$
12	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Healthy}[d0] = 1 \wedge \text{Random}[d0] \geq (0 + \frac{\text{Age}[d0]}{1000}) + \min(0.8, 0.1 * (1 + \text{Male}[d0]) + 0.01 * \text{Age}[d0] + (\frac{\text{BP}[d0] - 120}{100})^2) \wedge \text{Random}[d0] < 1$	$\text{Instruction Number}[d0] = 4$
13	$\text{Instruction Number}[d0] = 3$	$\text{Random}[d0] = \text{uniform}(0,1)$ $\text{Instruction Number}[d0] = 3.5$
14	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Sick}[d0] = 1 \wedge \text{Random}[d0] \geq 0 \wedge \text{Random}[d0] < (0 + 0.1)$	$\text{Sick}[d0] = 0$ $\text{Healthy}[d0] = 1$ $\text{Instruction Number}[d0] = 4$
15	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Sick}[d0] = 1 \wedge \text{Random}[d0] \geq (0 + 0.1) \wedge \text{Random}[d0] < (0 + 0.1) + \min(0.9, 0.2 * \text{Male}[d0] + 0.01 * \text{Age}[d0])$	$\text{Sick}[d0] = 0$ $\text{Dead}[d0] = 1$ $\text{Instruction Number}[d0] = 4$
16	$\text{Instruction Number}[d0] = 3.5 \wedge \text{Sick}[d0] = 1 \wedge \text{Random}[d0] < (0 + 0.1) + \min(0.9, 0.2 * \text{Male}[d0] + 0.01 * \text{Age}[d0]) \wedge \text{Random}[d0] < 1$	$\text{Instruction Number}[d0] = 4$
17	$\text{Instruction Number}[d0] = 4$	$\text{Random}[d0] = \text{uniform}(0,1)$ $\text{Instruction Number}[d0] = 4.5$
18	$\text{Instruction Number}[d0] = 4.5 \wedge 1$	$\text{Treatment}[d0] = \text{Age}[d0] + 1$ $\text{Instruction Number}[d0] = 5$
19	$\text{Instruction Number}[d0] = 4.5 \wedge 0$	$\text{Instruction Number}[d0] = 5$

(continued)

Table 5. Continued

	Trigger	Assignments
20	Instruction Number[d0] = 5	Random[d0] = uniform(0,1) Instruction Number[d0] = 5.5
21	Instruction Number[d0] = 5.5 ∧ 1	BP[d0] = BP[d0] − Treatment[d0] * 10 Instruction Number[d0] = 6
22	Instruction Number[d0] = 5.5 ∧ 0	Instruction Number[d0] = 6
23	Instruction Number[d0] = 6	Instruction Number[d0] = 6
24	Instruction Number[d0] = 6.5 ∧ 1	CostThisYear[d0] = Age[d0] + Treatment[d0] * 10 Instruction Number[d0] = 7
25	Instruction Number[d0] = 6.5 ∧ 0	Instruction Number[d0] = 7
26	Instruction Number[d0] = 7	Instruction Number[d0] = 7.5
27	Instruction Number[d0] = 7.5 ∧ 1	Cost[d0] = Cost[d0] + CostThisYear[d0] Instruction Number[d0] = 0
28	Instruction Number[d0] = 7.5 ∧ 0	Instruction Number[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 of *age, male*, and *blood pressure* (BP). There are three 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, male*, and *blood pressure* 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 and blood pressure by age/10 each simulation cycle.
Post-transition rules: Treatment = BP > 140, becomes 1 when blood pressure crosses the 140 threshold;
BP = BP − Treatment * 10, indicating a drop of 10 once treatment is applied; Cost This Year = Age + Treatment * 10, cost depends on age and whether treatment was taken;
Cost = Cost + Cost This Year, accumulates cost over time.
Initial conditions: Healthy = (50 men, 50 women with age = 1, 2, …, 50 for each individual); blood pressure =120; sick = (0,0); 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 sex and age (up to 30 and above 30) is produced.

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, which automatically generates any arbitrary number of individual copies, which is 100 for the examples shown.

3.2 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 events #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 must be checked: taking the first transition, taking 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.

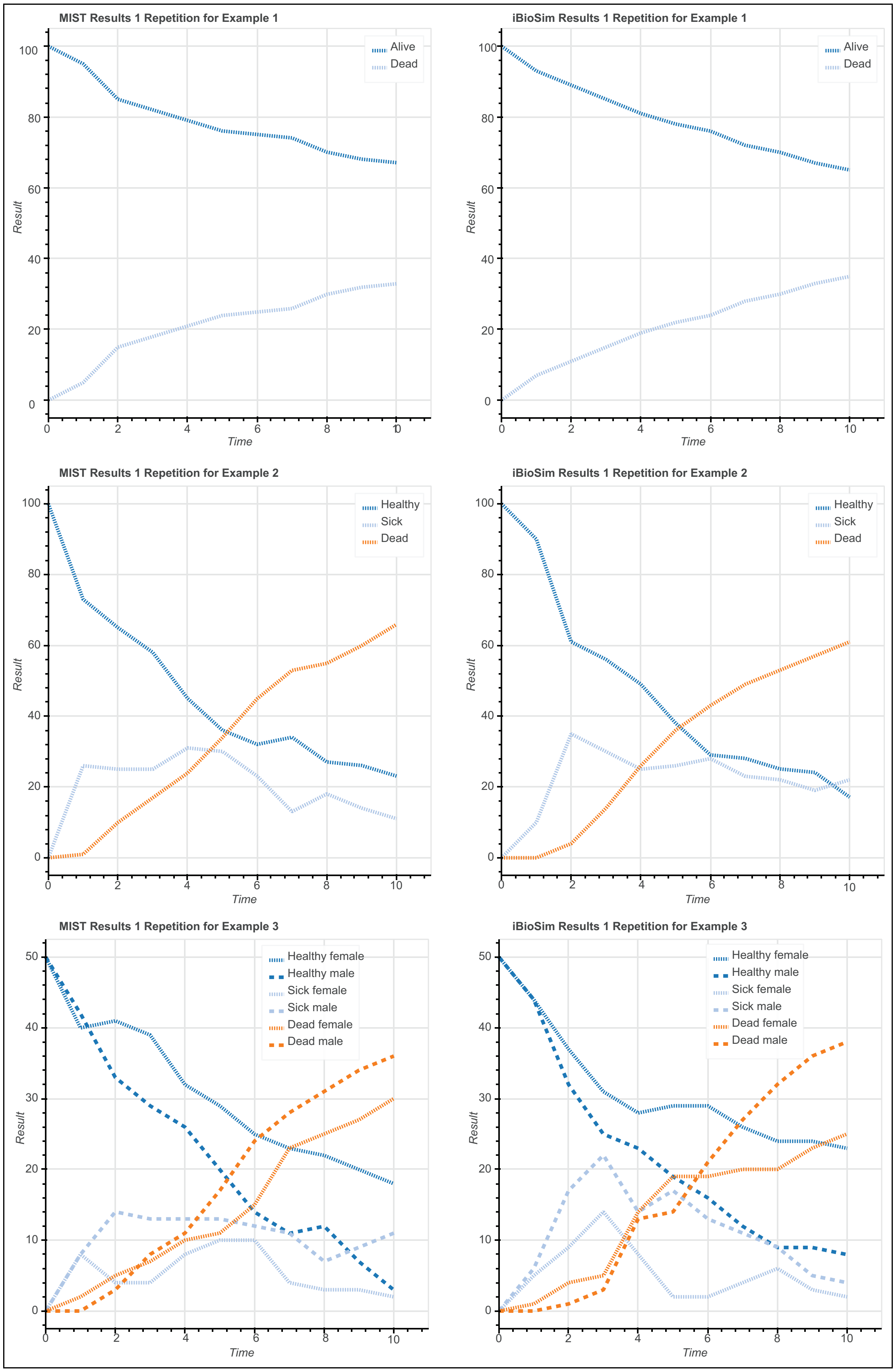


Figure 6. Continued

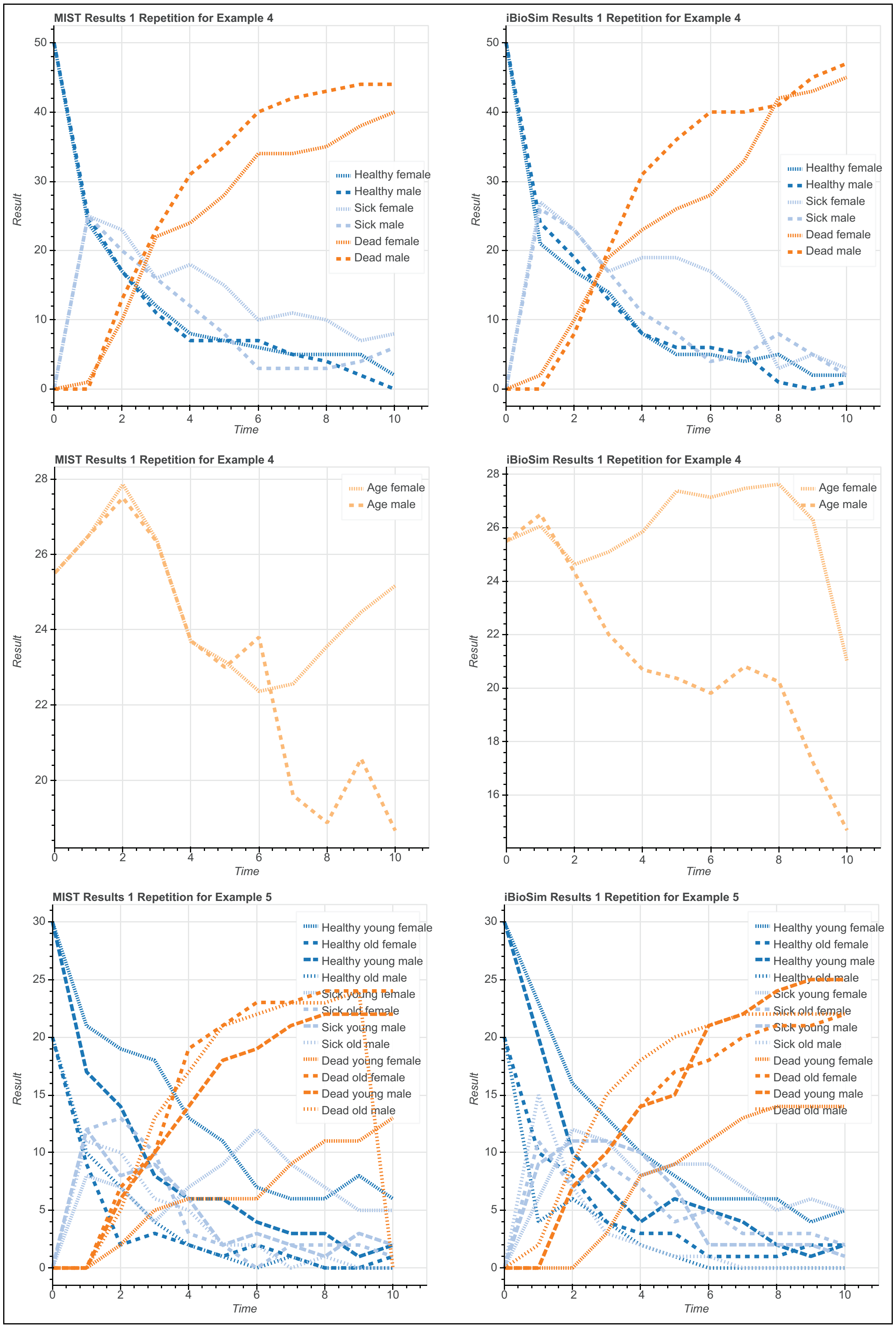


Figure 6. Continued

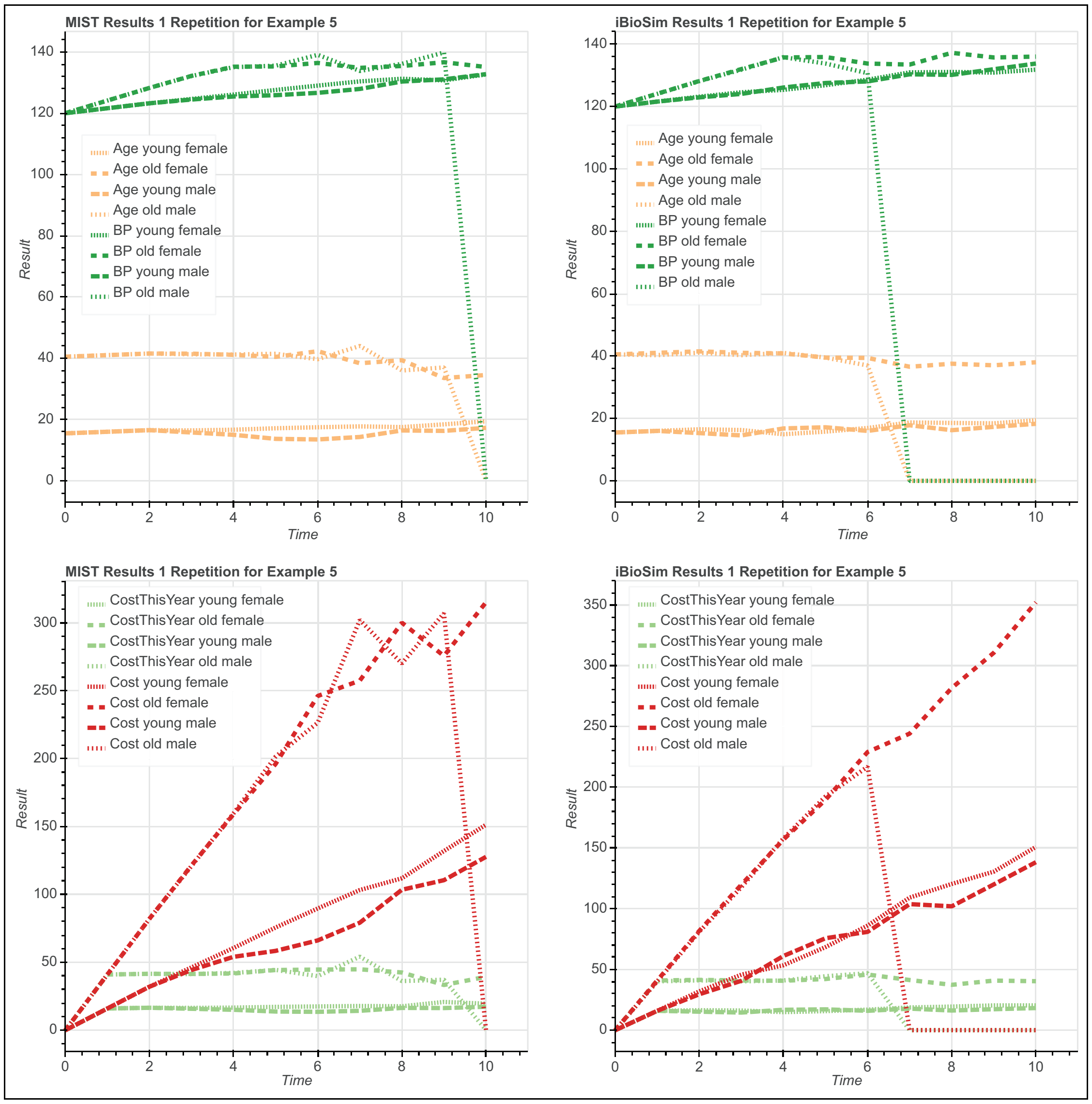


Figure 6. Results comparison between MIST and iBioSim for one run.

