International Microsimulation Association

RESEARCH ARTICLE

Sima – an Open-source Simulation Framework for Realistic Large-scale Individual-level Data Generation

Santtu Tikka[1]*, Jussi Hakanen[2], Mirka Saarela[2], Juha Karvanen[1]

[1]Department of Mathematics and Statistics, University of Jyvaskyla, Finland; [2]Faculty of Information Technology, University of Jyvaskyla, Jyvaskyla, Finland

Abstract We propose a framework for realistic data generation and the simulation of complex systems and demonstrate its capabilities in a health domain example. The main use cases of the framework are predicting the development of variables of interest, evaluating the impact of interventions and policy decisions, and supporting statistical method development. We present the fundamentals of the framework by using rigorous mathematical definitions. The framework supports calibration to a real population as well as various manipulations and data collection processes. The freely available open-source implementation in R embraces efficient data structures, parallel computing, and fast random number generation, hence ensuring reproducibility and scalability. With the framework, it is possible to run daily-level simulations for populations of millions of individuals for decades of simulated time. An example using the occurrence of stroke, type 2 diabetes, and mortality illustrates the usage of the framework in the Finnish context. In the example, we demonstrate the data collection functionality by studying the impact of nonparticipation on the estimated risk models and interventions related to controlling excessive salt consumption.

DOI: https://doi.org/10.34196/ijm.00240

1. Introduction

Simulation is an important tool for decision making and scenario analysis, and can provide great benefits in numerous fields of study. Synthetic data are important also for statistical method development because the underlying true parameters are known and the obtained estimates can be easily compared with these parameters. A simulation may start with real individual-level data and simulate future development with different assumptions. However, real data are often associated with legal and privacy concerns that complicate and sometimes even prevent their use in method development. Therefore, modeling the current state while using only synthetic data in simulation is often a viable option for a researcher who would like to evaluate the performance of a new method. These considerations lead to a conclusion that a general-purpose simulation framework should be modular and support different use cases.

In this paper, we propose an open-source simulation framework that we call *Sima* for the statistical programming language R (***R Core Team, 2020***). In our approach, we adopt for simulation at the individual level. Each individual of the population is subject to a number of events that can, for example, describe changes in behavior, occurrence of disease or illness, or mortality. The simulation uses a fixed-increment time progression, and the occurrence of events is controlled with stochastic processes. Thus, our simulation approach is an example of discrete event simulation. An efficient implementation enables high time resolution making it sometimes possible to approximate continuous-time processes. The source code of the implementation is available on GitHub at https://

***For correspondence:** santtu.tikka@jyu.fi

Author Keywords: calibration, data collection, discrete event simulation, interventions, missing data, synthetic data

github.com/santikka/Sima/, and extensive documentation can be found on the package website at https://santikka.github.io/Sima/.

Below, we outline the fundamental features and properties that an ideal simulator and the simulated population should support in our framework from the viewpoint of statistical method development and decision making. These features are related to the practical aspects of the simulated data, the technical aspects of the implementation, and the types of phenomena we are most interested in. We also provide the fundamentals of the framework using rigorous mathematical definitions.

Support for Alignment: In a typical use case, the goal is to match the simulated population to a real population of interest in the aspects viewed as the most important relative to the research questions at hand. The process used to accomplish this is typically referred to as alignment. To make alignment possible, the simulated system can be calibrated by using real data according to the chosen measures of similarity. The synthetic population under consideration should be sufficiently large to be considered representative. These principles are considered throughout the simulation period including the initial state of the population.

Support for Manipulations: The simulator should support external interventions. We approach this in three ways. An intervention can be a change in the system itself by either adding or removing individuals or events. Alternatively, an intervention can represent a policy decision that indirectly affects the individuals over a period of time. Finally, the intervention can directly target a specific individual, a subset of individuals or the entire population. An intervention of this third type is the idealized manipulation as defined by the do-operator in the causal inference literature (***Pearl, 2009***). For example, one may study the causal effects of a risk factor by forcing all members of the population to have a specific risk factor. Furthermore, interventions enable the study of counterfactual queries.

Support for Data Collection: The simulator should contain a specialized component for user defined data collection processes. This allows for the simulation of real-world sampling procedures or selective participation where only a subset of the entire population is available for practical analyses, for example via stratified sampling where different age groups have different sampling probabilities. Because the true simulated population is always available, the data collection feature can be used to assess the properties of data-analytical methods under a variety of conditions such as missing data that are prevalent in real-world data sets. Additionally, this feature can be used to assess the optimality of various study designs.

Support for Scalability: The simulated system can be scaled up in all aspects without incurring an unreasonable computational burden. Large populations should be supported both in the number of individuals and status variables through long simulation runs. Similarly, there should be no strict limitations in the number of possible events. Ideally, the system can be efficiently parallelized while minimizing the resulting overhead.

Support for Reproducibility: Any run of the simulator and sampling processes associated with it should be replicable. This allows for the replication of any statistical analysis carried out on data obtained from the simulator. The same requirement applies to simulation runs that take advantage of parallelized processing.

R is especially appropriate for an open-source microsimulation framework that incorporates statistical analyses within decision models (***Krijkamp et al., 2018***). There exist powerful open-source microsimulation frameworks developed in Python (***De Menten et al., 2014***), Java (***Richiardi and Richardson, 2017***; ***Mannion et al., 2012***; ***Kosar and Tomintz, 2014***; ***Tomintz et al., 2017***), and C++ (***Clements et al., 2021***). However, we did not find an open-source simulation framework accessible from R that could encapsulate all our desired features simultaneously. For example, the microsimulation package in R by ***Clements et al. (2021)***, even though being open-source, does not include direct support for manipulations, so a new simulator has to be constructed for each interventional scenario. A more comprehensive overview of the open-source frameworks included in our literature review is provided in ***Table A2*** of Appendix A.

To illustrate how the *Sima* framework can be used in simulating complex systems, data collection procedures, and interventions, we provide an example application in the health domain. Simulated health data allow us to predict the population-level development of risk factors, disease occurrence, and case-specific mortality under the given assumptions. Then, the predictions can be used in evaluating the effects of different policies and interventions in medical decision making in a prescriptive analytics approach. Our example considers the occurrence of stroke, type 2 diabetes, and mortality in the Finnish context studying the impact of nonparticipation on the estimated risk models. The simulation framework can also be utilized in medical decision making, where policies and interventions are optimized under multiple conflicting objectives. We see this as a topic for its own paper, so we concentrate here on demonstrating the functionalities of the framework.

The paper is organized as follows: Section 2 describes the simulator and its core components, such as events and the population. The nuanced practical aspects such as calibration and interventions are also described. Section 3 describes the data-collection process. Section 4 then presents the details of the implementation, including the chosen data structures and libraries, as well as optimization tools for calibration and other purposes. Section 5 illustrates a comprehensive application of the simulator for modeling the occurrence of stroke and diabetes with associated mortality. Section 6 concludes the paper with a discussion and possible directions for future work.

2. Simulation Framework

In this section, we present the key concepts that define our simulation framework in detail. To accomplish the goals we have set out for our ideal simulator, we take into account the fundamental features of the previous section as explicitly as possible. Formal details of the concepts presented in this section are available in Appendix B. ***Figure 1*** shows a flowchart that depicts the operation of a simulator constructed within the framework.

The main operation of the simulator is built on four components: a population, a collection of events, a vector of parameters $\boldsymbol{\theta}$, and a vector of latent variables $\boldsymbol{\psi}$, each of which can change over time. A population is essentially a collection of status variables, which record various details of the simulated individuals, such as age, gender, or disease events. Throughout the paper, we assume that individual status variables are one dimensional and real-valued. This assumption does not rule out, for example, categorical variables, which can be included by encoding them in the standard way with a binary indicator for each category. Events govern the changes that may occur in the status of the population between time points. The parameter $\boldsymbol{\theta}$ defines the functionality of the events, and the latent variables $\boldsymbol{\psi}$ are used to introduce random variation. For example, $\boldsymbol{\theta}$ might define the transition probabilities between states in a simple three-state model with the states "healthy", "sick" and

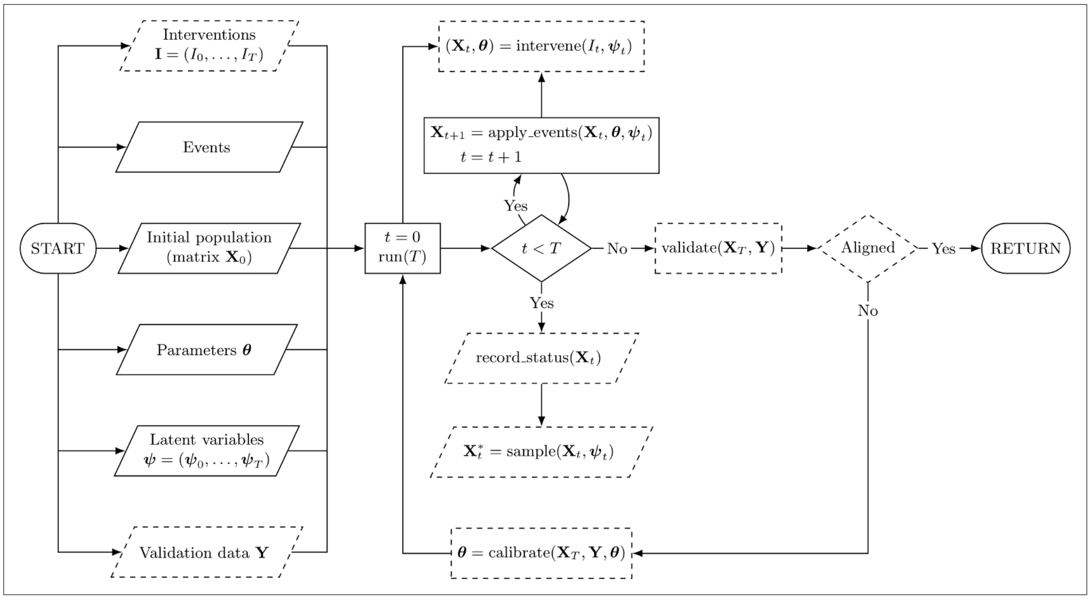
Interventions
$\mathbf{I} = (I_0, \ldots, I_T)$
Events
START
Initial population
(matrix $\mathbf{X}_0$)
Parameters $\boldsymbol{\theta}$
Latent variables
$\boldsymbol{\psi} = (\boldsymbol{\psi}_0, \ldots, \boldsymbol{\psi}_T)$
Validation data $\mathbf{Y}$
$t = 0$
run(T)
$(\mathbf{X}_t, \boldsymbol{\theta}) = \text{intervene}(I_t, \boldsymbol{\psi}_t)$
$\mathbf{X}_{t+1} = \text{apply\_events}(\mathbf{X}_t, \boldsymbol{\theta}, \boldsymbol{\psi}_t)$
$t = t + 1$
Yes
$t < T$
No
validate$(\mathbf{X}_T, \mathbf{Y})$
Aligned
Yes
RETURN
Yes
No
record_status$(\mathbf{X}_t)$
$\mathbf{X}_t^* = \text{sample}(\mathbf{X}_t, \boldsymbol{\psi}_t)$
$\boldsymbol{\theta} = \text{calibrate}(\mathbf{X}_T, \mathbf{Y}, \boldsymbol{\theta})$

Figure 1. Flowchart of a simulator constructed within the Sima framework. Slanted nodes are inputs and outputs, rectangular nodes are processes, and diamond nodes are decisions. Dashed nodes depict optional components. At the "RETURN" node, we return all generated outputs, meaning the final status of the population $\mathbf{X}_T$ and any samples $\mathbf{X}_t^*$ or entire records of the status $\mathbf{X}_t$ that may have been collected during the simulation. The node including "calibrate" should be understood as a procedure that provides a new values for the simulator parameters $\boldsymbol{\theta}$ based on the results of current run and the validation data.

"dead", and together, the transition probabilities and a random value of ψ define the next state for each individual.

2.1. Events

We draw a distinction between two types of events: ones that only modify the status variables of the existing individuals in the population and those that introduce new individuals into the population. We call these *manipulation events* and *accumulation events*, respectively. In both cases, the functionality of the events is based on the parameter $\boldsymbol{\theta}$ and the latent variables $\boldsymbol{\psi}$ specific to the simulator at hand.