3.3 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, men become sick with higher probability than women and therefore simulation should show a higher sickness and death rate among men. 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.

3.4 Example 4: State transition model dependent on changing parameters

This example (depicted in Figure 4) can no longer be implemented using Markov cohort models, owing 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 play 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,

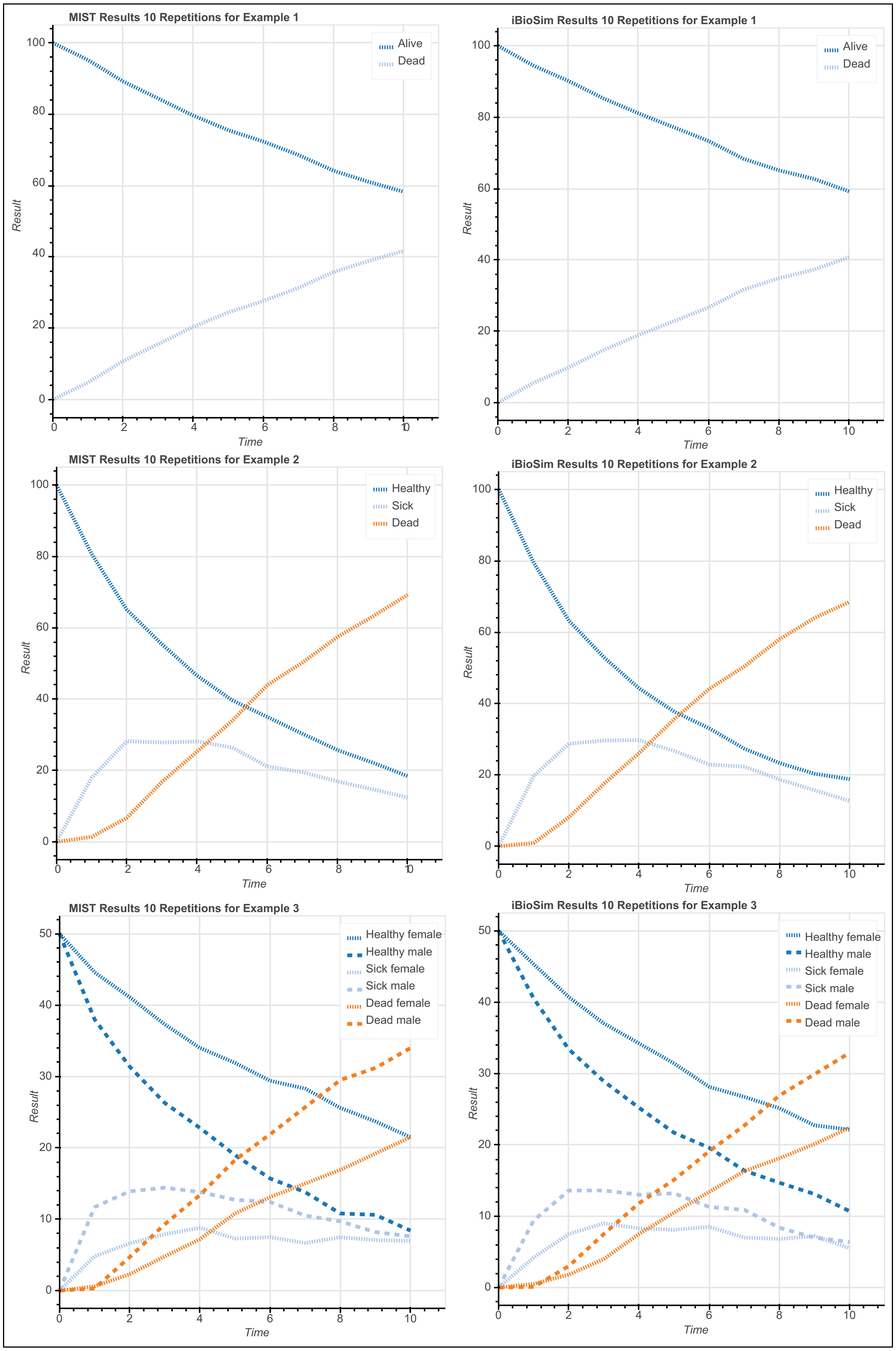


Figure 7. Continued

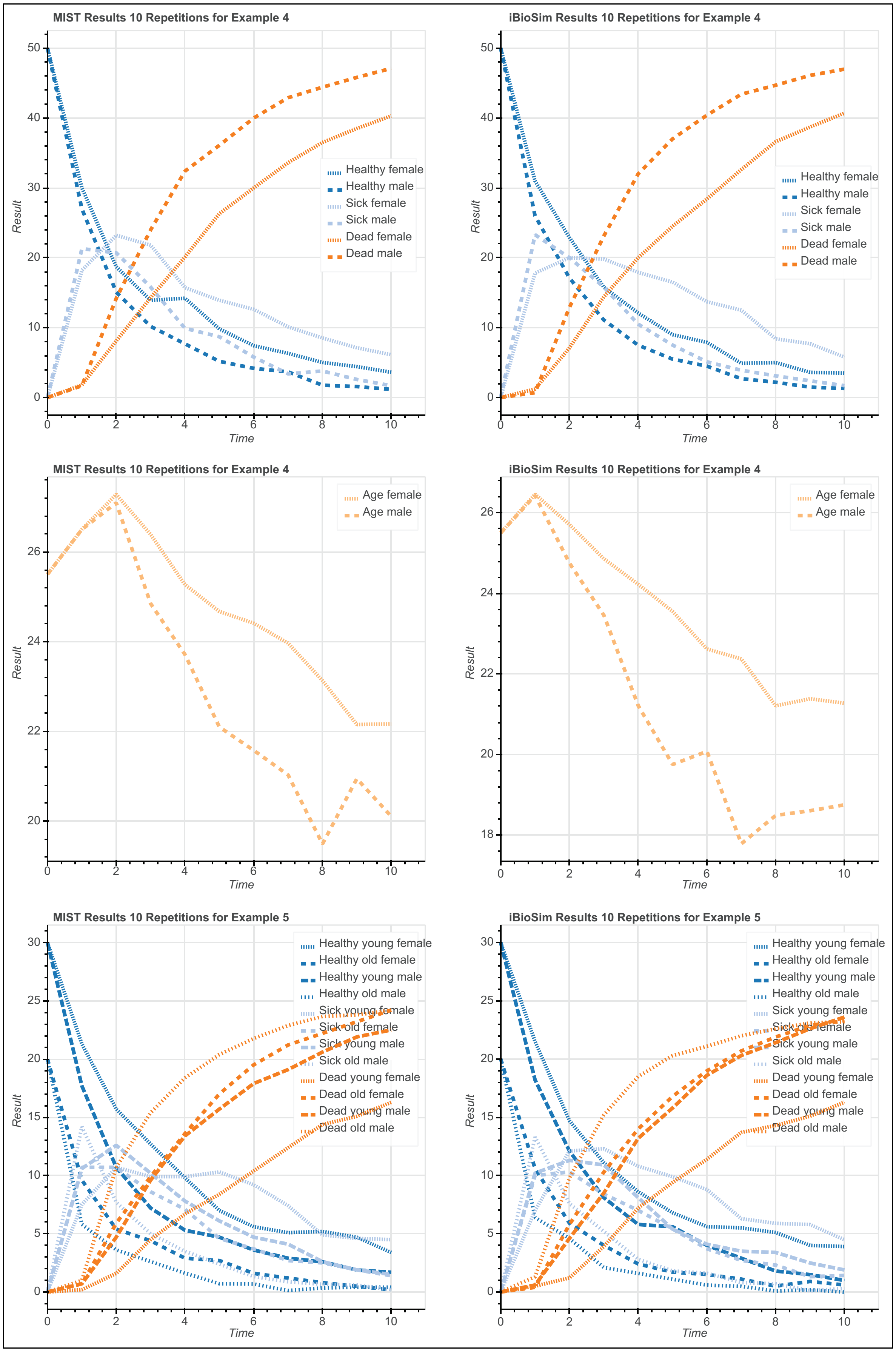


Figure 7. Continued

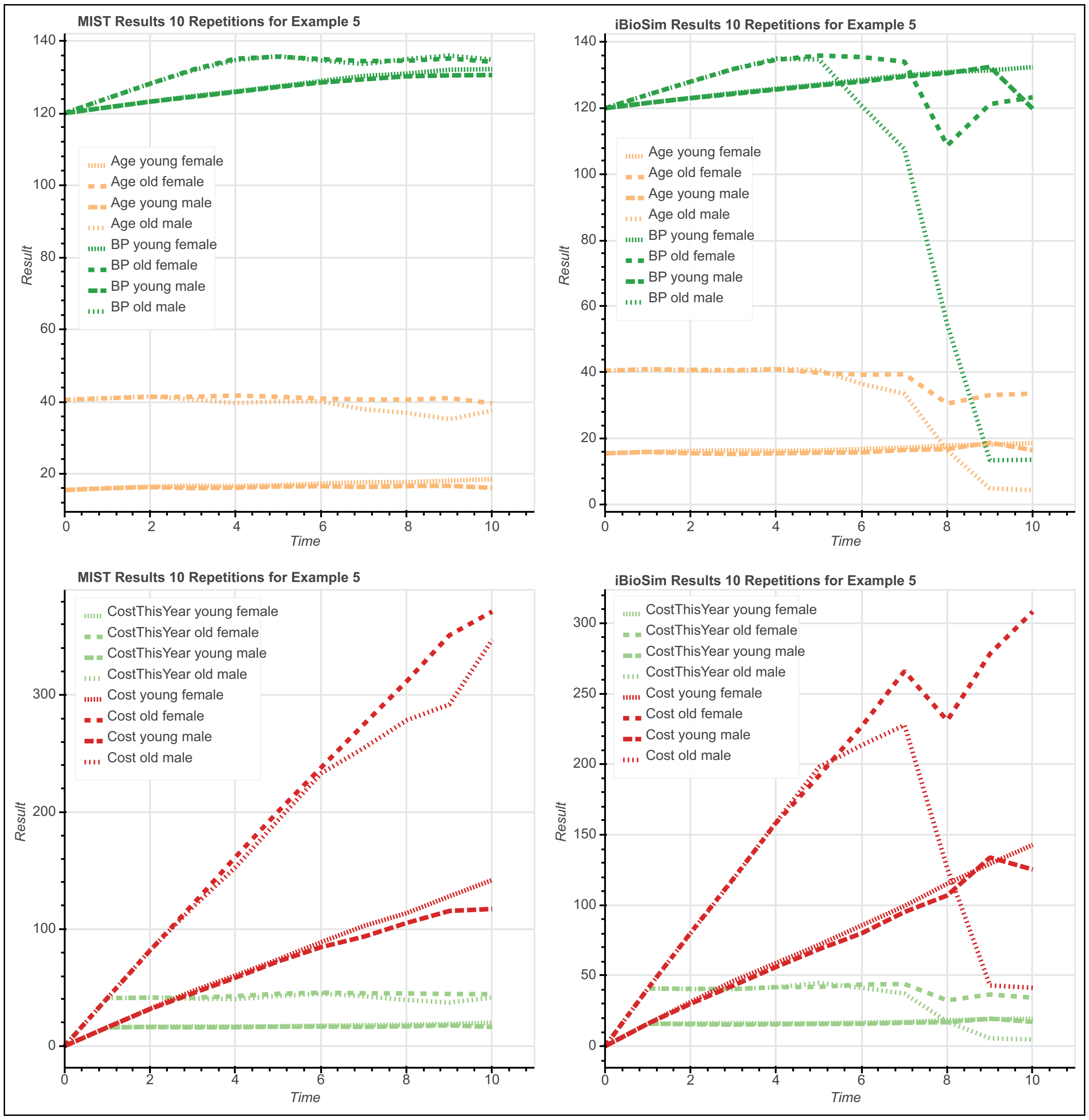


Figure 7. Results comparison between MIST and iBioSim for 10 runs.

which behave in a similar way 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 *Instruction Number* to guide the model during simulation, such that state transitions are considered at each simulation time step only after *InstructionNumber* reaches a 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.

3.5 Example 5: State transition model with treatment and costs

This example (depicted in Figure 5) adds *blood pressure* as another parameter that increases yearly at different rates. Once *blood pressure* is above a threshold, treatment is administered that reduces 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, hence making SBML Arrays essential. Table 5 shows the event scheme implementation in SBML. Notice that in this example there are a number of post-transition rules

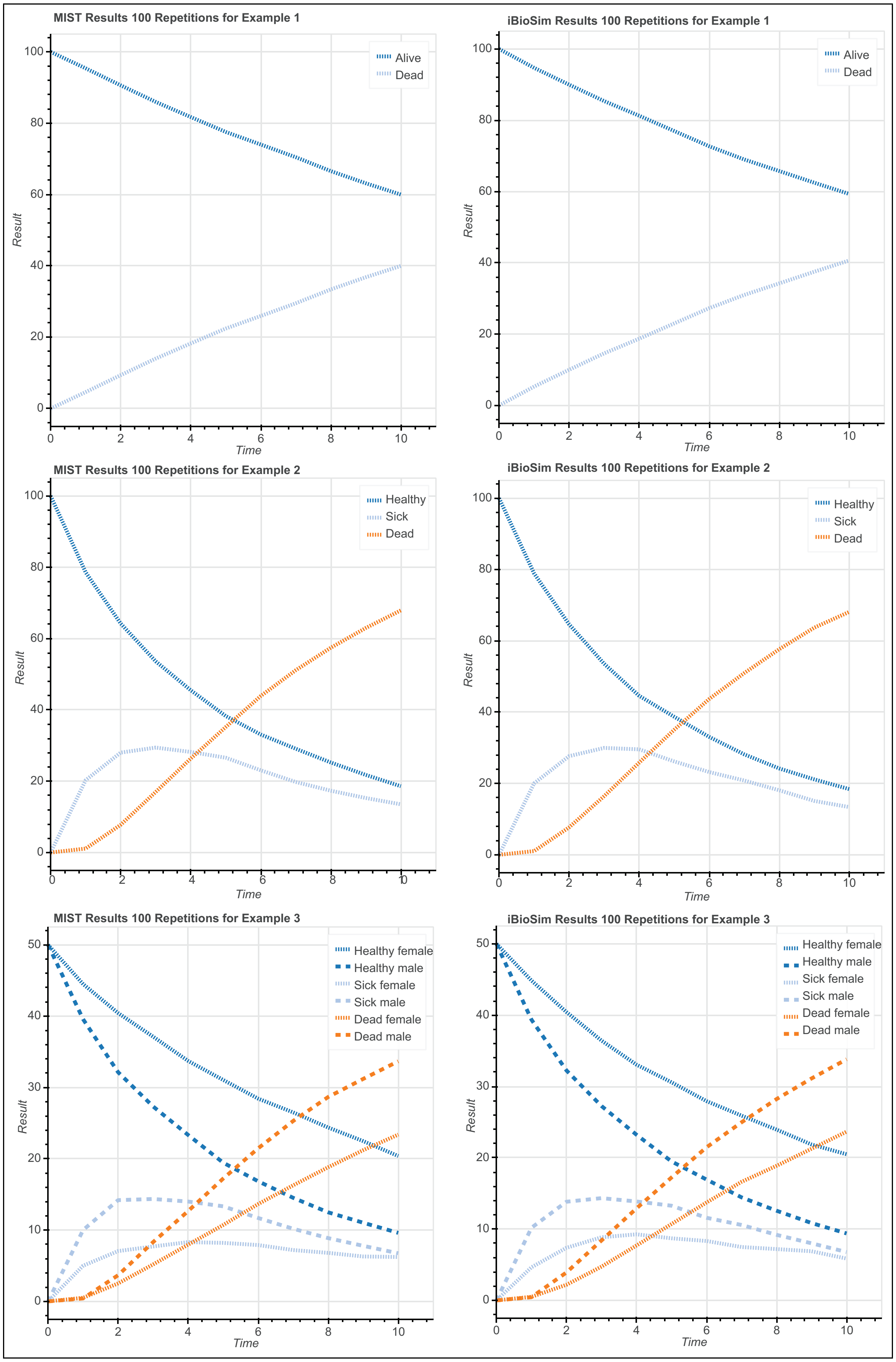


Figure 8. Continued

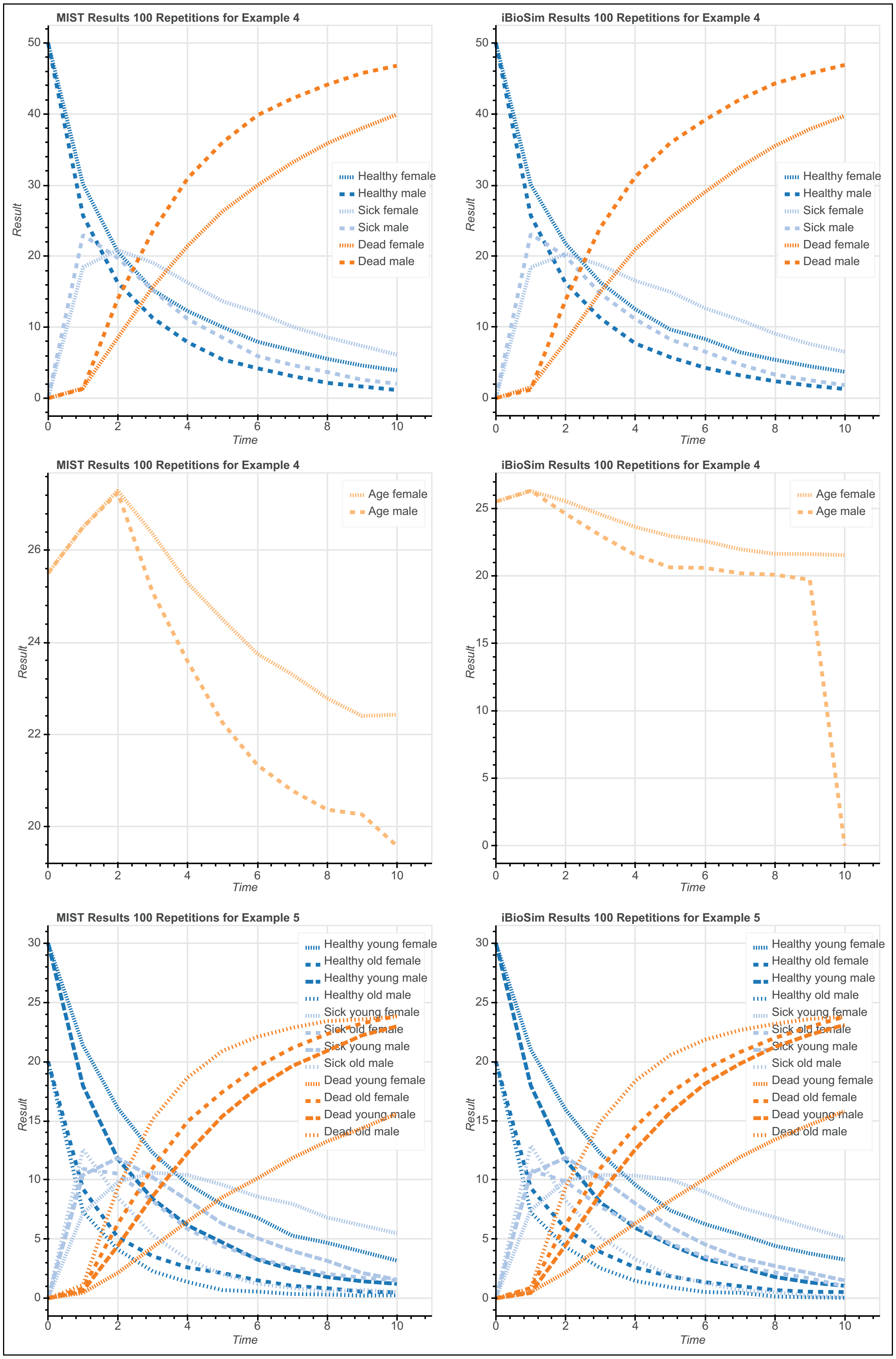


Figure 8. Continued

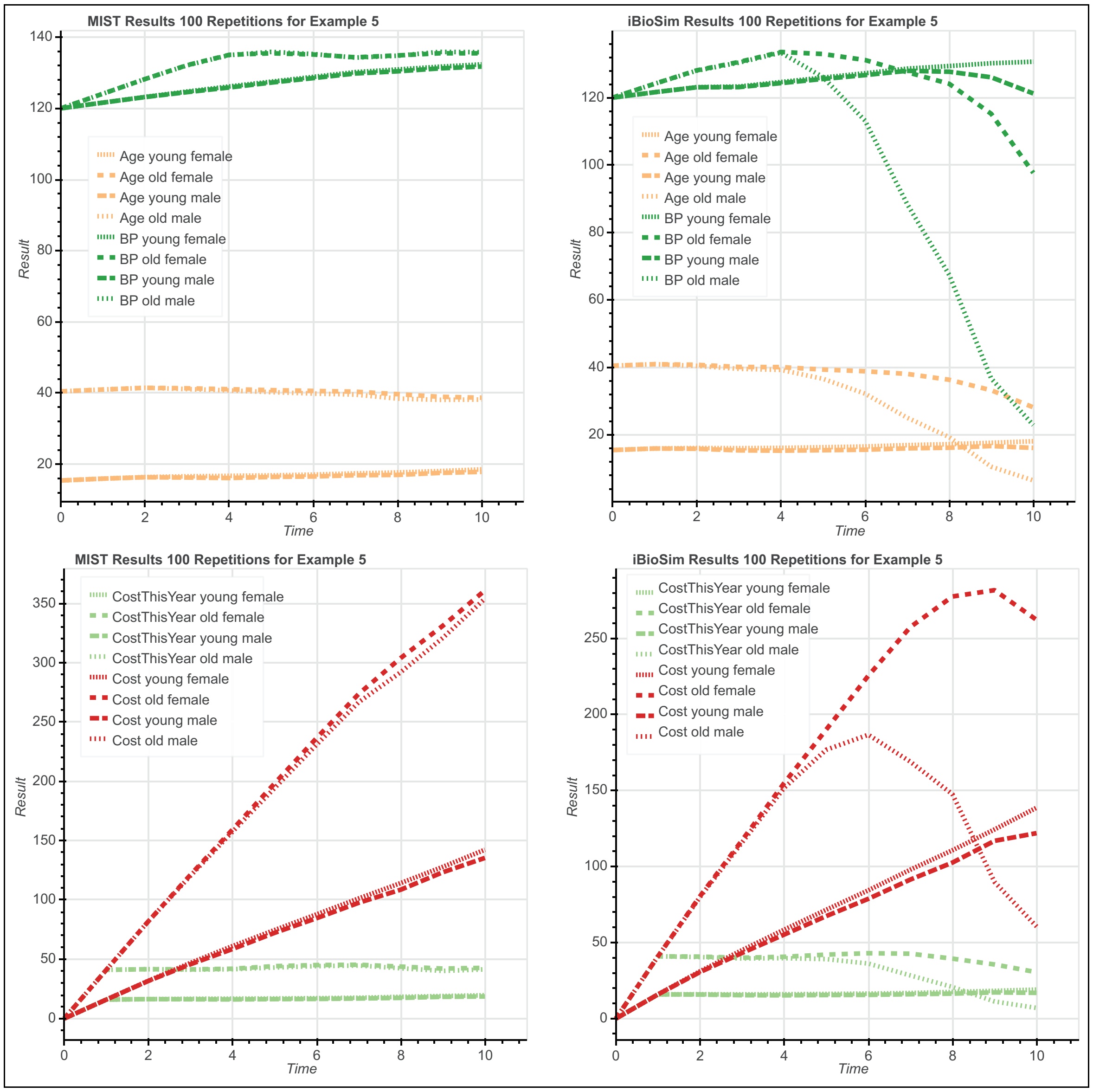


Figure 8. Results comparison between MIST and iBioSim for 100 runs.

implemented as event triplets: events #3–#5 handle *age* increment in pre-transition, events #6–#8 handle *blood pressure* pre-transition update, events #17–#19 determine whether treatment is administered post-transition, events #20–#22 adjust *blood pressure* according to treatment for next timestep post-treatment calculation, events #23–#25 calculate yearly cost that includes treatment cost, and finally events #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 of these using the *InstructionCounter*.

4 Results

The examples described in the previous section 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. Since MIST is particularly designed for disease modeling, comparable results provide sufficient support for the claim that SBML Arrays is suitable for creating reproducible disease models. Figures 6, 7, and 8 present

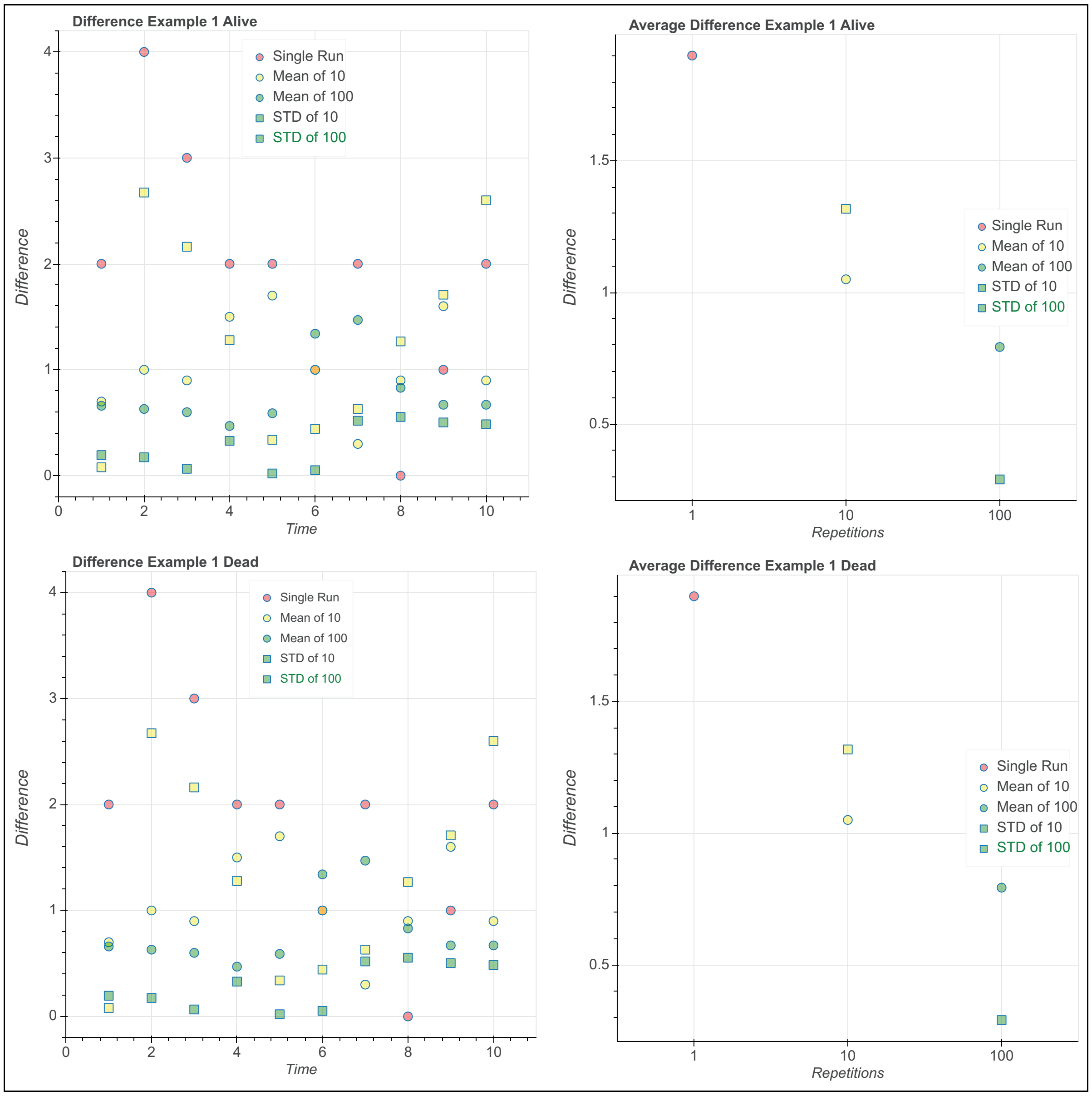


Figure 9. Statistical analysis for Example 1.
STD: standard deviation.