Manipulation events are used to describe deterministic changes between two time points in the status variables of the population, either on the individual level or for a collection of individuals simultaneously. A manipulation event can be used, for example, to model stroke incidents based on the status of the current risk factors according to a probabilistic risk model. Our interest lies in the general effects of policy decisions over longer periods of time; thus, we do not focus on phenomena where individuals interact directly, such as the spread of viral infections in a network of individuals. However, there is no fundamental restriction to non-interacting populations, or manipulation events that only operate on the individual level. Interacting individuals can be a hindrance on the scalability of the simulation, because a system of interacting individuals cannot be parallelized simply by partitioning the population appropriately, unlike in the case of independent individuals.

Accumulation events are used to introduce new individuals into the simulated population. For example, an accumulation event could be used to describe the incoming patients of a hospital. The initial members of the population and those added by accumulation events are never removed from the population. In the context of the hospital example, we still want to keep the records of discharged patients. However, a manipulation event may change the status of an individual in such a way that no further manipulation events have any affect on the status of the individual (e.g., mortality, or patient discharge). However, a simulated sampling process may encode missing data in such a way that information on all the status variables of a specific individual or a set of individuals is not available in the resulting sample.

The events are defined as deterministic functions given the parameters and the latent variables which, if assumed to be random variables, give a stochastic nature to the simulator. Another important aspect is the order of events. Because manipulation events map the vector of status variables of a single individual into another vector of status variables, the order of application is significant. For accumulation events, the order of application simply corresponds to a permutation of rows in the resulting population, meaning that the order does not matter.

2.2. Simulator

To live up to its name, a simulator must have the capacity to advance the initial population through the application of manipulation events and accumulation events. At a given time t, the simulator occupies a state S_t, which is an encapsulation of its current population and the parameter configuration. Transitions between states are carried out by the transition function, which is defined by the events of the simulator. In turn, the functionality of the events is governed by the current state, the parameter $\boldsymbol{\theta}$ and latent variables $\boldsymbol{\psi}$.

The transition function is the main workhorse of the entire simulation framework. Given a vector of latent variables, it defines the canonical transformation from one state to the next, that is, from S_t to S_{t+1}. This transformation consists of two parts corresponding to the manipulation events and the accumulation events of the simulator. In a practical implementation, the order of application for manipulation events can become a major performance concern if the population is large. Generating a random order for the manipulation events for every individual requires more computational effort than generating a single order for the entire population.

As an output of a single run of the simulator, the user is provided with a simulation sequence $\{S_t\}_{t=0}^{T}$ of desired length T. The simulation sequence begins from an initial state S_0, which is defined by the user, and a collection of latent variables obtained via pseudo-random number generation (PRNG). This sequence consists of all the states occupied during the simulation, allowing the study of the status variable processes. There are no restrictions for the time resolution. Note that in a single simulation sequence, the parameters are determined by the state at the initial time and remain unchanged throughout the sequence.

2.3. Alignment and Calibration

To ensure a realistic and sensible output, the simulation output is typically aligned to some external targets (***Harding, 2007***; ***Li and O'Donoghue, 2014***; ***Stephensen, 2015***). Such targets may include joint or marginal distributions of specific status variables, expected total or accumulative event occurrences, or event parameter values. Our primary approach to alignment is calibrating the simulator with respect to external data. For this purpose, we need to be able to change its parameter configuration while keeping the system otherwise intact. This is done by the configuration function, which manipulates the current state by altering the values of the parameter vector. More generally, calibration is based on optimization, which involves multiple simulation runs, where the vector of the latent variables is kept fixed between runs, but the parameters are adjusted between each run according to some optimization strategy.

Calibration is an example of parameter estimation where the best parameter values of the model are found by using an optimization algorithm. The calibration of a simulator can be formulated as an optimization problem:

$$\underset{\boldsymbol{\theta}\in\Theta}{\text{minimize}}\; g(\mathbf{Y}, \{\mathbf{Z}_t(\boldsymbol{\theta})\}_{t=0}^{T}),$$

where g is the objective function (with possibly multiple objectives), $\mathbf{Y}$ represents external validation data, and $\{\mathbf{Z}_t(\boldsymbol{\theta})\}_{t=0}^{T}$ is a sequence of a set of status variables from a simulation sequence of length T such that the initial state is defined by the user. The state of the latent variables ψ_t is kept fixed for different values of $\boldsymbol{\theta}$.

The main idea is to make the model compatible with some external validation data. Thus, the objective function should be chosen such that it measures deviations in population-level characteristics between the external validation data and the simulation output instead of the properties of specific individuals. Thus, the quality of the calibration depends on a number of user-dependent factors. The chosen objective function, quality of external data, optimization method used, and length of the simulation sequence should be carefully considered on a case-by-case basis. A common choice for the objective function is a least squares function, where the deviation is measured as

$$g(\mathbf{Y}, \{\mathbf{Z}_t(\boldsymbol{\theta})\}_{t=0}^{T}) = g(\mathbf{Y}, \widehat{\mathbf{Y}}(\boldsymbol{\theta})) = ||\mathbf{Y} - \widehat{\mathbf{Y}}(\boldsymbol{\theta})||^2,$$

where $\widehat{\mathbf{Y}}(\boldsymbol{\theta})$ is a function of the simulation output corresponding to the form of the validation data. In some cases, it may be necessary to scale up the simulation (e.g., increase the number of individuals or extend the length of the simulation) to obtain better calibration. Further, other types of deviation measures can be used instead of least squares, here depending on the properties of data used for calibration. In case of multiple calibration targets, the function g can be multiobjective (see, e.g., ***Enns et al., 2015*** for more details).

As an extension, we may also consider constrained optimization, where some properties of the simulation output are strictly enforced via the optimization procedure. This is closely related to alignment methods that aim to ensure that the output of the simulation conforms to some external criteria by adjusting the output itself, not the model parameters. In our framework, this kind of alignment can be accomplished via suitably constructed manipulation events that are then applied either as a post hoc correction, or during the simulation. In practice, this is accomplished via interventions, which are discussed in the next section.

2.4. Interventions and Counterfactuals

Recalling our requirement for manipulability as outlined in Section 1, the simulator is imbued with the capacity to allow for different types of interventions. The first type concerns a manipulation that either adds or removes individuals or events. This can be accomplished by constructing time-dependent events. A population change at a specific time can be either an accumulation event to add individuals, or a manipulation event to remove individuals by changing the value of status variable that indicates the presence of that individual in the population. On the other hand, event can be constructed in such a way that they only occur at specific time points during the simulation.

The second type of intervention is policy decisions. For example, consider a treatment regime consisting of two different treatments where the decision about the treatment to be prescribed is based on some threshold value of a health indicator, such as blood pressure. A policy decision could involve a change in this threshold value. Supposing that a parameter in $\boldsymbol{\theta}$ is the threshold value, a suitable intervention can be modeled by the configuration function such that only the specific parameter value is changed accordingly.

Finally, the last type of manipulation is the intervention encapsulated by the do-operator. In causal inference, the do-operator enforces a set of variables, making them take constant values instead of being determined by the values of their parents in the associated causal graph (***Pearl, 2009***). When applied to a simulated population, an intervention of this type enables the direct study of causal relationships. For example, we can study the effect of how changing precisely one feature in the entire population affects the outcomes of interest. To accurately simulate the do-operator, the events corresponding to the mechanisms that are desired to be subject to an intervention have to be defined in a way that encodes the intervention as a parameter in $\boldsymbol{\theta}$. This allows for the use of the configuration function to enable the desired actions on the events themselves.

The do-operator also allows for the direct study of counterfactual queries. Consider again the treatment regime with two different treatments. In the simplest case, these two potential outcomes correspond to treatment and control, where no treatment is prescribed. In real-world scenarios, it is impossible to observe both potential outcomes simultaneously from the same study unit, which is known as the fundamental problem of causal inference. We can bypass this problem with ease by simulating two populations that are identical in all aspects, save for the treatment assignment, here while using the same vector of latent variables. More generally, we can apply this approach to study any number of potential outcomes and parallel worlds (***Pearl, 2009***).

3. Data Collection Process

Simulating data collection adds another dimension to the simulation framework. Conceptually, the idea is the same as in causal models with design, where the causal model (event simulator) and data collection process together define the observed data (***Karvanen, 2015***). Data collection is important, especially in method development, because statistical methods aim to estimate the underlying processes based on limited data. Usually, data on most health variables cannot be collected daily. For instance, it is not realistic to assume that measurements requiring laboratory testing of a blood sample would be available for a population cohort every day. In a typical longitudinal study, most of the daily measurements are in fact missing by design. Using data from a small number of time points to learn about the underlying continuous-time causal mechanism may lead to unexpected results if the data collection process is not appropriately accounted for (***Aalen et al., 2016***; ***Strohmaier et al., 2015***; ***Albert et al., 2019***).

Even if the data are planned to be collected, things may go wrong. Real data are almost never of a perfect quality. Data may be incorrectly logged, completely lost, or otherwise degraded. Some values or entire study units may suffer from missing data or measurement error. In surveys and longitudinal studies nonparticipation, dropout, and the item nonresponse are always possible; this is a problem that cannot always be remedied even by recontacting the participants who dropped out or decided not to answer the survey. More generally, participation in a study may be influenced by a number of factors, hence resulting in selective participation. The aforementioned aspects are also an important component of our simulation framework. Obtaining a smaller subset than the full population after or during the simulation is an important feature for many data analytical tasks. The presented framework includes a highly general interface for simulating a realistic sampling process from a synthetic population.

In the presented framework, we are able to construct missing data mechanisms of any type, especially those recognizable as belonging to the traditional categories of missing completely at random, missing at random, and missing not at random (***Little and Rubin, 2002***). Having constructed the mechanism, we can obtain a sample from the full population, where the mechanisms dictate the values or observations that will be missing in the sample. Multiple imputation is a widely applied and studied method for replacing missing data with substituted values (***Van Buuren, 2018***). A simulated sample with missing data can be used to assess the possible bias introduced by imputation and the representativeness of any results obtained from the imputed data by comparing the results to those obtained from the same sample without any missing data.

4. Implementation

We chose R as the programming language and environment for the implementation of the simulator (***R Core Team, 2020***). R provides a vast multitude of computational and statistical methods as open-source packages via the Comprehensive R Archive Network (CRAN). Currently, the CRAN repository features over 15,000 packages that encompass methodology from a wide-ranging variety of scientific disciplines, including physics, chemistry, econometrics, ecology, genetics, hydrology, machine learning, optimization, and social sciences. This is especially useful when constructing manipulation events, because statistical models and other methods from state-of-the-art research can be directly implemented via events in the simulator to accurately model phenomena of interest. After carrying out the simulation, the synthetic data can be effortlessly exported from R to various external data formats to be used in other applications, if desired.

4.1. Data Tables

We opted for the data tables implemented in the data.table package to act as the container for the population (***Dowle and Srinivasan, 2019***). Compared with standard R data containers, data tables have a number of advantages, such as efficient subset construction, in-memory operations, internal parallelization, and a syntax that resembles that of SQL queries. Data tables are also highly memory efficient and capable of processing data with over one billion rows. The choice was further motivated by the extremely competitive performance of data tables compared with other popular open-source database-like tools in data science. A regularly updated operational benchmark can be found at https://h2oai.github.io/db-benchmark/ that measures the speed of data tables and other popular packages such as the datatable Python package and the DataFrames.jl Julia package in carrying out various tasks with input tables up to a size of one billion rows and nine columns (which corresponds to 50GB in memory). Internally, the data.table package does not depend on any other R packages, making it a future-proof choice and easily extendable by the user.

The operational performance of data tables is exemplified both in carrying out computations within groups in a single table and joining multiple tables. For our purposes, efficient operations within a single table are the primary interest. As an example, consider a manipulation event that specifies mortality because of stroke as a logistic regression model. In a data table, the event probabilities for the group that has had at least one stroke can be efficiently computed using vectorized covariates (a subset of the status variables) without making any unnecessary copies of the objects involved.

4.2. Reference Classes

Manipulation events, accumulation events, and the simulator itself are defined as R6 reference classes. R6 is an object-oriented programming system for R provided through the R6 package (***Chang, 2019***). R6 does not have any package dependencies and does not rely on the S4 or S3 class systems available in base R, unlike the default reference class system. R6 features several advantages over the built-in reference class system, such as vastly improved speed, ease of use, and support for public and private class members. The primary reason to adopt a reference class system is the mutability of the resulting objects; they are modified in place unlike standard R objects, which are always copied upon modification or when used in function calls. See the R6 package vignette titled "Performance" for speed and memory footprint comparisons between R6 and base R reference classes (also available at https://r6.r-lib.org/articles/Performance.html).