the results of simulation using MIST compared with SBML Arrays implemented in iBioSim. Since this is a random simulation, the results should not match exactly using a single run of the simulation. Figure 6 shows that this is the case, yet the results are comparable enough to indicate a similar simulation. To verify that the models are indeed 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 using 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 for different numbers of repetitions. The same models were executed using 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

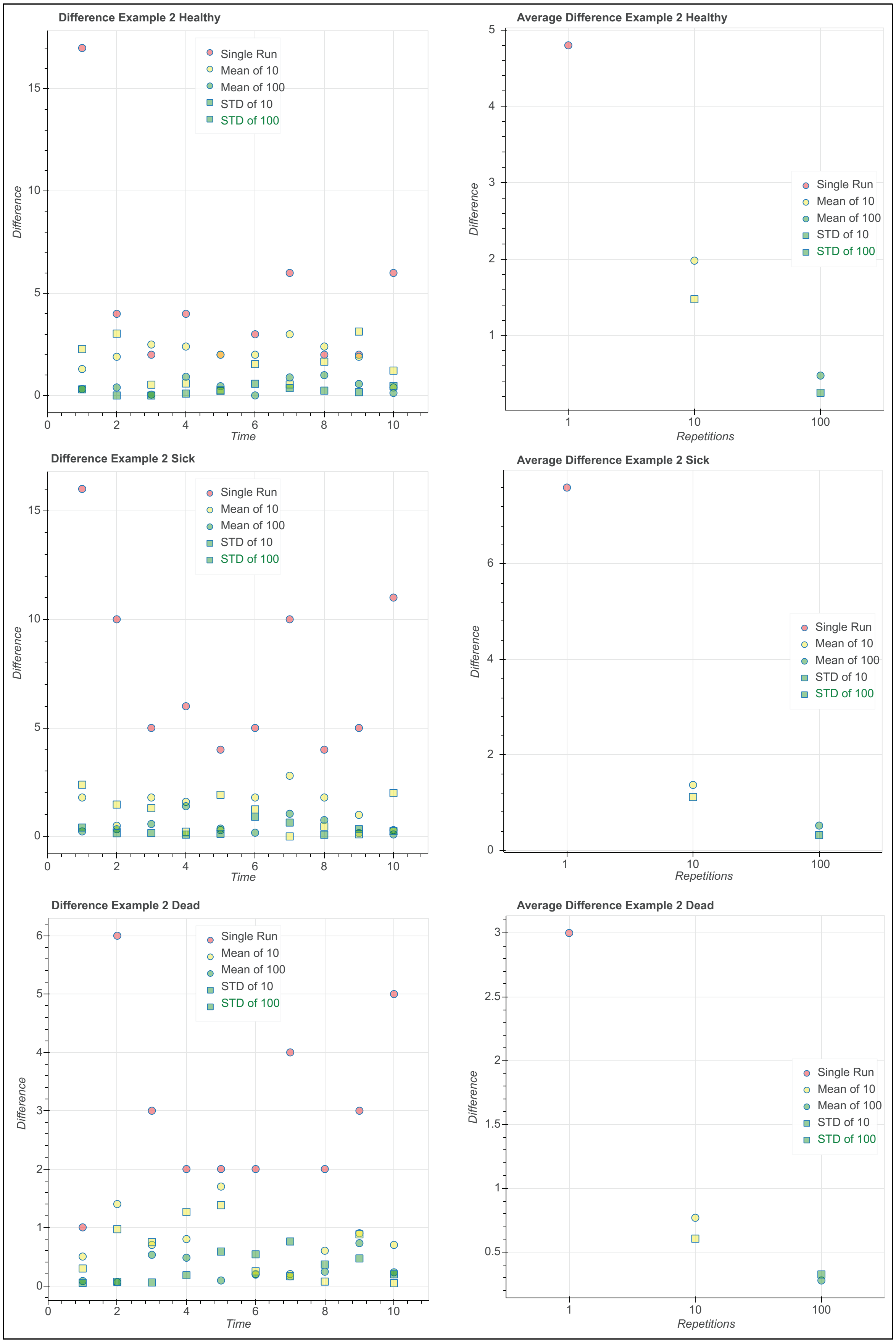


Figure 10. Statistical analysis for Example 2.
STD: standard deviation.

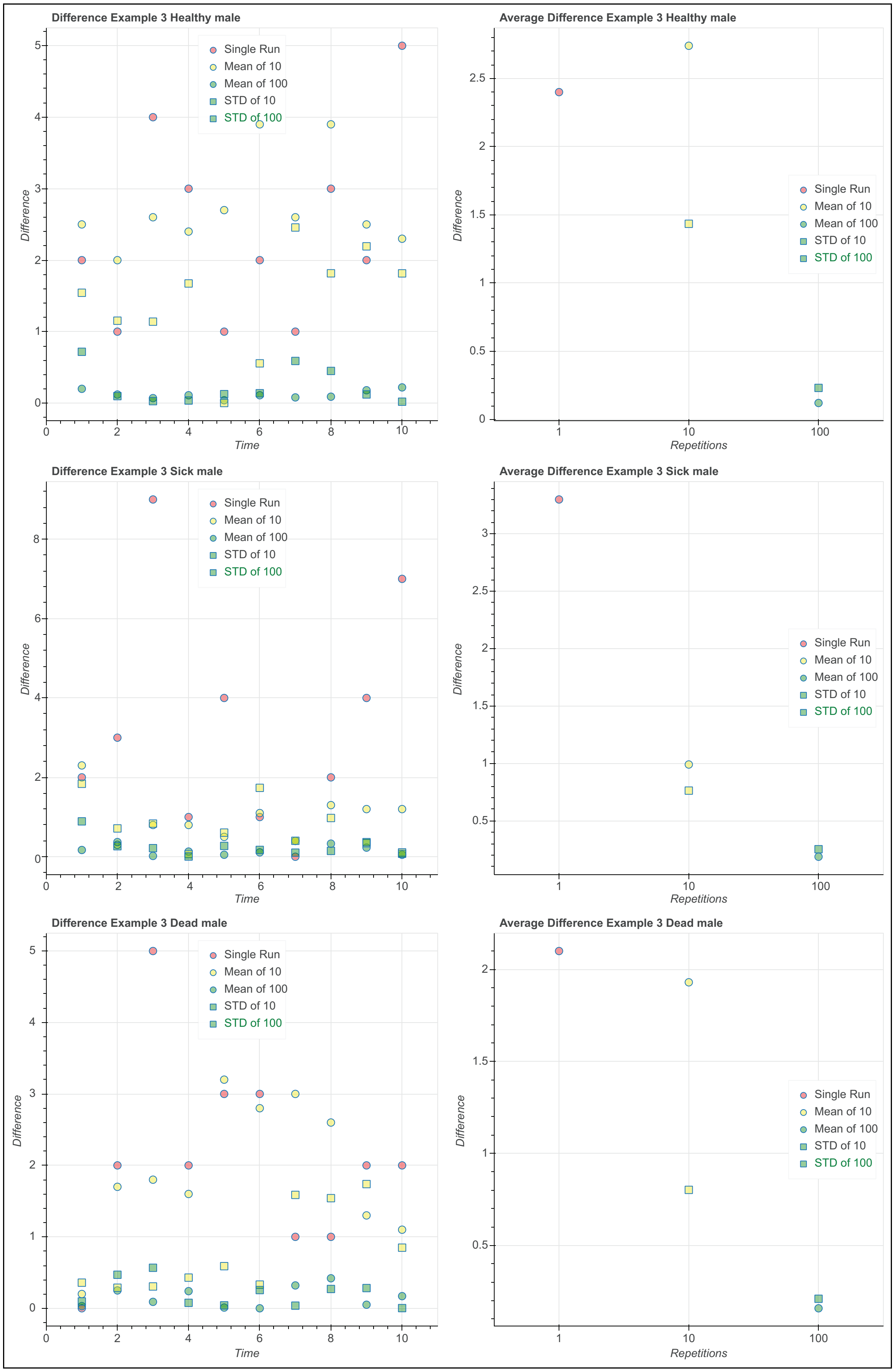


Figure 11. Statistical analysis for men in Example 3.
STD: standard deviation.

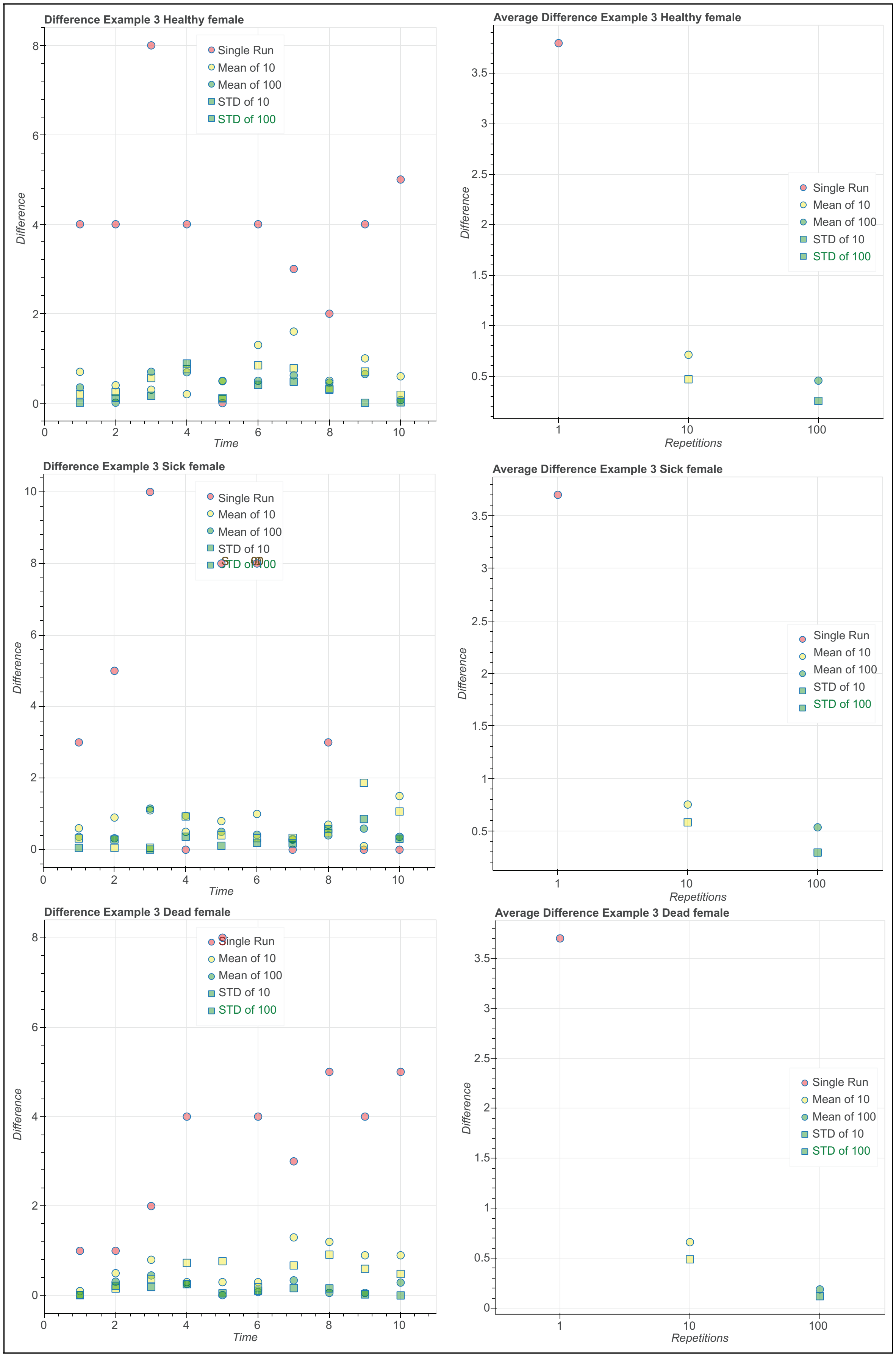


Figure 12. Statistical analysis for women in Example 3.
STD: standard deviation.

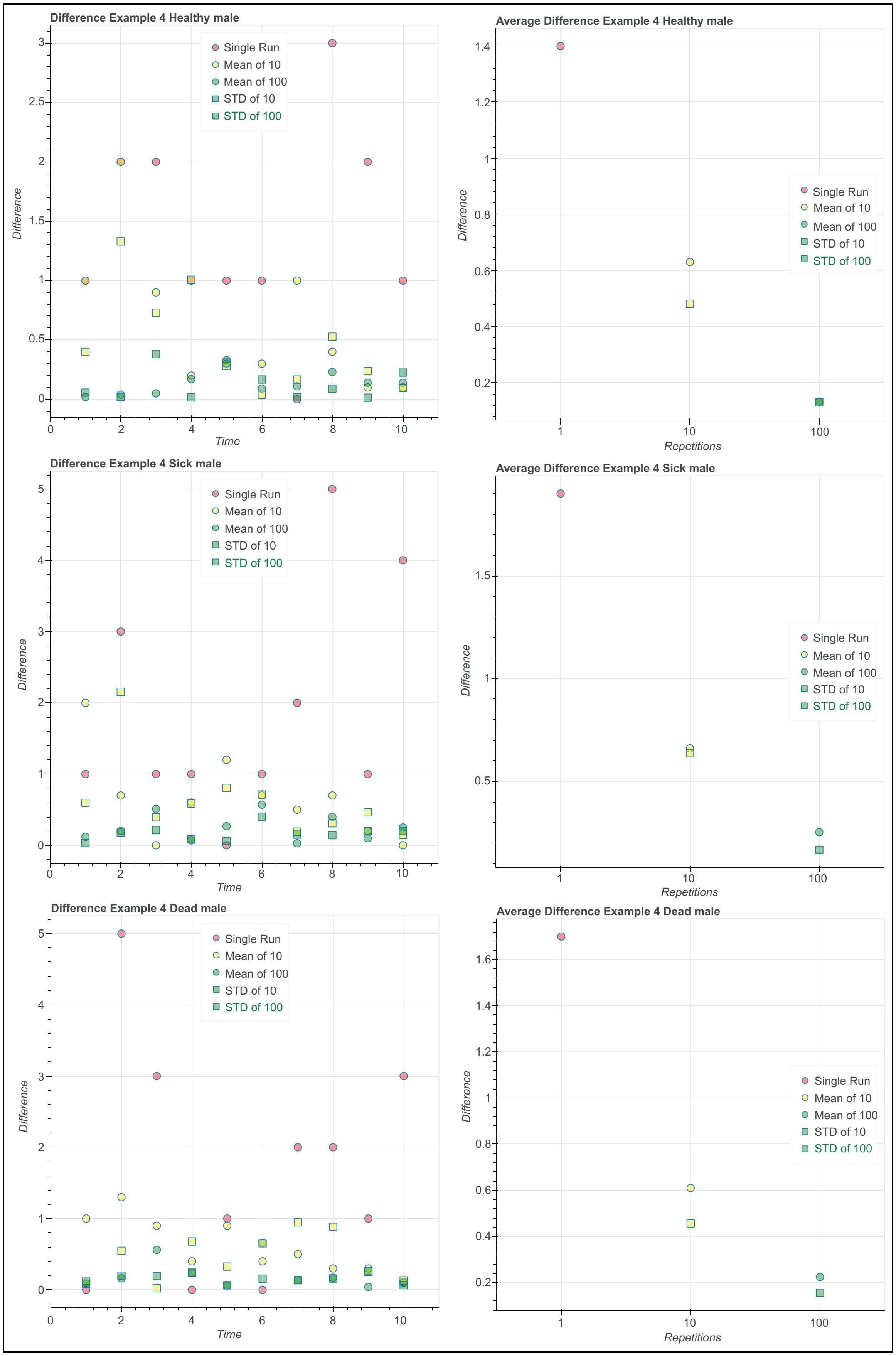


Figure 13. Continued

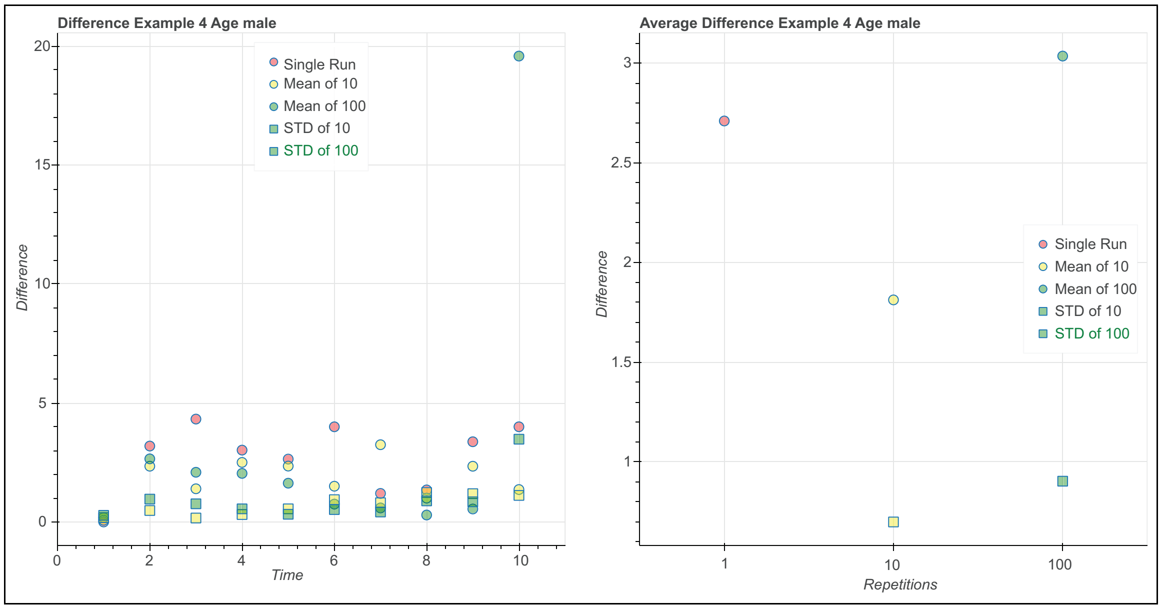


Figure 13. Statistical analysis for men in Example 4.
STD: standard deviation.

(Figures 11 and 12), Example 4 (Figures 13 and 14), and Example 5 (Figures 15, 16, 17, and 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 the 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 these statistics for most plots, where the differences in both the mean and standard deviation are reduced by adding iterations. There are several outliers where the mean does not follow this trend, such as in Example 3 (healthy, male, 10 repetitions, mean), yet even in those cases, the 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 these examples, the mean statistic improves and, considering that Example 5 is highly stratified, has some relatively rare events, as well as 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 Figure 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 for replication of the results in this paper in both MIST and iBioSim, as well as Python code, to assist SBML creation and additional statistical analysis. Although the results were generated using 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.[27] 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.

5 Discussion

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

These 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, which are regularly used to model chronic disease progression at the individual level, are presented here. These examples are sufficient to support such tasks 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 and splitting and joining disease processes and other elements supported by MIST with the intention of promoting SBML Arrays as part of the SBML standard. In

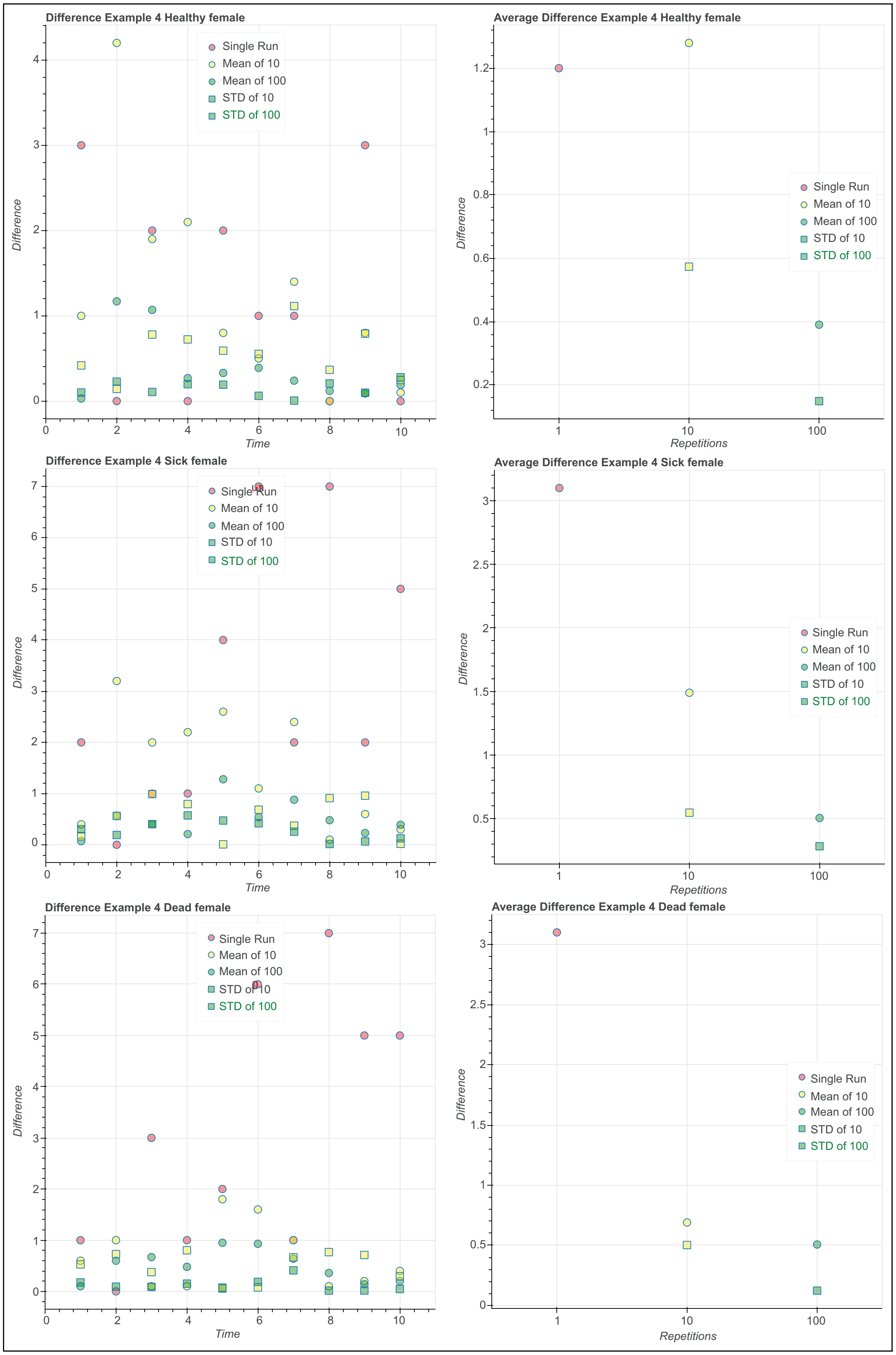


Figure 14. Continued

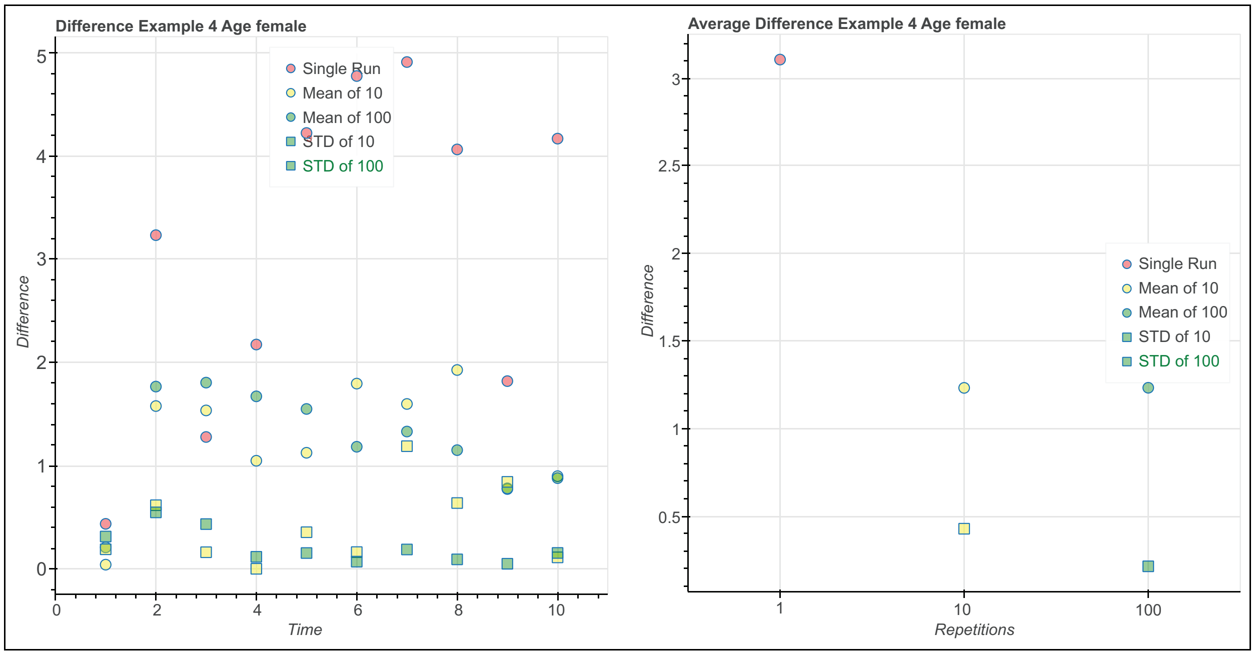


Figure 14. Statistical analysis for women in Example 4.
STD: standard deviation.

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 the feasibility of SBML Arrays as a model transport mechanism. It is only a step toward adoption by SBML editors into the SBML standard, which already provides SBML Arrays specification and tools toward such a goal.

The SBML standard is widely adopted and is very active within the Multi-Scale Modeling community,[28] which tries to address different types of modeling that traverse scales, from cells to organs to populations. When modeling many types of systems at different scales, it is essential to have many modeling capabilities. SBML has 280 reported systems that support it as well as an established development process and specifications, and there are two annual meetings for users. 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 who need to support many modeling systems.

Many modeling systems support microsimulation, as mentioned in a recent review by Sorensen et al.[29] However, we are aware of very few other similar approaches to allow model representation toward transport between systems. PharmML[13] is a close candidate that was already addressed by Smith et al.[15] and, since it works together with the human readable MDL,[14] we consider them together. 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 published in biomodels.net. Another effort worth mentioning toward supporting communicating models is the ODD (overview, design concepts, and details) protocol[30] used to describe agent-based models. However, this protocol, although helpful to convey models to human beings, is not a model transport mechanism that allows transporting models between computing systems. If we move beyond biomedical models, another known modeling standard is the Discrete Event System Specification (DEVS), which 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. The DEVS basic formulation is very simple, as can be seen in Tendeloo and Vangheluwe,[31] and the authors see potential in this formulation, which supports parallelism[32] that invites future exploration, yet despite its popularity, it is not widely used by biomedical communities. However, recent work connecting DEVS and SBML[27] 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