The typical structure of an R6 class object consists of both fields and methods. Following the standard object-oriented programming paradigm, fields and methods can be either public or private depending on whether they should be accessible externally or only within the class object itself. It is also possible to construct so-called active fields that are accessed as fields but are defined as functions, just like methods are.

The classes for both manipulation events and accumulation events are inherited from a superclass that is simply called an Event. An Event can be given a name and a short description to provide larger context. More importantly, an Event contains the fields mechanism and par, which determine the functionality and parameter configuration of the Event, respectively. The mechanism can be any R expression, and its functionality may depend on the parameters defined by par. The Event superclass provides a template method called apply to evaluate the mechanism, but the actual behavior of this method is determined by the type of the event, and fully controlled by the simulator. The inherited class ManipulationEvent defines the type of events of its namesake. For events of this class, the apply method provides access to the current status of the simulated population and evaluates the mechanism in this context, but cannot introduce new individuals. On the other hand, the inherited class AccumulationEvent allows its mechanism to add new members to the simulated population via the apply method, but is prohibited from manipulating the current population.

The Simulator class implements all of the primary functionality of the framework, including population initialization, state transitions, calibration, sampling, and interventions. The constructor of the class takes the lists of ManipulationEvent and AccumulationEvent objects as an input and calls a user defined function initializer, which constructs the initial population. The run method is the transition function of the simulator, and it uses the apply methods of the event objects given in the constructor at each time point to advance the simulation. The user may control how many state transitions are carried out sequentially for a single run command. The simulator can be instructed to conduct the state transitions in parallel by calling the start_cluster method beforehand. This method connects the simulator to a user defined parallel backend, such as a cluster initialized using the doMPI package, and distributes the population into chunks. Parallel computation can be stopped with the stop_cluster method, which also merges the individual population chunks back together. Configuration of the event parameters is managed by the configure method, which takes as an input the amount of state transitions to be used, the set of parameters to be configured, and an output function to be used as a part of the objective function in the optimization for the calibration. The output function can be simply a statistic that is obtained from the resulting simulation sequence to be compared with a set of validation data. For an example, see the objective function defined for calibration in Section 5.3. Simulated data collection is performed via the get_sample method, which takes as its input a user defined sampling function that returns a subset of the full population. Interventions can be applied at any point of the simulation via the intervene method. This method can apply a set of manipulation events, accumulation events, or parameter configurations to change the current state of the simulator.

4.3. Parallel Computation

The task of simulating the synthetic population in the described framework can be considered "embarrassingly parallel" (***Herlihy and Shavit, 2011***). Communication between the parallel tasks is required only when there is a need to consider the population as a whole, such as for calibration or for constructing a time series. In the ideal case, the end state of the population is solely of interest, which minimizes the required communication between tasks.

Parallel CPU processing functionality is provided by the doParallel and foreach packages (***Microsoft and S. Weston, 2019***; ***Microsoft and S. Weston, 2020***). These packages allow for parallel computation on a single machine with multiple processors or cores, or on multiple machines. Alternatively, Message Passing Interface (MPI) can be used through the packages doMPI and Rmpi (***Weston, 2017***; ***Yu, 2002***). More generally, any parallel adaptor to the foreach looping construct is supported. The parallel approach takes full advantage of the independence assumption of individuals. This allows us to partition the population into chunks, typically of equal size at the start of the simulation run, and to evaluate the events in each chunk separately by distributing the chunks to a desired number of workers. Shared memory between the workers is not required, and the full unpartitioned population does not have to be kept in memory during the simulation, even for the supervisor process. The dqrng package (***Stubner, 2019***) allows for multiple simultaneous random number streams, in turn enabling the use of different streams for each worker and making the parallel simulation sensible. The initial seeds for these streams can be manually set by the user for reproducibility. For more details on PRNG in the implementation, see Appendix C.

Additional overhead in both memory consumption and processing time is incurred by having to distribute metadata on the population and other necessary information to the subprocesses, such as variable names and the Event instances. Furthermore, constructing the chunks and reintegrating them back together at the end of simulation is an added hindrance. However, these operations do not

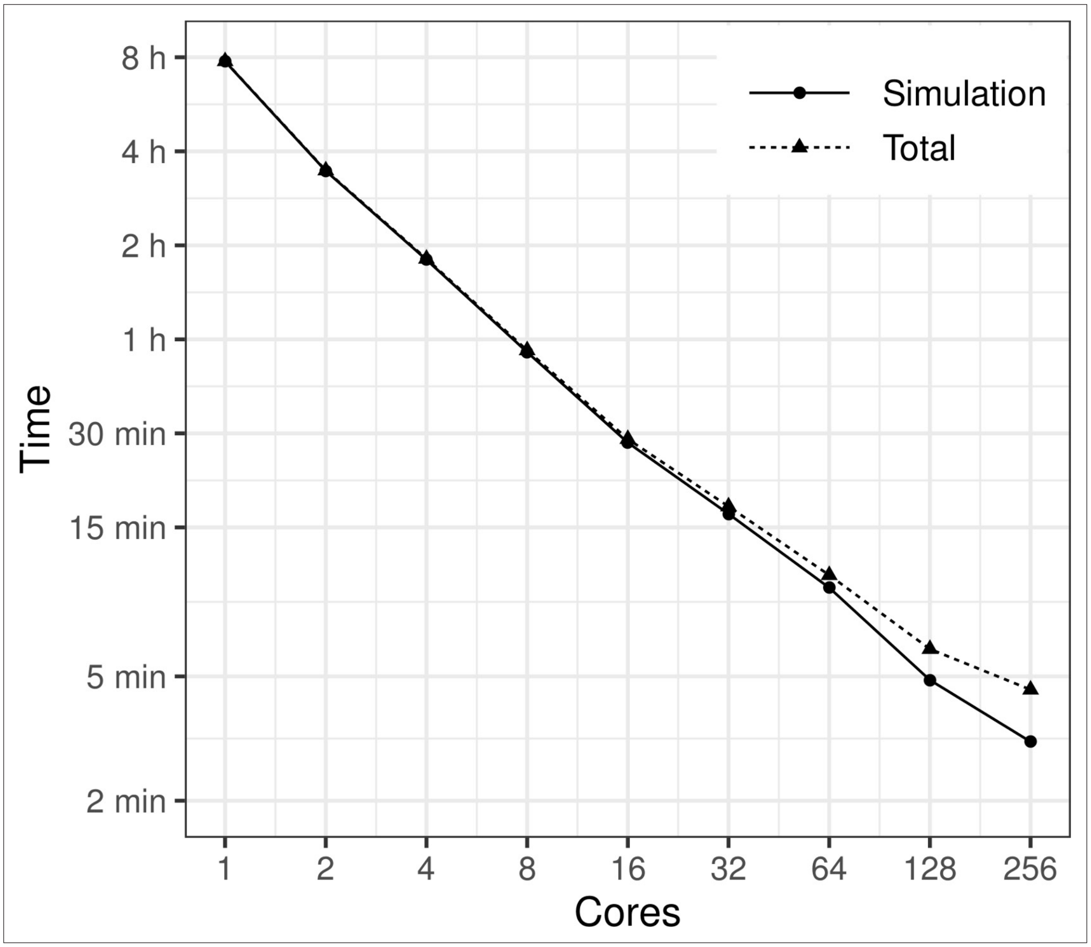

Figure 2. Performance benchmark using a population of 10 million individuals, 5 events and a simulation period of 10 years on a time scale of one day. Total time also includes the time it takes to generate the initial population, initialize the parallel computation cluster and to distribute the data to the cores. MPI was used to conduct the benchmark.

depend on the length of the simulation run, meaning that the overhead from these particular tasks becomes negligible as the length of the run grows.

Figure 2 shows how the processing time is affected by the number of computer cores. The benchmark scenario consists of a population of 10 million individuals, five event types, and a simulation period of 10 years, where the events are applied daily. More specifically, the scenario is the one described in Section 5. The results show that the runtime of the simulation is cut almost in half every time the number of cores is doubled (geometric mean of the consecutive runtime ratios is 0.534). As expected, the proportion of the overhead resulting from the communication between cores and other parallel tasks also grows as the number of cores is increased. It should be noted that in the runs involving up to 32 cores only one computing node was used. For the runs with 64, 128, and 256 cores, the tasks were distributed to three, four, and 10 computing nodes, respectively.

4.4. Optimization

In this paper, optimization is used for calibration of the simulator parameters $\boldsymbol{\theta}$. The optimization method used needs to be selected based on the properties of the objective function used for calibration. There exist many optimization tools implemented in R, and we briefly describe the optim function that provides a variety of optimization algorithms. The optim function is developed for general purpose optimization, where box constraints can be set for the decision variables. The function includes implementations of the Nelder-Mead, quasi-Newton, and conjugate gradient algorithms. In the example given in Section 5, we have used the least squares function, which is differentiable, and in the corresponding calibration problem, we do not have explicit bounds for the calibrated parameters, and we do not calculate any gradients for the objective function. Therefore, we will use the Nelder-Mead algorithm from the optim function which is suitable for our purposes (***Nelder and Mead, 1965***). The Nelder-Mead algorithm does not use any gradient information but works on a set of $N+1$ points in a optimization problem with N decision variables. At each iteration, the main idea is to improve the objective function value of the worst point by using reflections and contractions to identify locally optimal points.

5. Example on Diabetes and Stroke

We present an example use case of the simulator, where we model the occurrence of stroke, type 2 diabetes, and mortality in a synthetic population of 3.6 million individuals, corresponding to the Finnish population size of persons over 30 years of age in the year 2017. The goals of the illustration are threefold. First, we wish to exemplify the use of the simulation framework in building simulators. Second, we aim to build a simulator that can realistically model the occurrence of stroke, type 2 diabetes, and mortality in the Finnish population. Third, we want to illustrate the use of the simulator in methodological comparisons and carry out a simulation where we can study the impact of nonparticipation on the estimated risk model of stroke. See https://santikka.github.io/Sima/articles/example.html for the annotated R codes and data files used in the example.

We generate a synthetic population based on the population of Finland, including the status of the following risk factors for each individual: smoking, total cholesterol, high-density lipoprotein (HDL) cholesterol, systolic blood pressure, age, sex, body mass index, high blood glucose, waist circumference, diabetes, and whether either parent has suffered a stroke. A single state transition in the simulator corresponds to one day in real time. The events include the occurrence of stroke, occurrence of diabetes, death after stroke, and death without suffering a stroke. The events modify the status variables that describe the health status of the individual. The daily probabilities of the events depend on the risk factors. The goal is to create a synthetic follow-up study about this population with selective nonparticipation and use the obtained sample to estimate the effect of the risk factors on the risk of stroke and diabetes. Because the daily event probabilities are low, it is unlikely that a person has more than one event per day, so the simulation can be used to study competing risks.

5.1. The Models for the Events

Mortality is modeled separately for individuals who have suffered a stroke and for those who have not. A stroke is considered fatal if an individual dies within 28 days of the initial stroke incident. For simplicity, it is assumed that there can be at most one stroke incident for any individual. The long-term stroke survival rates are modeled according to the results by ***Brønnum-Hansen et al. (2001)***, and the remaining mortality is calibrated to match the Finnish population in the year 2017, here based on two factors: mortality rates in each age group and the proportion of deaths because of stroke out of all deaths over one year. We use the official mortality statistics from Statistics Finland for this purpose (***Official Statistics of Finland (OSF), 2017***).

The model for the occurrence of stroke is based on the FINRISK 10-year risk score (***Vartiainen et al., 2016***). In the risk score calculation, the 10-year risk of cardiovascular disease is modeled as a sum of two risks: the risk of coronary heart disease and the risk of stroke, both of which are modeled via logistic regression. For our purposes, we use only the stroke portion of the risk score model. The one-day risk for the corresponding manipulation event in obtained by assuming a Poisson process for the stroke. Similarly, the model for the occurrence of type 2 diabetes is based on the FINDRISC (Finnish Diabetes Risk Score) 10-year risk score (***Lindström and Tuomilehto, 2003***). Both the FINRISK and the FINDRISC risk score models take into account a number of risk factors. Because these models only consider individuals over the age of 30, we also restrict our simulated population in the same way in terms of age.