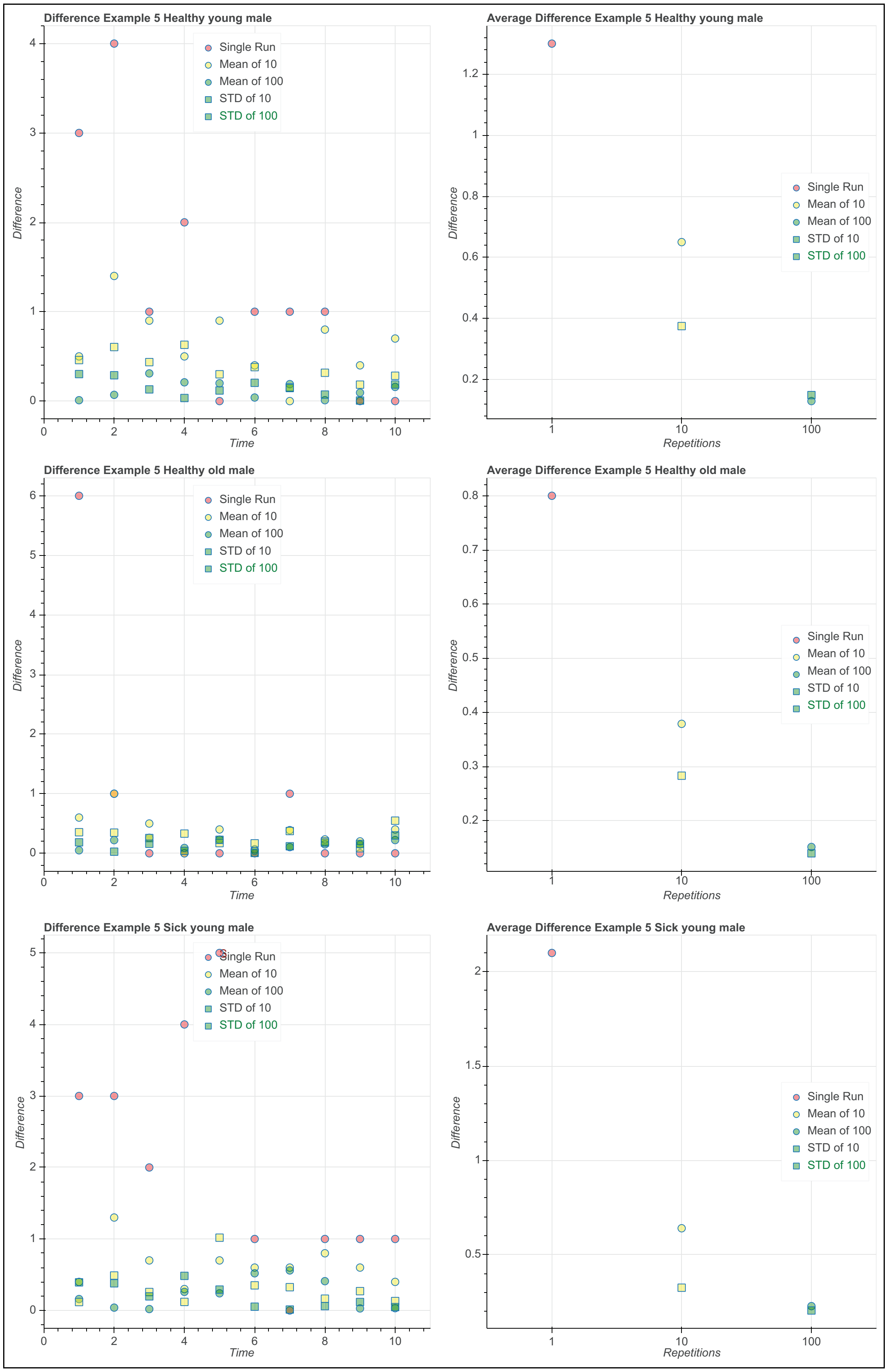


Figure 15. Continued

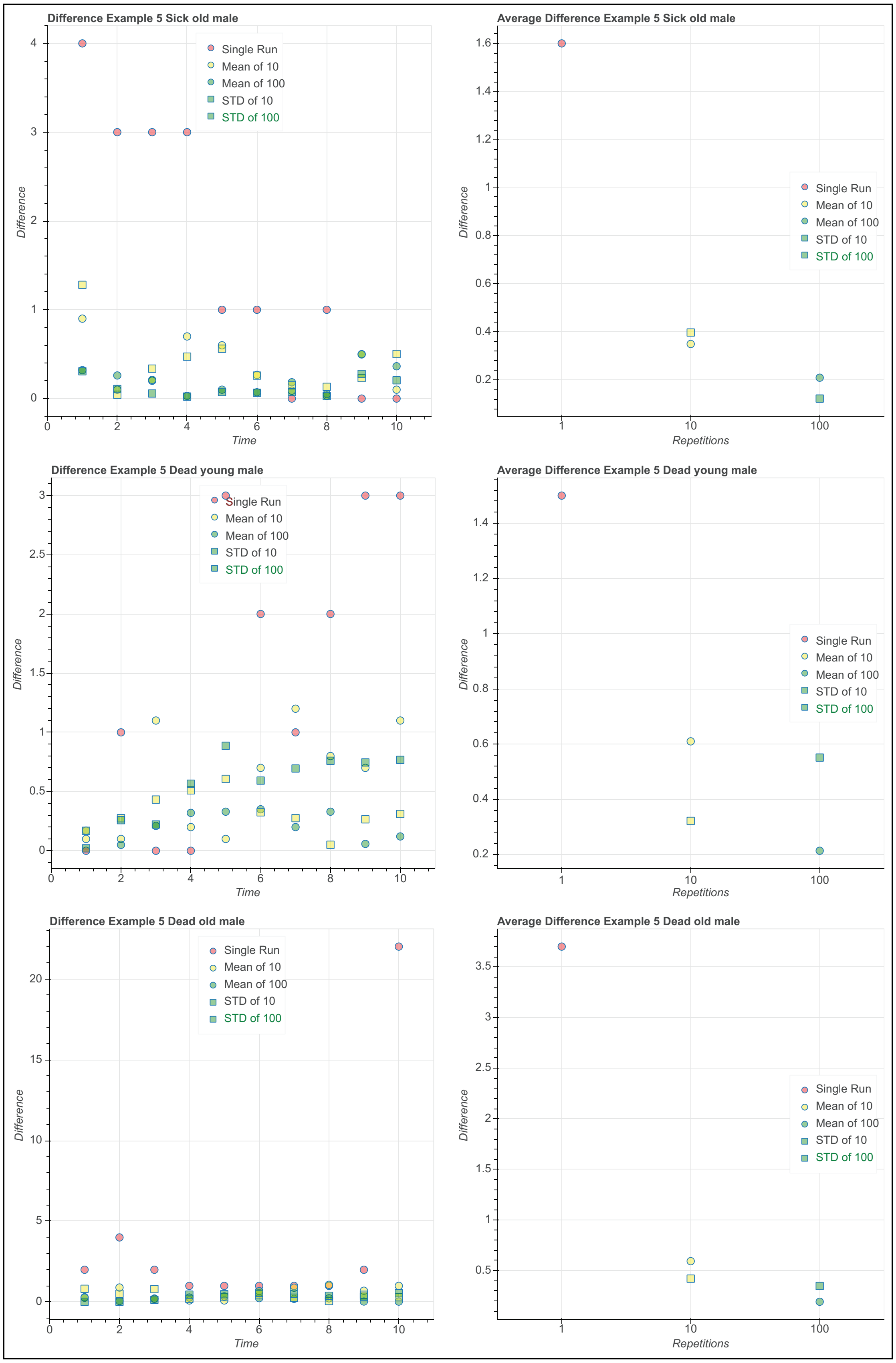


Figure 15. Continued

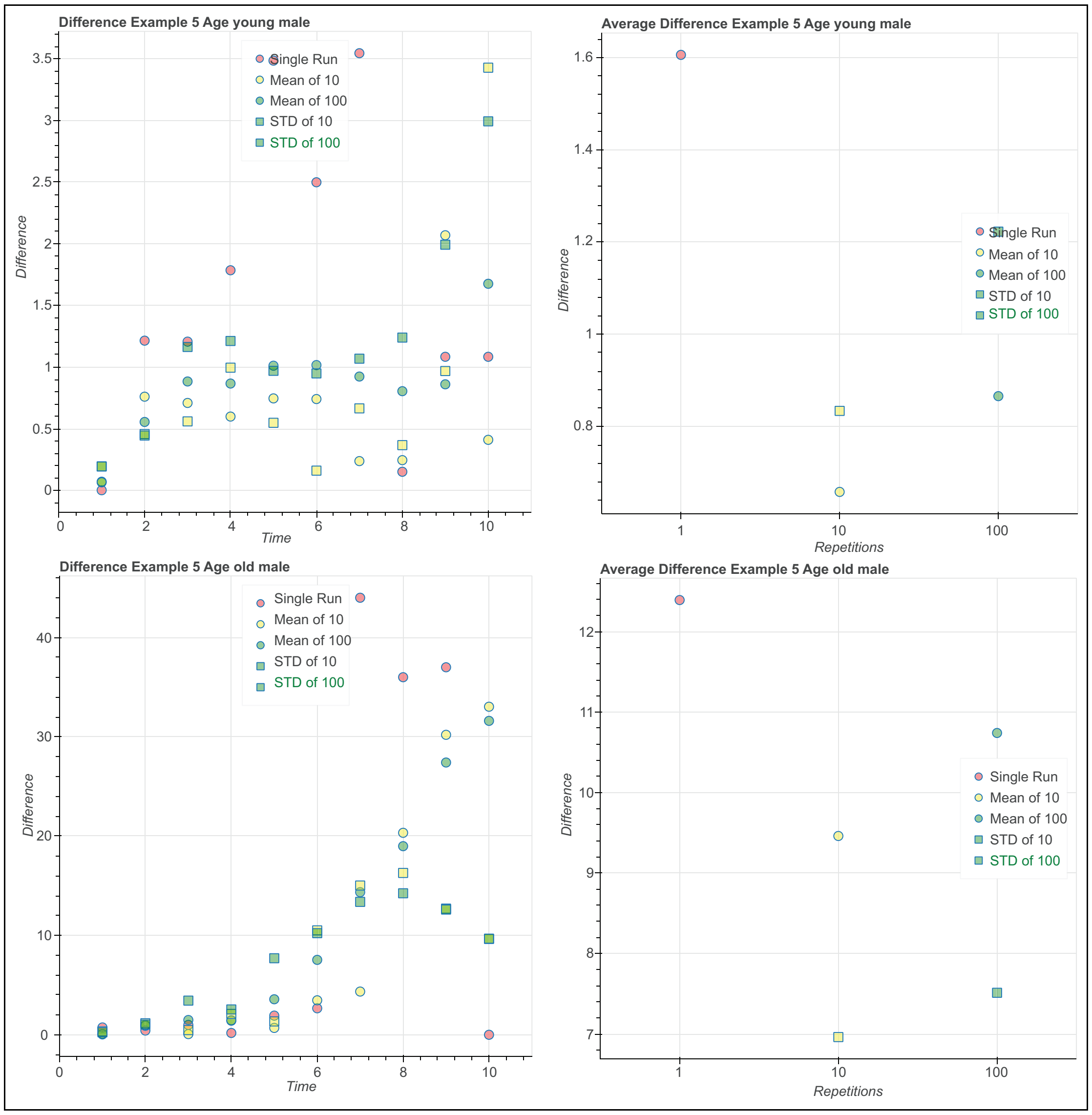


Figure 15. Statistical analysis for the average age of living men and the numbers of healthy, dead, and sick men in Example 5. STD: standard deviation.

generate different results. Even if two different tools are provided the same random numbers, the 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 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 toward the same statistical solution, as was demonstrated. In this paper, we made an effort to include all possible information to allow reproduction of 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 assist infectious disease modelers who often use variations of SIR (susceptible, infectious, recovered) models and their extensions.[3] An example of a 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; this may