The 10-year risk of stroke is modeled separately for both genders while also including the same risk factors. More specifically, the risk factors for stroke include age, smoking status, HDL cholesterol, systolic blood pressure, diabetes status, and an indicator whether either parent of the individual has suffered a stroke. We let $C_{it}(t^*)$ denote a Poisson process that counts the number of stroke events that have occurred after t^* days since time t for each individual i, hence giving us the following relation:

$$\mathrm{logit}(\mathbb{P}(C_{it}(3650) > 0)) = \mathbb{I}(G_i = 0)\mathbf{x}_{it}^{\top}\boldsymbol{\beta}_m + \mathbb{I}(G_i = 1)\mathbf{x}_{it}^{\top}\boldsymbol{\beta}_f,$$

where G_i denotes the gender (1 = woman), $\mathbf{x}_{it}^{\top}$ is the row vector of status variables for individual at time t, and is an indicator function. The parameter vectors for the stroke model are denoted by $\boldsymbol{\beta}_m$ and $\boldsymbol{\beta}_f$ for men and women, respectively. The values of the model parameters are obtained from the FINRISK risk score. From the 10-year risk, we first obtain the intensity τ_{it} of the underlying Poisson process via the following relation:

$$\mathbb{P}(C_{it}(t^*) = m) = \tfrac{(\tau_{it}t^*)^m}{m!}e^{-\tau_{it}t^*}.$$

When applied to the 10-year risk we have the following:

$$\tau_{it} = -\tfrac{\log(\mathbb{P}(C_{it}(3650)=0))}{3650}.$$

The one-day risk of stroke is now modeled as follows:

$$\mathrm{logit}(\mathbb{P}(S_{it} = 1|S_{it} = 0)) = 1 - \exp(-\tau_{it}),$$

where S_{it} is an indicator denoting whether individual *i* has suffered a stroke at time t or earlier.

The one-day risk of diabetes is obtained in the exact same way from the model of the 10-year risk of diabetes as the risk of stroke. For the diabetes FINDRISC model, the set of risk factors is different, and the same model parameters are now used for both genders. The risk factors used in the model are age, body mass index, waist circumference, use of blood pressure medication, and high blood glucose.

We use the findings reported in a study by ***Brønnum-Hansen et al. (2001)*** to model the long-term survival after stroke. The study was conducted as a part of the World Health Organization (WHO) MONICA (Monitoring Trends and Determinants in Cardiovascular Disease) project during 1982–1991 and consisted of subjects aged 25 years or older in the Glostrup region of Copenhagen County in Denmark. In the study, a number of stroke subtypes were considered, but for our purposes, we simply use the aggregate results over all variants to model the mortality due to stroke.

The cumulative risk of death reported by ***Brønnum-Hansen et al. (2001)*** is 28%, 41%, 60%, 76%, and 86% at 28 days, 1 year, 5 years, 10 years, and 15 years after the stroke, respectively. Because the survival probability decreases rapidly during the first 28 days, we construct two models taking into account this 28-day period, and the remaining period up to 15 years from the first stroke incident. The probability that an individual who has suffered a stroke does not die at time t given that they are still alive at time $t - 1$ and have survived for over 27 days is modeled using a regression model with a logistic link:

$$\mathrm{logit}(\mathbb{P}(Y_{it} = 0|Y_{i,t-1} = 0, S_{it} = 1)) = \alpha_1 + \alpha_2 \exp(\alpha_3(t_i - 27)), \quad t_i > 27, \tag{1}$$

where $Y_{it} = 0$ denotes the survival of individual *i* at time t. The time from the stroke occurrence in days is t_i for individual *i*. The values of the parameters α_1, α_2, and α_3 are found by directly fitting the model of (1) to the reported survival rates.

The survival during the first 28 days from the stroke incident is assumed to be constant for each day:

$$\mathrm{logit}(\mathbb{P}(Y_{it} = 0|Y_{i,t-1} = 0, S_{it} = 1)) = \alpha_0, \quad t_i < 28.$$

The reported survival during the initial period varies drastically between the different stroke variants and levels off as more time passes from the first stroke incident. Because our focus lies in simulating a period of 10 years, we consider this simple model to be sufficient for such a small time window of the entire simulation period.

The secondary mortality from other causes is modeled using a Weibull distribution with the following parametrization for the density function:

$$f(x) = \tfrac{\lambda}{\kappa}\left(\tfrac{x}{\kappa}\right)^{\lambda-1}\exp\left(-\left(\tfrac{x}{\kappa}\right)^{\lambda}\right), \quad x > 0,$$

where $\lambda > 0$ is the shape parameter and $\kappa > 0$ is the scale parameter of the distribution. The probability that individual i dies at a specific time t without having previously suffered a stroke given that they are still alive at time $t-1$ is defined by the following:

$$\mathbb{P}(Y_{it} = 1 | Y_{i,t-1} = 0, S_{it} = 0) = F(A_{it}; \lambda, \kappa) = 1 - \exp\left(-\left(\frac{A_{it}}{\kappa}\right)^{\lambda}\right),$$

where A_{it} is the age of individual at time t is the cumulative distribution function of the Weibull distribution.

5.2. Initial Population

Equally important with the accurate modelling of the events is the generation of realistic initial values for the various health indicators, risk factors, and other variables in the population. Each individual is defined as being alive at the initial state, and the gender is generated simply by a fair coin flip. The ages are generated separately for both genders based on the age structure of Finland in the year 2017 from the official statistics, here with the restriction that every individual is at least 30 years old (***Official Statistics of Finland (OSF), 2017***).

We use Finnish data (the 20% sample available as open data) from the final survey of the MONICA project to obtain the initial values for BMI, waist circumference, cholesterol, HDL cholesterol, and systolic blood pressure (***Tunstall-Pedoe et al., 2003***). First, we transform these variables in the data into normality using a Box-Cox transformation separately in each age group for both genders and based on smoking status (***Box and Cox, 1964***). Smoking is defined here as a dichotomous variable so that an individual is considered a smoker if their self-reported smoking frequency for cigarettes, cigars, or pipe is either "often" or "occasionally" in the MONICA data. Next, we fit a multivariate normal distribution to the transformed data in each subgroup and generate values from this distribution to take the correlation structure of the variables into account. Finally, the inverse transformation is applied to the generated values. We also apply a post-hoc correction to the generated values by restricting them to the ranges observed in the Health 2000–2011 study data. Values that fall outside the valid ranges are replaced by other generated values that fall within the valid ranges from individuals in the same age group that are of the same gender and have the same smoking status.

The report by ***Koponen et al. (2018)*** provides the distributions of smoking and stroke in a representative sample of the general Finnish population in the year 2017. For example, the study reports the proportion of nonsmokers for each age group starting from 30-year-olds with 10-year intervals by gender. For the FINRISK model, the variable that either parent of an individual has suffered a stroke is initialized by setting the probability of having at least one parent who has suffered a stroke to be the sum of the gender-specific, age-adjusted stroke incidence rates for persons over 50 years old in the FinHealth 2017 study. For smoking, we use the reported gender and age-specific proportions directly as parameters for binomial distributions to generate the initial values. For those individuals in the initial population who have suffered a stroke, an accumulated survival time is generated from a Weibull distribution fitted to the 1-year, 5-year, 10-year and 15-year survival rates that were reported by ***Brønnum-Hansen et al. (2001)*** while ensuring that the stroke could not have occurred before 30 years of age.

After all other initial values have been generated, the initial diabetes status is generated by first using the FINDRISC model to predict the 10-year risk for each individual in the initial population, and then adjusting the individual risks such that the total expected gender-specific diabetes prevalence matches that of Finland in the year 2017 (15% for men and 10% for women). The adjusted risks are then used as the probabilities of having diabetes at initialization for each individual.

5.3. Calibration

We employ calibration to obtain the values for the parameters λ and κ of the non-stroke-related mortality model, as well as for the parameter α_0 of the fatal stroke model. The objective function g compares the official mortality statistics with those obtained from the simulation for a period of one year.

$$g(\mathbf{Y}, \widehat{\mathbf{Y}}(\boldsymbol{\theta})) = \textstyle\sum_{k=30}^{99} w_k \left[\log(\hat{y}_k) - \log(y_k)\right]^2 + w^* \left[\log(\hat{p}) - \log(p)\right]^2 \qquad (2)$$

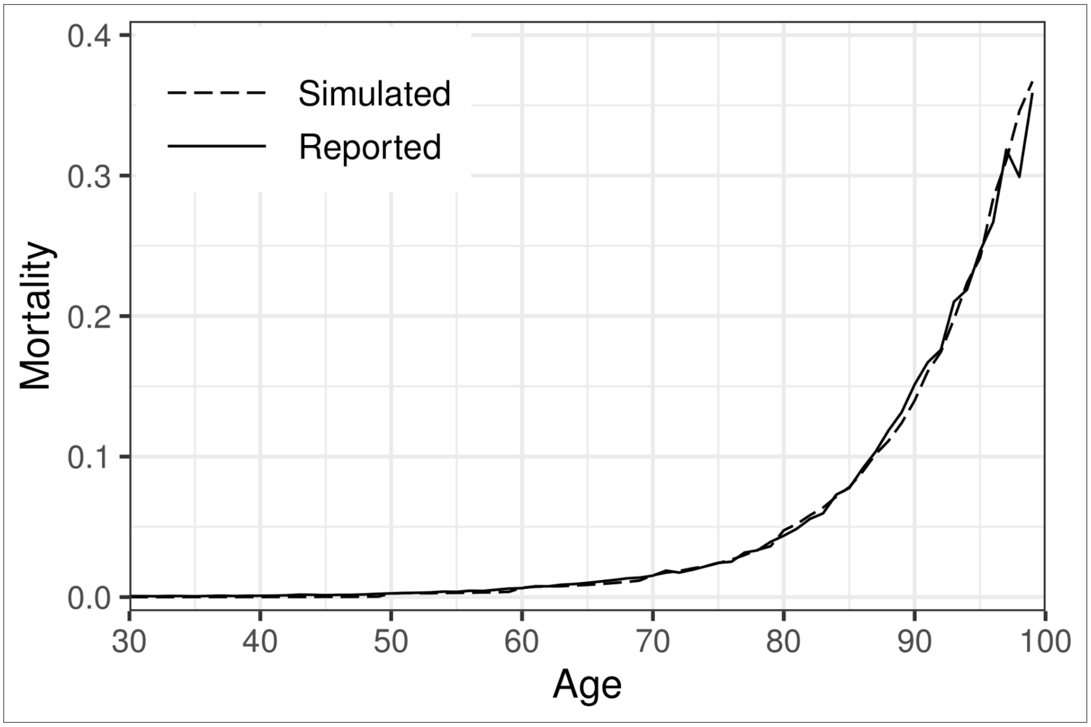

Figure 3. Simulated mortality in a population of 3.6 million individuals after calibration and official mortality statistics of Finland for the year 2017 by age. A population of one million individuals was used for the calibration.

where $\widehat{\mathbf{Y}}(\boldsymbol{\theta}) = (\hat{y}_{30}, \ldots, \hat{y}_{99}, \hat{p})$ is the simulation output with the event parameters set to $\boldsymbol{\theta}$, w_k and w^* are weights, $\hat{y}_{30}, \ldots, \hat{y}_{99}$ denotes the estimated mortality for each age group k, and $\hat{p}$ is the estimated proportion of stroke deaths. $\mathbf{Y}$ denotes the corresponding true numbers $(y_{30}, \ldots, y_{99}, p)$ obtained from the official statistics. The weights w_k were chosen so that $w_k = 1$ when $k < 80$ and $w_k = 100$ otherwise. In the framework, the calibration is carried out by implementing the objective function g as an R function that takes advantage of the configure method of the simulator to change the parameter values during the optimization.

In this case, we observed that the optimum is reached faster when using a weighted least squares approach, because the mortality is much higher in the older age groups. On the other hand, the total yearly mortalities for younger age groups are close to zero, meaning that the size of the simulated population must be sufficiently large for the estimates to be accurate. The value 2000 was chosen for the weight w^*, which roughly translates to equal importance of the age-group-specific mortalities and the proportion of strokes in the optimization.

We found that increasing the population size beyond one million individuals did not change the results of the calibration in this scenario. We also considered more elaborate weighting schemes, but the effect of the weights diminishes as the population size grows. The described weights were chosen based on trial and error with small populations (less than 10,000 individuals). ***Figure 3*** shows the simulated mortalities after calibration for each age group, as well as those obtained from the official statistics for ages 30–99. In the year 2017, 7.5% out of all deaths in Finland were caused by a stroke (classification falling under ICD-10 Chapter IX I60–I69). For the synthetic population, the same number is 7.8%.

5.4. Sampling and Nonparticipation

We simulate a data collection process where individuals from the simulated population are invited into a health examination survey. The measurements of the relevant risk factors are taken at baseline, and the study participants are then subsequently followed for the 10-year period. In real-world studies, participation may depend on a number of factors, especially when dealing with sensitive information about an individual's health. This leads to selective nonparticipation. We use a logistic regression

Table 1. Estimated parameter values, odds ratios, and their 95% confidence intervals for the model of baseline nonparticipation.

	$\widehat{\rho_k}$	$\widehat{\exp(\rho_k)}$
Intercept	−9.48 (−10.68, −8.32)	
Gender	−0.31 (−0.62, −0.01)	0.73 (0.54, 0.99)
Age	0.11 (0.10, 0.12)	1.12 (1.11, 1.13)
BMI	−0.09 (−0.15, −0.03)	0.91 (0.86, 0.97)
Smoking	0.69 (0.44, 0.94)	1.99 (1.56, 2.56)
Waist Circumference	0.04 (0.02, 0.06)	1.04 (1.02, 1.06)

model to simulate the effect that the relevant status variables have on the probability of nonparticipation at baseline. The chosen model is defined as follows:

$$\mathrm{logit}(\mathbb{P}(M_i = 1)) = \rho_0 + \rho_1 G_i + \rho_2 A_i + \rho_3 B_i + \rho_4 O_i + \rho_5 W_i, \tag{3}$$

where M_i is an indicator of nonparticipation and G_i, A_i, B_i, O_i, and W_i are the gender (1 = woman), age (in years), body mass index (BMI), smoking indicator, and waist circumference (in cm) of individual at baseline, respectively. Much like the calibration described in the previous section, the nonparticipation model (3) is implemented in the framework as an R function, and provided to the get_sample method of the simulator as an argument.

The estimates for the parameters ρ_k, $k = 0, \ldots, 5$ are based on data from the Health 2000–2011 study (***Härkänen et al., 2016***) and are shown in ***Table 1***, along with their 95% confidence intervals.

5.5. Simulation Results on Stroke Incidence and Risk Factors

After generating the initial population, we then proceeded to run the simulation for a 10-year period. From the initial population of 3.6 million individuals, a subset of 10,000 individuals were randomly sampled to be invited into a health examination survey with nonparticipation occurring at baseline. Individuals who had already suffered a stroke were excluded.

From the initial invitees, 336 individuals were excluded because of a previous stroke incident. Afterwards, 1,644 invitees decided not to participate. In this scenario, we assume that there is no item nonresponse for the 7,900 remaining participants. Much like the FINRISK risk score, we model the 10-year risk of stroke separately for both genders via a logistic regression. We compare the results from; A) the sample with baseline nonresponse, B) the sample without nonparticipation, and C) the entire synthetic population. ***Tables 2 and 3*** show the results for the parameter estimation for men and women, respectively.

The incidence of stroke in sample A was 543 of 100,000 per year whereas in sample B, the same number is 728 of 100,000 per year, showing the bias caused by the baseline nonresponse because the incidence in the full population in sample C is 723 of 100,000 per year. As expected, the odds ratio estimates do not differ greatly between samples A and B because the missingness mechanism of (3) is ignorable because it does not depend on the stroke occurrence itself.

Although some discrepancies remain between the true values and the estimates even when using the entire synthetic population, namely in the case of smoking and diabetes, overall, the results indicate that adapting the one-day risk from the 10-year risk by assuming a Poisson process is a valid strategy here. The example further highlights the importance of taking the underlying correlation structure of the risk factors into account in the initial data generation. For example, the MONICA data do not contain information on diabetes, so some loss of information likely occurs when the initial diabetes status is based purely on the FINDRISC model. Similarly, the initial values for smoking are generated based on the FinHealth 2017 study, and the distribution of smokers has changed significantly in Finland from 1992 when the MONICA data were collected. Furthermore, our definition of a smoker is not identical to the original FINRISK study (***Vartiainen et al., 2016***), where a person was considered a smoker if they had smoked for at least a year and had smoked in the last month. In the study, it is also not reported whether only cigarette smokers were considered or if, for example, cigar

Table 2. Estimated odds ratios and their 95% confidence intervals for the model of 10-year risk of stroke for men.

	A	B	C	FINRISK
Age	1.11 (1.10, 1.13)	1.10 (1.10, 1.11)	1.10 (1.09, 1.10)	1.12
Smoking	2.13 (1.52, 2.96)	1.67 (1.27, 2.19)	1.53 (1.50, 1.55)	1.65
Systolic blood pressure	1.01 (1.00, 1.02)	1.01 (1.01, 1.02)	1.02 (1.02, 1.02)	1.02
HDL cholesterol	0.73 (0.48, 1.07)	0.67 (0.49, 0.91)	0.65 (0.64, 0.66)	0.64
Diabetes	2.30 (1.67, 3.13)	2.38 (1.87, 3.02)	2.31 (2.28, 2.34)	2.41
Parents' stroke	1.25 (0.83, 1.84)	1.41 (1.04, 1.89)	1.33 (1.31, 1.35)	1.34

Table 3. Estimated odds ratios and their 95% confidence intervals for the model of 10-year risk of stroke for women.

	A	B	C	FINRISK
Age	1.06 (1.05, 1.08)	1.05 (1.04, 1.06)	1.05 (1.05, 1.05)	1.07
Smoking	1.67 (1.01, 2.66)	1.62 (1.05, 2.44)	1.40 (1.37, 1.44)	1.52
Systolic blood pressure	1.01 (1.00, 1.01)	1.01 (1.01, 1.02)	1.01 (1.01, 1.01)	1.01
HDL cholesterol	0.50 (0.32, 0.76)	0.54 (0.37, 0.78)	0.45 (0.44, 0.46)	0.47
Diabetes	3.70 (2.54, 5.34)	3.37 (2.43, 4.62)	3.21 (3.16, 3.27)	3.45
Parents' stroke	2.18 (1.42, 3.27)	2.02 (1.39, 2.87)	1.69 (1.65, 1.72)	1.73

and pipe smokers were included as well. The MONICA data do not provide enough information on smoking history of the participants to fully match the FINRISK definition.

In general, constructing a daily-level simulator is challenging if only long-term prediction models are available. In a more ideal setting, we would have access to survival models, and, thus, more accurate information on the relation between the risk and risk factors because no information would be lost via discretization into binary outcomes. Additionally, data containing information on the joint distribution of the risk factors is crucial for replicating the correlation structure accurately in the synthetic intitial population.

5.6. Simulation Results on the Effect of Interventions

We demonstrate the use of interventions in the framework by considering the causal effect of salt intake on the risk of stroke as mediated by blood pressure. The typical salt intake in many countries is 9–12 g/day, while the recommended level is 5–6 g/day (***WHO, 2003***; ***He et al., 2013***). Based on a meta-analysis (***He et al., 2013***), a 100 mmol (6 g of salt per day) reduction in 24-hour urinary sodium was associated with a fall in systolic blood pressure of 5.8 mmHg. We consider three scenarios. The first scenario corresponds to the full population sample without interventions. In the second scenario, it is assumed that the food industry reduces the salt content in all products, so the daily sodium intake is 1 g lower. The systolic blood pressure will then be 0.97 mmHg lower on average. In the third scenario, an intervention is targeted to individuals with high blood pressure, which is defined here as systolic blood pressure at or above 140 mmHg. These individuals are advised to stop adding salt in cooking and at the table. It is assumed that 50% of the target group will follow the advice. Based on a recent analysis (***Karvanen et al., 2021***), this intervention is expected to reduce systolic blood pressure on average by 2.0 mmHg among the compliers. In the framework, both interventions are implemented as ManipulationEvent instances and applied via the intervene method at the start of the simulation.

Figure 4 shows the total number of new stroke cases in the three scenarios. We see that in this example, the intervention where the food industry reduces the salt content in all products is the most effective at reducing the number of total stroke cases, while the advice to stop adding salt in

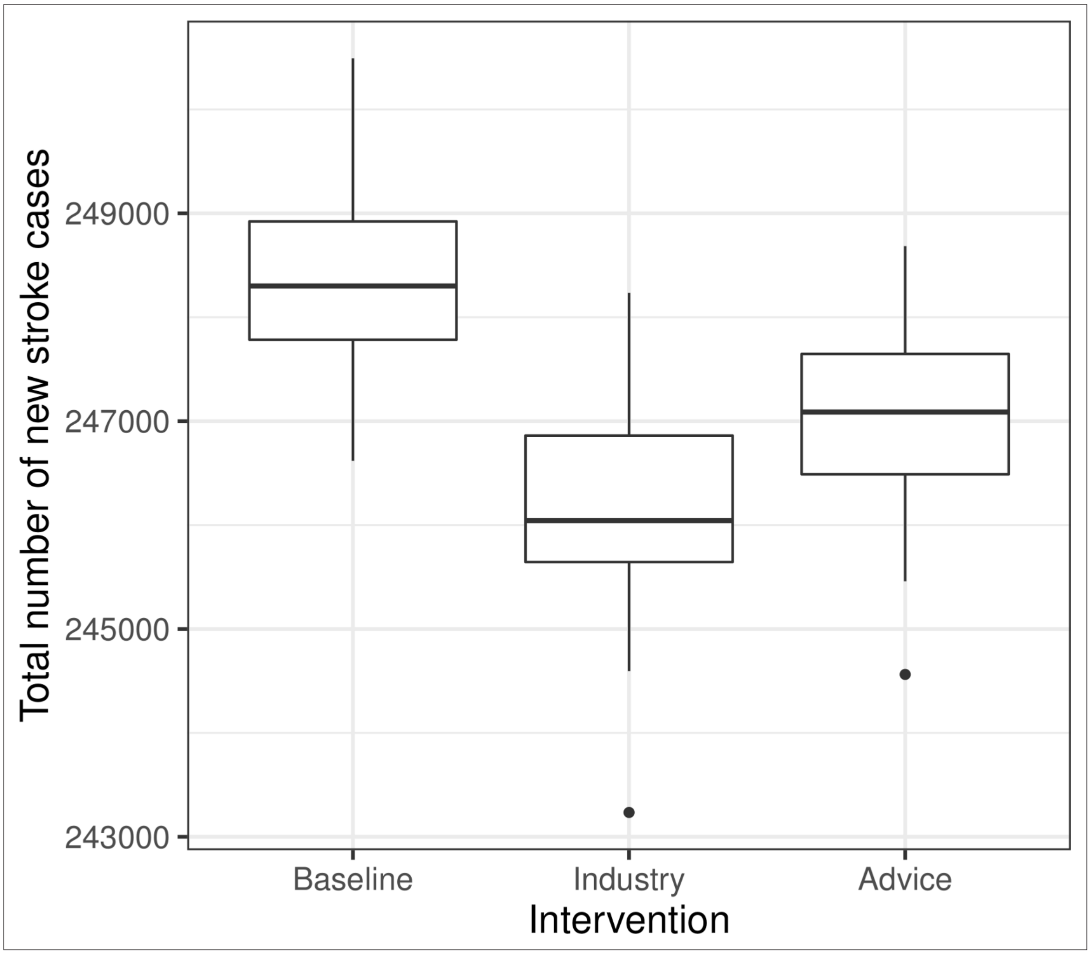

Figure 4. Boxplots of the total number of new stroke cases in the three scenarios from 100 replications of each scenario (from the same initial population in each scenario and replication). "Baseline" is the scenario without interventions, "Industry" corresponds to the salt content reduction carried out by the food industry, and "Advice" is the recommendation regarding salt use targeted towards individuals with high blood pressure.

cooking and at the table targeted toward individuals with high blood pressure is slightly less effective in comparison.