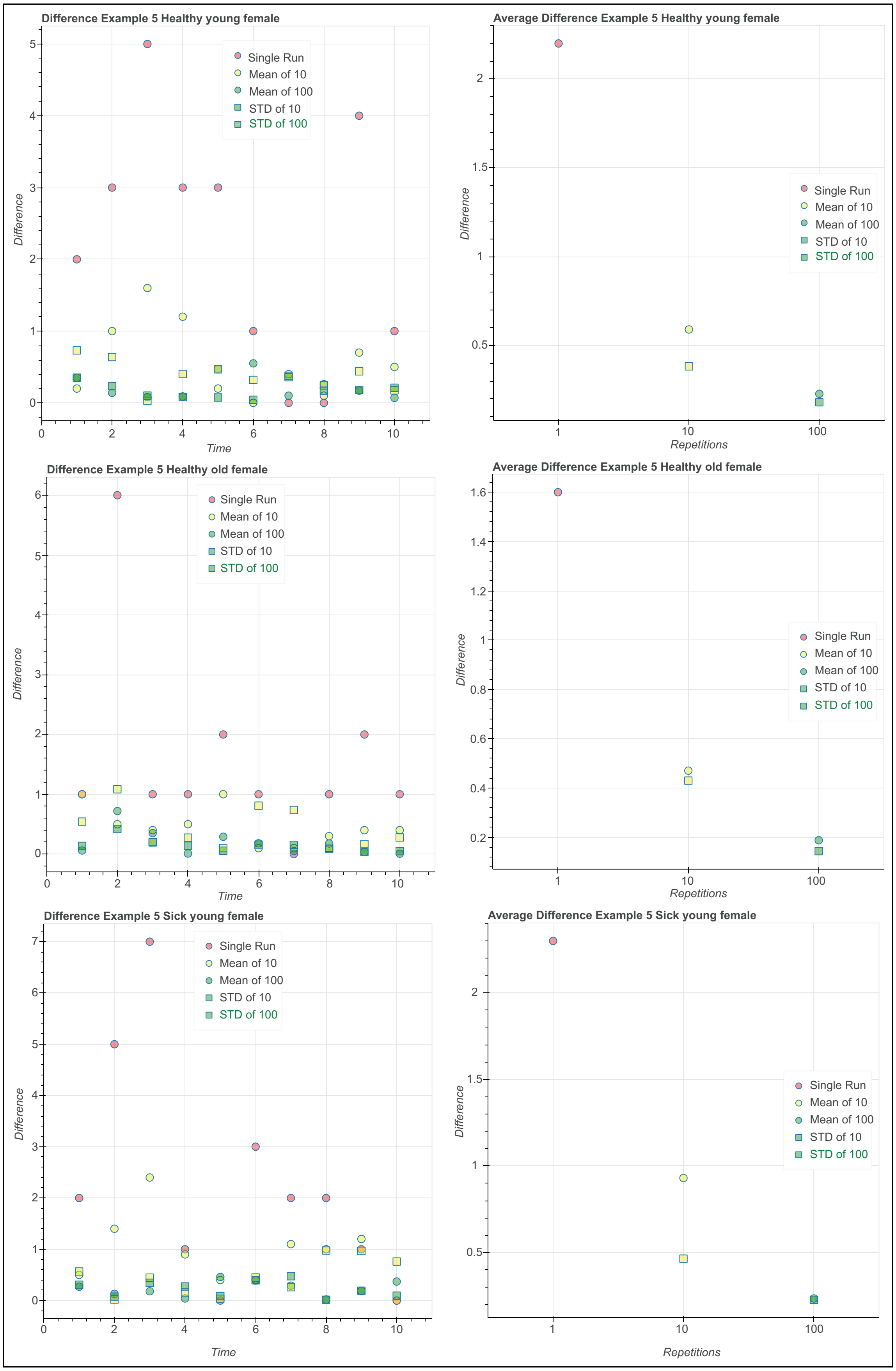


Figure 16. Continued

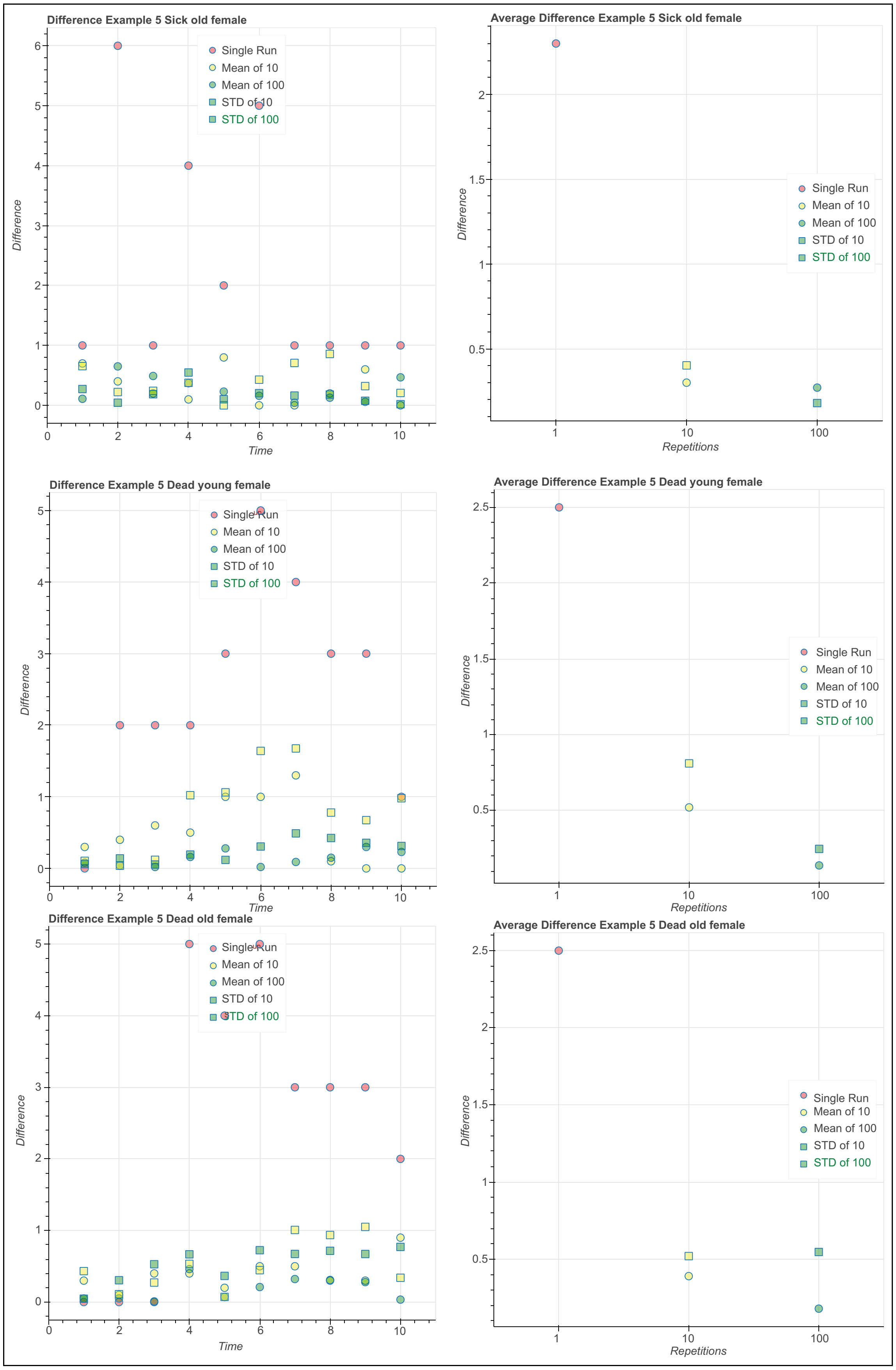


Figure 16. Continued

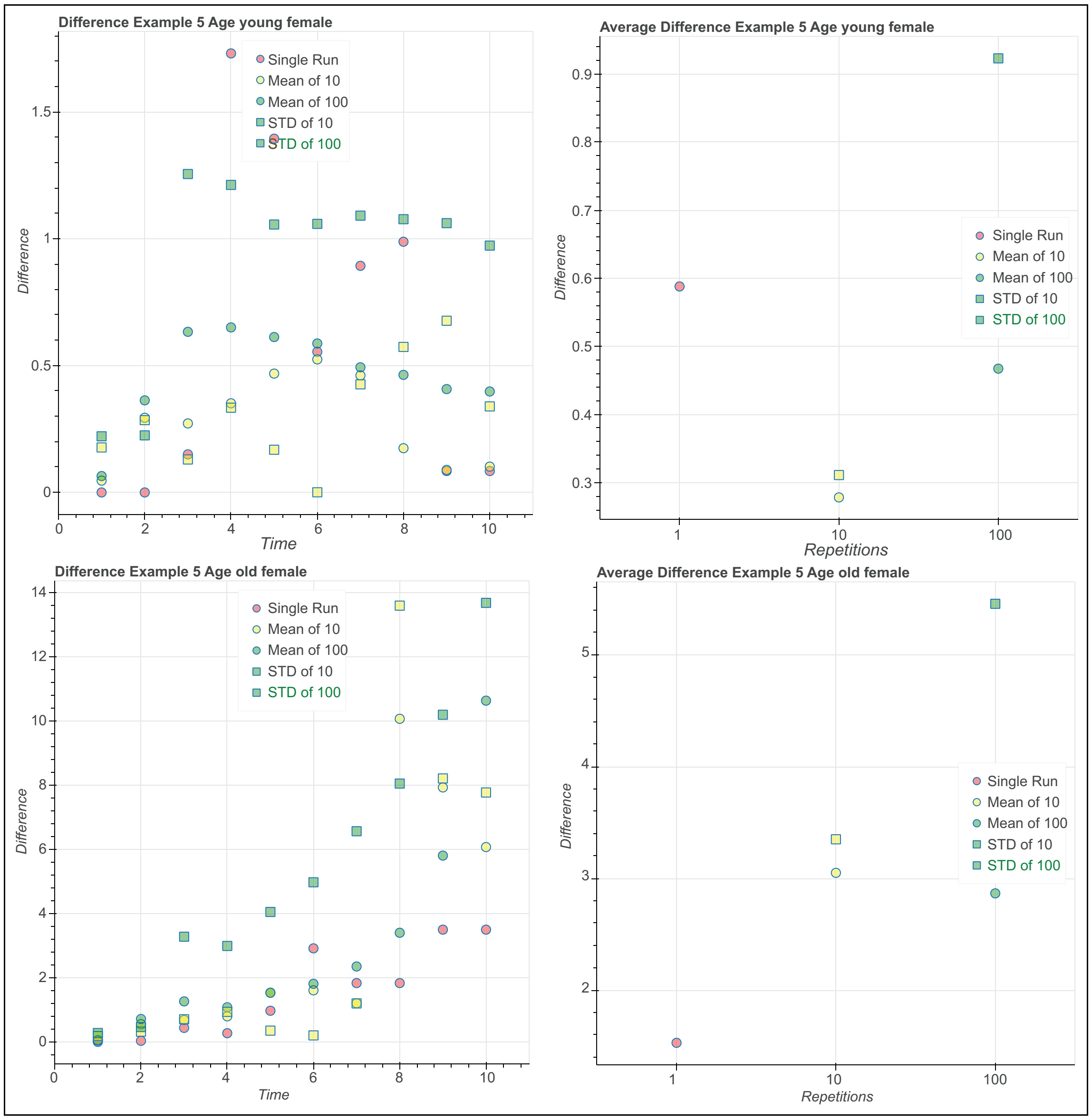


Figure 16. Statistical analysis for the average age of living women and the numbers of healthy, dead, and sick women in Example 5. STD: standard deviation.

open opportunities in the future to model interactions at the individual level.

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

6 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 using 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

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not

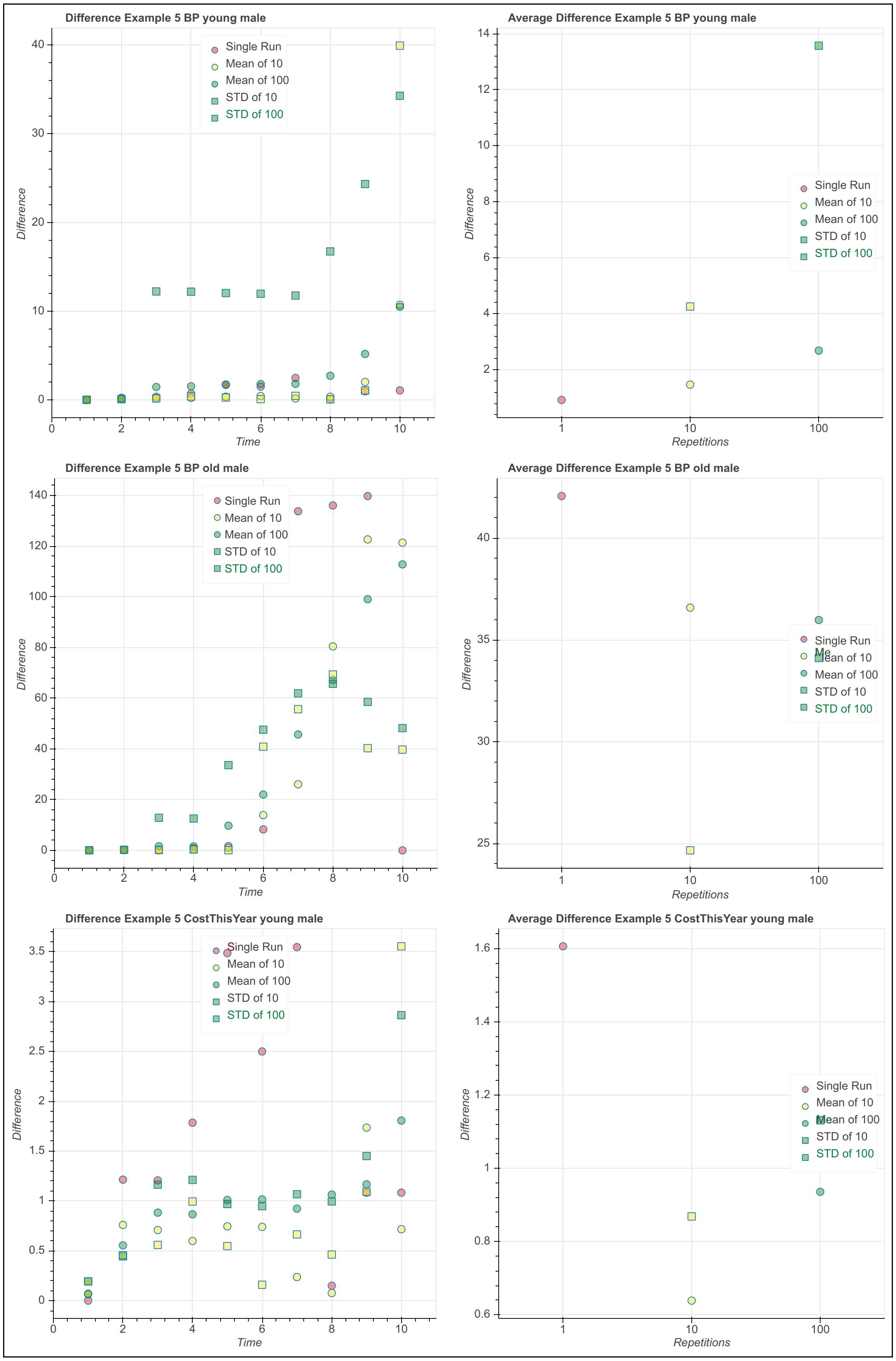


Figure 17. Continued

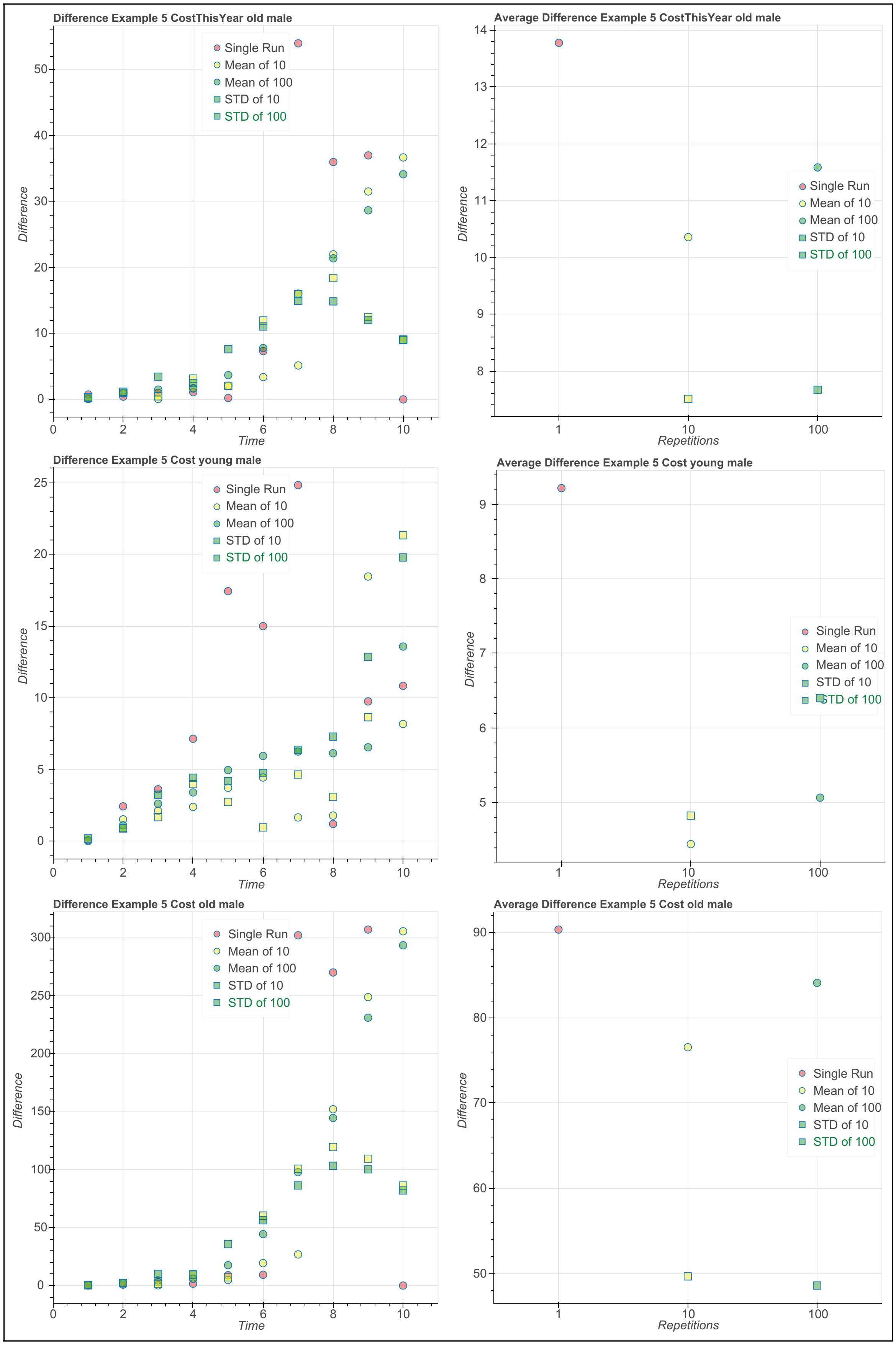


Figure 17. Statistical analysis for blood pressure, cost, and cost this year for men in Example 5.