6. Discussion

In this paper, we have presented a framework for the realistic data generation and simulation of complex systems, demonstrating its use in the heath domain. The framework implements functionalities that are in line with the features listed in the Introduction. The framework supports individual-level events on an arbitrary time scale and contains functions to calibrate the simulation parameters with external data sources. These functionalities, together with sufficient knowledge for modeling data-generating mechanisms, are the prerequisites for realistic simulations. The framework has a modular structure, which allows for simulating manipulations of different types. The modular structure also makes it possible to simulate complex data collection processes. The state-of-the-art data structures, efficient random number generation, and built-in parallelization ensure the scalability of the framework, making it possible to run daily-level simulations for populations of millions individuals for decades. The simulation runs are fully reproducible. An open-source implementation in R makes the framework directly available for statistics and data science communities.

The main use cases of the simulation framework are predicting the development of risk factors and disease occurrence, evaluating the impact of interventions and policy decisions, and statistical method development. In the first two use cases, the support for calibration, the support for manipulations, and the support for scalability are important features. The support for manipulations, especially the support for interventions, is crucial for decision making. In method development, the role of data collection mechanisms becomes central. Our example comprehensively demonstrates all features of the simulation framework.

In future work, we would like to apply the presented framework to real-world scenarios that are similar to our example in Section 5, where decision making involving healthcare policies and treatment interventions is required. A key challenge for the use of simulations in medical decision making is related to causality. Information on the causal effects of risk factors is needed to simulate the effect of interventions realistically. By default, risk prediction models based on observational data are not causal models. This holds also for the FINRISK and the FINDRISC risk score models defined in Section 5.1. The advances in causal inference and epidemiology are expected to improve the possibilities to conduct increasingly realistic simulations in the future.

Acknowledgements

This work belongs to the thematic research area "Decision analytics utilizing causal models and multiobjective optimization" (DEMO) supported by Academy of Finland (grant number 311877).

Funding

No specific funding for this article is reported.

Conflict of Interest

No competing interests reported.

Data and code availability

Sima source code, code for the examples, and data used for simulations are available via the Sima GitHub repository https://github.com/santikka/Sima and the package website https://santikka.github.io/Sima/. Please contact the authors for further information.

REFERENCES

Aalen O, Røysland K, Gran JM, Kouyos R, Lange T. 2016. Can we believe the DAGs? A comment on the relationship between causal dags and mechanisms. *Statistical Methods in Medical Research* **25**: 2294–2314. DOI: https://doi.org/10.1177/0962280213520436

Ahmad S, Billimek J. 2007. Limiting youth access to tobacco: Comparing the long-term health impacts of increasing cigarette excise taxes and raising the legal smoking age to 21 in the United States. *Health Policy* **80**: 378–391. DOI: https://doi.org/10.1016/j.healthpol.2006.04.001

Albert JM, Li Y, Sun J, Woyczynski WA, Nelson S. 2019. Continuous-time causal mediation analysis. *Stat in Med* **38**: 4334–4347.

Blackman D, Vigna S. 2021. Scrambled Linear Pseudorandom Number Generators. *ACM Transactions on Mathematical Software* **47**: 1–32. DOI: https://doi.org/10.1145/3460772

Box GE, Cox DR. 1964. An analysis of transformations. *Journal of the Royal Statistical Society. Series B* **26**: 211–243.

Brønnum-Hansen H, Davidsen M, Thorvaldsen P. 2001. Long-term survival and causes of death after stroke. *Stroke* **32**: 2131–2136. DOI: https://doi.org/10.1161/hs0901.094253

Caro JJ, Huybrechts KF, Xenakis JG, O'Brien JA, Rajagopalan K, Lee K. 2006. Budgetary impact of treating acute bipolar mania in hospitalized patients with quetiapine: an economic analysis of clinical trials. *Current Medical Research and Opinion* **22**: 2233–2242. DOI: https://doi.org/10.1185/030079906X148265

Chang W. 2019. R6: Encapsulated Classes with Reference Semantics. R package version 2.4.1. https://CRAN.R-project.org/package=R6

Clements M, Jauhiainen A, Karlsson A, Carzaniga C, L'Ecuyer P. 2021. Microsimulation: Discrete Event Simulation in R and C++, with Tools for Cost-Effectiveness Analysis. R package version 1.3.4. https://CRAN.R-project.org/package=microsimulation

Cooper K, Davies R, Roderick P, Chase D, Raftery J. 2002. The development of a simulation model of the treatment of coronary heart disease. *Health Care Management Science* **5**: 259–267. DOI: https://doi.org/10.1023/A:1020378022303

De Menten G, Dekkers G, Bryon G, Liégeois P, O'Donoghue C. 2014. LIAM2: a new open source development tool for discrete-time dynamic microsimulation models. *Journal of Artificial Societies and Social Simulation: JASSS* **17**: 9. DOI: https://doi.org/10.18564/jasss.2574

Dowle M, Srinivasan A. 2019. data.table: Extension of 'data.frame. R package version 1.12.4. https://CRAN.R-project.org/package=data.table

Eastman RC, Javitt JC, Herman WH, Dasbach EJ, Zbrozek AS, Dong F, Manninen D, Garfield SA, Copley-Merriman C, Maier W. 1997. Model of complications of NIDDM: I. model construction and assumptions. *Diabetes Care* **20**: 725–734.

Eldabi T, Paul R, Taylor S. 2000. Simulating economic factors in adjuvant breast cancer treatment. *The Journal of the Operational Research Society* **51**: 465–475. DOI: https://doi.org/10.1057/palgrave.jors.2600881

Enns EA, Cipriano LE, Simons CT, Kong CY. 2015. Identifying best-fitting inputs in health-economic model calibration: a pareto frontier approach. *Medical Decision Making: An International Journal of the Society for Medical Decision Making* **35**: 170–182.

Harding A. 2007. Challenges and opportunities of dynamic microsimulation modelling. *In 1st General Conference of the International Microsimulation Association*. Vienna.

Härkänen T, Karvanen J, Tolonen H, Lehtonen R, Djerf K, Juntunen T, Koskinen S. 2016. Systematic handling of missing data in complex study designs–experiences from the Health 2000 and 2011 surveys. *Journal of Applied Statistics* **43**: 2772–2790.

He FJ, Li J, MacGregor GA. 2013. Effect of longer term modest salt reduction on blood pressure: Cochrane systematic review and meta-analysis of randomised trials. *BMJ* **346**: f1325.

Hennessy DA, Flanagan WM, Tanuseputro P, Bennett C, Tuna M, Kopec J, Wolfson MC, Manuel DG. 2015. The population health model (POHEM): an overview of rationale, methods and applications. *Population Health Metrics* **13**: 24. DOI: https://doi.org/10.1186/s12963-015-0057-x

Herlihy M, Shavit N. 2011. The Art of Multiprocessor Programming. Elsevier.

Huang ES, Zhang Q, Brown SE, Drum ML, Meltzer DO, Chin MH. 2007. The cost-effectiveness of improving diabetes care in US federally qualified community health centers. *Health Services Research* **42**: 2174–2193.

Karvanen J. 2015. Study design in causal models. *Scandinavian Journal of Statistics* **42**: 361–377. DOI: https://doi.org/10.1111/sjos.12110

Karvanen J, Tikka S, Hyttinen A. 2021. Do-search: A tool for causal inference and study design with multiple data sources. *Epidemiology* **32**: 111–119.

Katsaliaki K, Mustafee N. 2011. Applications of simulation within the healthcare context. *The Journal of the Operational Research Society* **62**: 1431–1451.

Koponen P, Borodulin K, Lundqvist A, Sääksjärvi K, Koskinen S. 2018. Terveys, toimintakyky ja hyvinvointi Suomessa – FinTerveys 2017-tutkimus (Health, performance and welfare in Finland – FinTerveys 2017 study). Report 4/2018. Helsinki: National Institute for Health and Welfare.

Kosar B, Tomintz M. 2014. simSalud: A web-based spatial microsimulation to model the health status for small areas using the example of smokers in Austria. *IGI Forum*. Vol. 2 p. 207–216.

Krijkamp EM, Alarid-Escudero F, Enns EA, Jalal HJ, Hunink MM, Pechlivanoglou P. 2018. Microsimulation modeling for health decision sciences using R: a tutorial. *Medical Decision Making: An International Journal of the Society for Medical Decision Making* **38**: 400–422. DOI: https://doi.org/10.1177/0272989X18754513

Kuchenbecker U, Chase D, Reichert A, Schiffner-Rohe J, Atwood M. 2018. Estimating the cost-effectiveness of a sequential pneumococcal vaccination program for adults in Germany. *PloS One* **13**: 1–23. DOI: https://doi.org/10.1371/journal.pone.0197905

Larsson S, Prioux M, Fasth T, Ternhag A, Struwe J, Dohnhammar U, Brouwers L. 2018. A microsimulation model projecting the health care costs for resistance to antibacterial drugs in Sweden. *European Journal of Public Health* **29**: 392–396. DOI: https://doi.org/10.1093/eurpub/cky209

Li J, O'Donoghue C. 2014. Evaluating binary alignment methods in microsimulation models. *Journal of Artificial Societies and Social Simulation: JASSS* **17**: 15.

Lindström J, Tuomilehto J. 2003. The diabetes risk score: a practical tool to predict type 2 diabetes risk. *Diabetes Care* **23**: 725–731.

Little RJA, Rubin DB. 2002. Statistical Analysis with Missing Data. Hoboken, New Jersey: Wiley.

Mannion O, Lay-Yee R, Wrapson W, Davis P, Pearson J. 2012. Jamsim: a microsimulation modelling policy tool. *Journal of Artificial Societies and Social Simulation: JASSS* **15**: 8. DOI: https://doi.org/10.18564/jasss.1902

Manuel DG, Tuna M, Hennessy D, Bennett C, Okhmatovskaia A, Finès P, Tanuseputro P, Tu JV, Flanagan W. 2014. S. T. for Applied Research Team, et al. Projections of preventable risks for cardiovascular disease in Canada to 2021: a microsimulation modelling approach. *CMAJ Open* **2**: 94–101.

Manuel DG, Garner R, Finès P, Bancej C, Flanagan W, Tu K, Reimer K, Chambers LW, Bernier J. 2016. Alzheimer's and other dementias in Canada, 2011 to 2031: a microsimulation population health modeling (POHEM) study of projected prevalence, health burden, health services, and caregiving use. *Population Health Metrics* **14**: 37. DOI: https://doi.org/10.1186/s12963-016-0107-z

Marsaglia G, Tsang WW. 2000. The ziggurat method for generating random variables. *Journal of Statistical Software* **5**: 1–7. DOI: https://doi.org/10.18637/jss.v005.i08

Matsumoto M, Nishimura T. 1998. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation* **8**: 3–30. DOI: https://doi.org/10.1145/272991.272995

Microsoft and S. Weston. 2019. doParallel: Foreach Parallel Adaptor for the 'parallel' package. R package version 1.0.15. https://CRAN.R-project.org/package=doParallel

Microsoft and S. Weston. 2020. foreach: Provides Foreach Looping Construct. R package version 1.5.0. https://CRAN.R-project.org/package=foreach

Mielczarek B. 2016. Review of modelling approaches for healthcare simulation. *Operations Research and Decisions* **26**: 1. DOI: https://doi.org/10.5277/ord160104

Mustafee N, Katsaliaki K, Taylor SJ. 2010. Profiling literature in healthcare simulation. *Simulation* **86**: 543–558. DOI: https://doi.org/10.1177/0037549709359090

Nelder JA, Mead R. 1965. A simplex method for function minimization. *The Computer Journal* **7**: 308–313. DOI: https://doi.org/10.1093/comjnl/7.4.308

Official Statistics of Finland (OSF). 2017. Deaths (e-publication). Statistics Finland, Helsinki (Accessed 7.4.202). http://www.stat.fi/til/kuol/index_en.html

Official Statistics of Finland (OSF). 2017. Population structure (e-publication). Statistics Finland, Helsinki (Accessed: 7.4.2020). http://www.stat.fi/til/vaerak/index_en.html

O'Neill ME. 2014. PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation. Technical Report HMC-CS-2014-0905. Claremont, CA: Harvey Mudd College.

Pearl J. 2009. Causality: Models, Reasoning, and Inference. 2nd edn. Cambridge University Press.

Prakash MK, Lang B, Heinrich H, Valli PV, Bauerfeind P, Sonnenberg A, Beerenwinkel N, Misselwitz B. 2017. CMOST: an open-source framework for the microsimulation of colorectal cancer screening strategies. *BMC Med. Inform. Decis* **17**: 80. DOI: https://doi.org/10.1186/s12911-017-0458-9

R Core Team. 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Richiardi MG, Richardson RE. 2017. JAS-mine: A new platform for microsimulation and agent-based modelling. *International Journal of Microsimulation* **10**: 106–134. DOI: https://doi.org/10.34196/ijm.00151

Sai A, Vivas-Valencia C, Imperiale TF, Kong N. 2019. Multiobjective calibration of disease simulation models using gaussian processes. *Medical Decision Making: An International Journal of the Society for Medical Decision Making* **39**: 540–552.

Salleh S, Thokala P, Brennan A, Hughes R, Booth A. 2017. Simulation modelling in healthcare: An umbrella review of systematic literature reviews. *PharmacoEconomics* **35**: 937–949. DOI: https://doi.org/10.1007/s40273-017-0523-3

Salmon JK, Moraes MA, Dror RO, Shaw DE. 2011. Parallel random numbers: As easy as 1, 2, 3. *In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. New York, USA: Association for Computing Machinery. DOI: https://doi.org/10.1145/2063384.2063405

Spielauer M, Dupriez O. 2019. A portable dynamic microsimulation model for population, education and health applications in developing countries. *International Journal of Microsimulation* **12**: 6–27. DOI: https://doi.org/10.34196/ijm.00205

Spielauer M. 2007. Dynamic microsimulation of health care demand, health care finance and the economic impact of health behaviours: survey and review. *International Journal of Microsimulation* **1**: 35–53. DOI: https://doi.org/10.34196/ijm.00005

Stephensen P. 2015. Logit scaling: A general method for alignment in microsimulation models. *International Journal of Microsimulation* **9**: 89–102. DOI: https://doi.org/10.34196/IJM.00144

Strohmaier S, Røysland K, Hoff R, Borgan Ø, Pedersen TR, Aalen OO. 2015. Dynamic path analysis–a useful tool to investigate mediation processes in clinical survival trials. *Statistics in Medicine* **34**: 3866–3887. DOI: https://doi.org/10.1002/sim.6598

Stubner R. 2019. dqrng: Fast Pseudo Random Number Generators. R package version 0.2.1. https://CRAN.R-project.org/package=dqrng

Tomintz M, Kosar B, Garcia-Barrios V. 2017. simSALUD: Design and implementation of an open-source wizard based spatial microsimulation framework. *International Journal of Microsimulation* **10**: 118–143. DOI: https://doi.org/10.34196/ijm.00159

Tunstall-Pedoe H, Kuulasmaa K, Tolonen H, Davidson M, Mendis S. 2003. MONICA Monograph and Multimedia Sourcebook. Geneva: World Health Organization. https://www.thl.fi/publications/monica/monograph_cd/contents.htm

Van Buuren S. 2018. Flexible Imputation of Missing Data. 2nd edn. Boca Raton, Florida: CRC Press.

Vartiainen E, Laatikainen T, Peltonen M, Puska P. 2016. Predicting coronary heart disease and stroke: The FINRISK calculator. *Global Heart* **11**: 213–216. DOI: https://doi.org/10.1016/j.gheart.2016.04.007

Weston S. 2017. doMPI: Foreach Parallel Adaptor for the Rmpi Package. R package version 0.2.2. https://CRAN.R-project.org/package=doMPI

Weycker D, Sato R, Strutton D, Edelsberg J, Atwood M, Jackson LA. 2012. Public health and economic impact of 13-valent pneumococcal conjugate vaccine in US adults aged 50 years. *Vaccine* **30**: 5437–5444. DOI: https://doi.org/10.1016/j.vaccine.2012.05.076

WHO. 2003. Joint WHO/FAO expert consultation on diet, nutrition and the prevention of chronic diseases, volume 916 of WHO Technical Report Series. Geneva: WHO.

Wohlin C. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* 1–10. DOI: https://doi.org/10.1145/2601248.2601268

Wolfson MC. 1994. POHEM: a framework for understanding and modelling the health of human populations. *World Health Statistics Quarterly. Rapport Trimestriel de Statistiques Sanitaires Mondiales* **47**: 157–176.

Yu H. 2002. Rmpi: Parallel statistical computing in R. *R News* **2**: 10–14.

Zur RM, Zaric GS. 2016. A microsimulation cost-utility analysis of alcohol screening and brief intervention to reduce heavy alcohol consumption in Canada. *Addiction* **111**: 817–831. DOI: https://doi.org/10.1111/add.13201

Appendix A Related Work in the Health Domain

There exist a large number of review papers considering the usage of simulation in the healthcare domain. An umbrella review by ***Salleh et al. (2017)*** summarizes 37 surveys and divides the original works into four categories: *types of applications*, *simulation techniques used*, *data sources*, and *software used for simulation modelling*. According to this classification, our work falls under *medical decision making* when considering the application domain. A considerable amount of literature has been published on simulation for medical decision making applications. ***Table A1*** shows a selection of these studies, identified by going through the references of those reviews, that according to the umbrella review by ***Salleh et al. (2017)***, reviewed the most medical decision making application studies (***Katsaliaki and Mustafee, 2011***; ***Mielczarek, 2016***; ***Mustafee et al., 2010***). As summarized in the table, the models developed for this application area are typically not open-source. Moreover, in comparison to our work, most of these related studies dealt with only one particular decision making application for one particular country.

Table A1. Selection of related work on using simulation for medical decision making applications. Typically, models are not made open-source and—with the exception of DYNAMIS-POP—developed only for one particular country and decision making problem

Authors	Simulation Model, Objective	Software, Tools, Open-source Availability
Eldabi et al. (2000)	Discrete event simulation to simulate economic factors in adjuvant breast cancer treatment in England.	A package called ABCSim was created using the commercial simulation software Simul8. ABCSim is not open-source.
Cooper et al. (2002)	Discrete event simulation to model the progress of English patients who have had a coronary event through their treatment pathways and subsequent coronary events.	The simulation was written using the POST (Patient Oriented Simulation Technique) software with a Delphi interface. The developed model is not open-source.
Caro et al. (2006)	Discrete event simulation to simulate the course of individuals with acute mania in bipolar I disorder and estimation of the budget impact of treatments for them from a United States healthcare payer perspective.	The model is not open-source and the article does not provide any information on the used software.
Ahmad and Billimek (2007)	A 75-year dynamic simulation model comparing the long-term health benefits to society of various levels of tax increase to a viable alternative: limiting youth access to cigarettes by raising the legal purchase age to 21 in the United States.	Vensim (i.e., commercial software) was used to develop a dynamic simulation model for estimating the population health outcomes resulting from raising taxes on cigarettes and raising the legal smoking age to 21. The developed model is not open-source.
Huang et al. (2007)	A Monte Carlo simulation model to estimate the cost-effectiveness of improving diabetes care in federally qualified community health centers in the United States.	Microsoft Excel 2000 and @Risk 4.5.4 for Windows were used to conduct the simulations and an older diabetes complications model (Eastman et al., 1997) was adapted. The model is not open-source but the inputs are described in the article.
Zur and Zaric (2016)	Microsimulation model of alcohol consumption and its effects on alcohol-related causes of death in the Canadian population. Estimation of the cost-effectiveness of implementing universal alcohol screening and brief intervention in primary care in Canada.	The model was programmed and run in C and the results analyzed in R version 3.1.2. The programmed model is not open-source.
Larsson et al. (2018)	Individual-based microsimulation model to project economic consequences of resistance to antibacterial drugs for the Swedish health care sector.	A dynamic microsimulation model developed by the Swedish Ministry of Finance called SESIM was used. Currently (May 2021), it is not possible to download SESIM.

Authors	Simulation Model, Objective	Software, Tools, Open-source Availability
Prakash et al. (2017); Sai et al. (2019)	CMOST is a microsimulation model for modeling the natural history of colorectal cancer, simulating the effects of colorectal cancer screening interventions, and calculating the resulting costs.	CMOST (Colon Modeling Open Simulation Tool) was implemented in Matlab and is freely available under the GNU General Public License at https://gitlab.com/misselwb/CMOST.
Kuchenbecker et al. (2018)	The authors adapted an existing microsimulation US model (Weycker et al., 2012) to depict lifetime risks and costs of invasive pneumococcal diseases and nonbacteremic pneumonia, as well as the expected impact of different vaccination schemes, in a hypothetical population of German adults.	No link to a documentation of the model or open-source framework can be found in the articles (Weycker et al., 2012; Kuchenbecker et al., 2018).
Spielauer and Dupriez (2019)	DYNAMIS-POP is a dynamic micro-simulation model for population and education projections, health applications, and for the simulation of policies. Adaptation of the model to a specific country only requires adapting a single setup script and simulation module.	All components of DYNAMIS-POP including its code and all statistical analysis files are freely available and documented online at http://dynamis.ihsn.org/. Most statistical analysis scripts and scripts for post-processing and visualization of the results are implemented in R.
Hennessy et al. (2015); Manuel et al. (2014); Manuel et al. (2016); Wolfson (1994)	POHEM (POpulation HEalth Model) is a longitudinal microsimulation model of health and disease. The model simulates representative populations and allows the rational comparison of competing health intervention alternatives, in a framework that captures the effects of disease interactions.	POHEM was developed in Modgen (see ***Table A2***). However, the POHEM model implementation itself is not open-source (and so far still limited to Canada, i.e., one country). https://www.statcan.gc.ca/eng/microsimulation/modgen/new/mods/pohem.

Table A2. Related work on open-source frameworks for microsimulation models

Authors	Simulation Model/ Objective	Link / Open-source Availability
De Menten et al. (2014)	LIAM2 is an open-source development tool for discrete-time dynamic microsimulation models. The framework is for general purposes (not specifically health related). It allows for the simulation of discrete-time dynamic models with a cross-sectional time step of whatever kind of objects the modeler chooses. It provides functionalities to calibrate to exogenous information of any number of dimensions.	LIAM2 was developed primarily in Python, it can be downloaded from http://liam2.plan.be. It is licensed under the GNU General Public License, meaning one can freely use, copy, modify and redistribute the software.
Mannion et al. (2012)	JAMSIM (JAva MicroSIMulation) is a synthesis of open-source packages that provides an environment and set of features for the creation of dynamic discrete-time microsimulation models that are to be executed, manipulated and interrogated by nontechnical, policy-oriented users.	JAMSIM is freely available as an open-source tool, for public reuse and modification at http://code.google.com/p/jamsim/. It combines R and the Java-based agent-based modeling graphical tool Ascape.
Richiardi and Richardson (2017)	JAS-mine is a Java-based computational platform that features tools for discrete-event simulations encompassing both dynamic microsimulation and agent-based modeling. Object-relational mapping is used to embed a relational database management system. It is a general-purpose platform (not specifically health related).	JAS-mine is freely available as an open-source tool, for public reuse and modification at https://github.com/jasmineRepo/JAS-mine-core. It is written in Java.
Kosar and Tomintz (2014); Tomintz et al. (2017)	simSALUD is a deterministic spatial microsimulation framework. Its main aim is to model health related issues for small areas using spatial microsimulation modeling to simulate information where no data exists or is accessible.	simSALUD is an open-source web-application, programmed using Java and based on Apache's model-view-controller application Struts2. It can be accessed at http://www.simsalud.org/simulation/.

Authors	Simulation Model/ Objective	Link / Open-source Availability
Clements et al. (2021)	microsimulation package for R provides implementations of discrete event simulations in both R and C++. The R implementation builds on several R5 classes (useful from a pedagogical perspective, but slow for larger microsimulations). For speed, the authors provide C++ classes.	The package is freely available at https://github.com/mclements/microsimulation.
Developed by Statistics Canada; used, for example, in Spielauer and Dupriez (2019)	Modgen and its open-source implementation openM++ are generic microsimulation software supporting the creation, maintenance and documentation of dynamic microsimulation models. Several types of models can be accommodated, be they continuous or discrete time, with interacting or noninteracting populations. Compared to its closed source predecessor Modgen, openM++ has many advantages like portability, scalability and open source.	openM++ is freely available at https://github.com/openmpp.