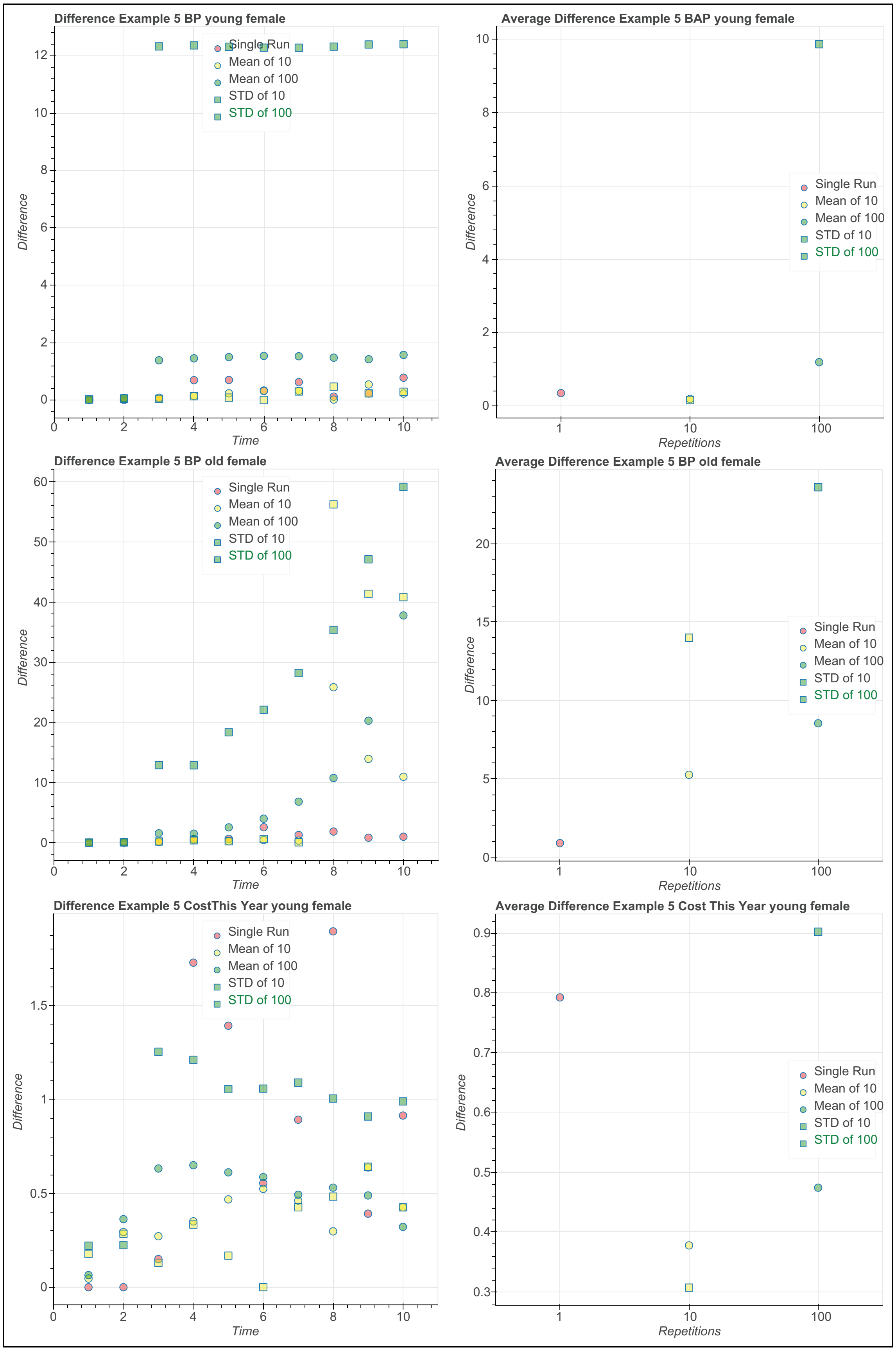


Figure 18. Continued

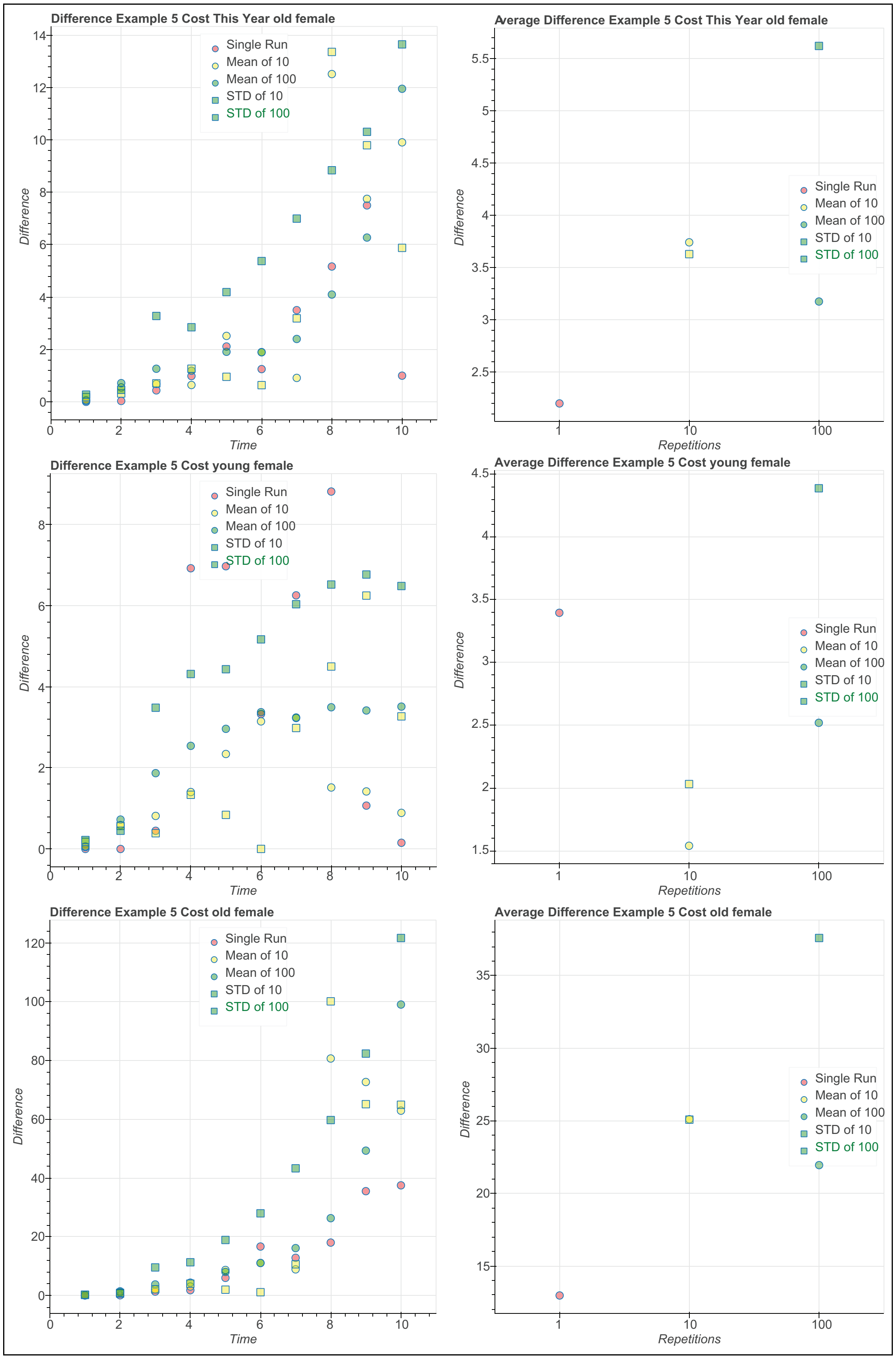


Figure 18. Statistical analysis for blood pressure, cost, and cost this year for women in Example 5.
STD: standard deviation.

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, who helped with this work by participating in the discussion or providing solutions: Lucian Smith, Brett Olivier, Nicolas Le Novere, Fengkai Zhang, and Sarah Keating.

Funding

This work was supported by the National Science Foundation (grant numbers CCF-1218095 and CCF-1748200).

ORCID ID

Leandro Watanabe https://orcid.org/0000-0001-7030-8690

References

1. Hayes AJ, Leal J, Gray AM, et al. 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 30 year United Kingdom Prospective Diabetes Study, UKPDS 82. *Diabetologia* 2013; 56(9): 1925–1933.
2. D'Agostino RB, Vasan RS, Pencina MJ, et al. General cardiovascular risk profile for use in primary care. *Circulation* 2008; 117(6): 743–753.
3. Salem D and 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*, Montreal, Quebec, Canada, 24–27 July 2016, p. 3:1–3:8. San Diego, CA, USA: Society for Computer Simulation International.
4. Lasry A, Zaric GS and Carter MW. Multi-level resource allocation for HIV prevention: A model for developing countries. *Eur J Oper Res* 2007; 180(2): 786–799.
5. Leff HS, Dada M and Graves SC. An LP planning model for a mental health community support system. *Manage Sci* 1986; 32(2): 139–155.
6. Scholz S, Batram M and Greiner W. The SILAS model: Sexual infections as large-scale agent-based simulation. In: *Proceedings of the conference on summer computer simulation. SummerSim '15*, Chicago, IL, USA, 26–29 July 2015. San Diego, CA, USA: Society for Computer Simulation International.
7. Hippisley-Cox J, Coupland C, Vinogradova Y, et al. Derivation and validation of QRISK: A new cardiovascular disease risk score for the United Kingdom; prospective open cohort study. *BMJ* 2007; 335(7611): 136.
8. Hippisley-Cox J, Coupland C, Vinogradova Y, et al. Predicting cardiovascular risk in England and Wales: Prospective derivation and validation of QRISK2. *BMJ* 2008; 336(7659): 1475–1482.
9. Isaman DJ, Barhak J and Ye W. Indirect estimation of a discrete-state discrete-time model using secondary data analysis of regression data. *Stat Med* 2009; 28(16): 2095–2115.
10. Barhak J, Isaman DJ, Ye W, et al. Chronic disease modeling and simulation software. *J Biomed Inf* 2010; 43(5): 791–799.
11. Palmer AJ, Si L, Tew M, et al. *Economics Modelling and Diabetes: The Mount Hood 2016 Challenge*. https://docs.wixstatic.com/ugd/4e58240964b3878cab490da965052ac6965145.pdfm (accessed 11 August 2018).
12. Hucka M, Finney A, Sauro HM, et al. The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics* 2003; 19(4): 524–531.
13. Swat M, Moodie S, Wimalaratne S, et al. Pharmacometrics Markup Language (PharmML): Opening new perspectives for model exchange in drug development. *CPT Pharmacometrics Syst Pharmacol* 2015; 4(6): 316–319.
14. Smith MK, Moodie SL, Bizzotto R, et al. Model Description Language (MDL): A standard for modeling and simulation. *CPT Pharmacometrics Syst Pharmacol* 2017; 6(10): 647–650.
15. Smith L, Swat MJ and Barhak J. Sharing formats for disease models. In: *Proceedings of the summer computer simulation conference. SCSC '16*, Montreal, Quebec, Canada, 24–27 July 2016. San Diego, CA, USA: Society for Computer Simulation International.
16. Barhak J. MIST: Micro-simulation tool to support disease modeling. SciPy, bioinformatics track, https://github.com/scipy-conference/scipy2013_talks/tree/master/talks/jacob_barhak (accessed 11 August 2018).
17. Hucka M, Nickerson DP, Bader GD, et al. Promoting coordinated development of community-based information standards for modeling in biology: The COMBINE initiative. *Front Bioeng Biotechnol* 2015; 3: 19.
18. Davidich MI and Bornholdt S. Boolean network model predicts cell cycle sequence of fission yeast. *PloS One* 2008; 3(2): e1672.
19. Goss PJ and Peccoud J. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc Natl Acad Sci USA* 1998; 95(12): 6750–6755.
20. Madsen C, Zhang Z, Roehner N, et al. Stochastic model checking of genetic circuits. *J Emerg Technol Comput Syst* 2014; 11(3): 23:1–23:21.
21. Smith LP, Hucka M, Hoops S, et al. SBML level 3 package: Hierarchical model composition, version 1 release 3. *J Integr Bioinform* 2015; 12(2): 603–659.
22. Olivier BG and Bergmann FT. The Systems Biology Markup Language (SBML) level 3 package: Flux balance constraints. *J Integr Bioinform* 2015; 12(2): 660–690.
23. Gauges R, Rost U, Sahle S, et al. The Systems Biology Markup Language (SBML) level 3 package: Layout, version 1 core. *J Integr Bioinform* 2015; 12(2): 550–602.
24. Watanabe L and Myers CJ. Efficient analysis of systems biology markup language models of cellular populations using arrays. *ACS Synth Biol* 2016; 5(8): 835–841.
25. Bornstein BJ, Keating SM, Jouraku A, et al. LibSBML: An API Library for SBML. *Bioinformatics* 2008; 24(6): 880–881.
26. Rodriguez N, Thomas A, Watanabe L, et al. JSBML 1.0: Providing a smorgasbord of options to encode systems biology models. *Bioinformatics* 2015; 31(20): 3383–3386.
27. Belloli L, Wainer G and Najmanovich R. Parsing and model generation for biological processes. In: *Proceedings of the symposium on theory of modeling & simulation. TMS-DEVS*

'16, Pasadena, CA, USA, 3–6 April 2016, p. 21:1–21:6. Piscataway, NJ, USA: IEEE.

28. Interagency Modeling and Analysis Group. Frequently Asked Questions (FAQ), https://www.imagwiki.nibib.nih.gov/content/frequently-asked-questions-faq (accessed 11 August 2018).
29. Sorensen RJD, Flaxman AD, Deason A, et al. Microsimulation models for cost-effectiveness analysis: A review and introduction to CEAM. In: *Proceedings of the summer simulation multi-conference. SummerSim '17*, Bellevue, WA, USA, 9–12 July 2017, p. 32:1–32:11. San Diego, CA, USA: Society for Computer Simulation International.
30. Grimm V, Berger U, Bastiansen F, et al. A standard protocol for describing individual-based and agent-based models. *Ecol Modell* 2006; 198(1): 115–126.
31. Tendeloo YV and Vangheluwe H. Introduction to parallel DEVS modelling and simulation, http://msdl.cs.mcgill.ca/people/yentl/papers/2017-DEVSTutorial.pdf. (2017, accessed 12 February 2018).
32. Zeigler BP. Using the parallel DEVS protocol for general robust simulation with near optimal performance. *Comput Sci Eng* 2017; 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 a 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 developed 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 Department of Electrical and Computer Engineering, University of Utah, USA, and has adjunct appointments with the School of Computing and the Department of Biomedical Engineering, University of Utah, USA.