At the same time, there is a relatively small body of literature that is concerned with open-source microsimulation frameworks within the medical domain (a summary of this literature is provided in **Table A2**). In fact, **Spielauer (2007)** pointed out that microsimulation techniques largely focus on simulating tax-benefit and pension systems, and that extensions for health-related policies and interventions are sparse. Although this situation has changed since 2007, the related work on open-source frameworks for this domain is still limited. For example, a Scopus search in May 2020 with the TITLE-ABS-KEY search terms (open AND source AND framework AND microsimulation AND health) yielded only two results (**Prakash et al., 2017**; **Kuchenbecker et al., 2018**). Similarly, a PubMed search with the same keywords also returned only two studies (**Prakash et al., 2017**; **Sai et al., 2019**). As documented in **Table A1**, **Prakash et al. (2017)** developed and used a specific open-source tool for colorectal cancer microsimulation that was later used by **Sai et al. (2019)**, while **Kuchenbecker et al. (2018)** used an existing model to study pneumococcal diseases. In sum, these studies are limited to specific health problems and only **Prakash et al. (2017)** developed an open-source tool. The remaining studies in **Table A2** were identified with a Google Scholar search with the same keywords specified above and by using the snowballing technique (**Wohlin, 2014**) among all already identified articles.

The first three frameworks listed in **Table A2** are all for general purposes (not specifically health related) open-source simulation frameworks that—like our framework—support large-scale individual-level data generation. The next framework in the table, simSALUD, is meant to simulate only small area health related issues. The last rows of the table describe the closest existing tools to our framework. The microsimulation package in R and openM++ (both based on C++) can be used from R. However, they do not include direct support for manipulations.

As summarized by **Krijkamp et al. (2018)**, R is appropriate not only for developing microsimulation models but also for the open-source distribution of the implementations. They emphasize that despite these advantages, open source models for medical decision making in R are scarce. To encourage more developers to use R, they provide an easy microsimulation example of a medical decision model (i.e., the sick-sicker model, first introduced by **Enns et al. (2015)**). We compared the performance of Sima with their implementation. Running their vectorized example (by sourcing directly from GitHub and using the authors' own measure of running time) took ≈ 6 seconds and running the Sima version took ≈ 2 seconds if we record the entire simulation history (our implementation of the sick-sicker model is available at the Sima website https://santikka.github.io/Sima/). Thus, we conclude that to our knowledge, no open-source simulation framework in R for realistic large-scale individual-level data generation for medical decision making was developed so far that is as scalable as Sima.

Appendix B Mathematical Details

We present here a mathematical formulation of our simulation framework. The implementation of Sima builds on these concepts and definitions.

Simulation domain defines the scope of the status variables, parameters and latent variables.

Definition 1 (Simulation Domain)

A *simulation domain* is a triple

$$(\mathcal{X}, \Theta, \Psi),$$

where $\mathcal{X} =_{j=1}^{m} \mathcal{X}_j$ is the space of the status variables, Θ is the parameter space and Ψ is the latent variable space.

We assume that the spaces $\mathcal{X}_j$ of individual status variables are one dimensional and real-valued. This assumption does not rule out for example categorical variables, which can be included by encoding them in the standard way with a binary indicator for each category. It should be noted we do not impose any restrictions on the possible combinations of values in the Cartesian product $\mathcal{X}$.

Definition 2 (Population)

Let $\mathcal{D} = (\mathcal{X}, \Theta, \Psi)$ be a simulation domain. A *population* $\mathbf{X}$ in $\mathcal{D}$ is a $n \times m$ matrix

$$\mathrm{X} = \begin{bmatrix} \boldsymbol{x}_1^{\mathsf{T}} \\ \vdots \\ \boldsymbol{x}_n^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix},$$

where each row vector $\boldsymbol{x}_i^{\mathsf{T}}$ refers to the *individual* of the population, n is the population size, and $x_{ij} \in \mathcal{X}_j$ for all $i = 1, \ldots, n$, $j = 1, \ldots, m$. The element x_{ij} is called the *status variable* j of individual n.

The set of populations of size n having m status variables in $\mathcal{D}$ is denoted by $\mathcal{P}^{n \times m}[\mathcal{D}]$. Changes in the population are carried by the event functions.

Definition 3 (Manipulation Event)

Let $\mathcal{D} = (\mathcal{X}, \Theta, \Psi)$ be a simulation domain. A *manipulation event* μ in $\mathcal{D}$ is a function $\mu : \mathcal{X} \times \Theta \times \Psi \to \mathcal{X}$.

Manipulation events are used to describe deterministic changes between two time points in the status variables of a single individual $\boldsymbol{x}_i^{\mathsf{T}}$ of a population $\mathbf{X}$. We also define the following.

Definition 4 (Accumulation Event)

Let $\mathcal{D} = (\mathcal{X}, \Theta, \Psi)$ be a simulation domain. An *accumulation event* in $\mathcal{D}$ is a pair

(η, ν), where η is a function $\eta : \Theta \times \Psi \to \mathbb{N}$ and ν is a sequence $\{b_k\}_{k=1}^{\infty}$ of functions$b_k : \Theta \times \Psi \to \mathcal{P}^{k \times m}[\mathcal{D}]$.

The operation of accumulation events is twofold. First, the function η selects the number of individuals k to be added to the population. Second, the corresponding function b_k is used to generate the new set of individuals to be added. The case where $\eta(\boldsymbol{\theta}, \psi) = 0$ for some assignment of $\boldsymbol{\theta} \in \Theta$ and $\psi \in \Psi$ should be understood to mean that no new individuals will be added to the population. The events are defined as deterministic functions given the parameter $\boldsymbol{\theta}$ and the latent variables ψ which, if assumed to be random variables, give a stochastic nature to the simulator. Event order is defined as follows.

Definition 5 (Event Order)

Let $\mathcal{D} = (\mathcal{X}, \Theta, \Psi)$ be a simulation domain and let $k \in \mathbb{N}$. An *event order* in $\mathcal{D}$ of k manipulation events is a set of functions $\{\sigma_1, \ldots, \sigma_n\}$, such that

$$\sigma_n : \mathbb{N}_k \times \Psi \to \mathbb{N}_k, \qquad \text{for all } n \in \mathbb{N},$$

$$\mathbb{N}_k = \{1, \ldots, k\}\text{where}$$

A single function σ_j is essentially a permutation function that defines the application order of manipulation events for individual j, which can be random, as these functions depend on the latent variables Ψ. We are now ready to define the simulator itself.

Definition 6 (Simulator)

A *simulator* is a quintuple

$(\mathcal{D}, \mathbf{X}_0, \{\mu_i\}_{i=1}^r, E_r, A)$, where $\mathcal{D}$ is a simulation domain, $\mathbf{X}_0$ is the initial population in $\mathcal{D}$, $\{\mu_i\}_{i=1}^r$ is a sequence of manipulation events in $\mathcal{D}$, E_r is an event order in $\mathcal{D}$ of r manipulation events and A is a set of accumulation events in $\mathcal{D}$.

At a given time, the simulator occupies a state.

Definition 7 (State).

Let

$$\mathcal{S} = ((\mathcal{X}, \Theta, \Psi), \mathbf{X}_0, \{\mu_i\}_{i=1}^r, E_r, A)$$

be a simulator. A *state* of $\mathcal{S}$ is a pair

$(\mathbf{X},\ \boldsymbol{\theta})$,where $\mathbf{X}$ is a population in $(\mathcal{X}, \Theta, \Psi)$ and $\boldsymbol{\theta} \in \Theta$. A state of $\mathcal{S}$ is its initial state if $\mathbf{X} = \mathbf{X}_0$. The set of all states of $\mathcal{S}$ is denoted$\Sigma[\mathcal{S}]$.

A state of the simulator encapsulates the status of its population ($\mathbf{X}$) and its parameter configuration ($\boldsymbol{\theta}$). Starting from an initial state, transitions between states are carried out by the transition function.

Definition 8 (Transition Function)

Let

$$\mathcal{S} = ((\mathcal{X}, \Theta, \Psi), \mathbf{X}_0, \{\mu_i\}_{i=1}^r, E_r, A)$$

be a simulator. The *transition function* δ of $\mathcal{S}$ is a function $\delta : \Sigma[\mathcal{S}] \times \Psi \to \Sigma[\mathcal{S}]$ such that

$$\delta(S, \psi) = \delta((\mathbf{X}, \boldsymbol{\theta}), \psi) = (d(\mathbf{X}, \boldsymbol{\theta}, \psi), \boldsymbol{\theta}) = (\mathbf{X}', \boldsymbol{\theta}),$$

$S \in \Sigma[\mathcal{S}]$where and is defined as follows

where $(\alpha_j, \{b_{j,k}\}_{k=1}^{\infty}) \in A$, $k(j) = \alpha_j(\boldsymbol{\theta}, \boldsymbol{\psi})$, $\{\ell_1, \dots, \ell_s\} = \{j \,|\, j = 1, \dots, |A|, k(j) > 0\}$ and $\sigma_j \in E_r$. The symbol $\bigcirc_{i=1}^{m}$ denotes an ordered composition of a sequence of functions $\{f_i\}_{i=1}^{m}$ over the leftmost argument, i.e., $\bigcirc_{i=1}^{m} f_i = f_1 \circ f_2 \circ \cdots \circ f_m$.

Given a latent vector $\boldsymbol{\psi}$, the transition function defines the transformation from one state to the next. The functions d_m and d_a correspond to the transformations carried out by manipulation events and the accumulation events. Since accumulation events do not depend on the current population, the order in which they are applied does not matter in d_a. However, the order is significant for manipulation events in d_m. Furthermore, this order can vary between individuals if the elements of the event order are defined as $\sigma_j(i, \boldsymbol{\psi})$ instead of just $\sigma(i, \boldsymbol{\psi})$, which would indicate a shared order for all individuals instead.

A single run of the simulator provides the user with a simulation sequence.

Definition 9 (Simulation Sequence)

Let

$$\mathcal{S} = ((\mathcal{X}, \Theta, \Psi), \mathbf{X}_0, \{\mu_i\}_{i=1}^{r}, E_r, A)$$

be a simulator, $S_0 \in \Sigma[\mathcal{S}]$, and let $\{\boldsymbol{\psi}_t\}_{t=0}^{T}$ be a sequence of $(\Psi, \mathcal{F})$-measurable random vectors. A simulation sequence of $\mathcal{S}$ is a sequence of its states $\{S_t\}_{t=0}^{T}$ such that

$$\mathrm{S}_t = \begin{cases} S_0 & if t=0, \\ \delta(S_{t-1}, \boldsymbol{\psi}_t) & if t>0, \end{cases}$$

where δ is the transition function of $\mathcal{S}$.

The index T defines the length of the simulation sequence. In practice, T can be viewed as a length of a specific time interval and incrementation of time t by 1 can be understood to mean the progress of time at the desired granularity (in days, weeks, years, etc.). Note that in a single simulation sequence the parameters $\boldsymbol{\theta}$ are determined by the state at time $t = 0$ and remain unchanged throughout the sequence.

In order to calibrate the simulator, we need to be able to change its parameter configuration while keeping the system otherwise intact. This feature is provided by the configuration function.

Definition 10 (Configuration Function)

Let

$$\mathcal{S} = ((\mathcal{X}, \Theta, \Psi), \mathbf{X}_0, \{\mu_i\}_{i=1}^{r}, E_r, A)$$

be a simulator. The *configuration function* c of $\mathcal{S}$ is a function $c : \Sigma[\mathcal{S}] \times \Theta \to \Sigma[\mathcal{S}]$ such that

$$c(S, \boldsymbol{\theta}') = c((\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\theta}') = (\mathbf{X}, \boldsymbol{\theta}'),\text{where}$$

As the name suggests, the configuration function c can be used to configure the simulator, meaning that it can be used to manipulate the current state by exchanging the parameter vector $\boldsymbol{\theta}$ for $\boldsymbol{\theta}'$. The data collection process of the simulator is provided by the sampler function.

Definition 11 (Sampler).

Let $\mathbf{X}$ be a population in $\mathcal{D}$. A *sampler* is a function $s : \Psi \times \mathcal{P}^{n \times m}[\mathcal{D}] \to \mathcal{P}^{n \times m}[\mathcal{D}^*]$ such that

$$s(\psi, \mathbf{X}) = \mathbf{X}^*,$$

where $\mathbf{X}^*$ is called a *sample* with either $x_{ij}^* = e(\psi, x_{ij})_j$ or $x_{ij}^* = \mathrm{NA}$ and $e_j : \Psi \times \mathcal{X}_j \to \mathcal{X}_j$ is an error function for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$. NA denotes a missing value and $\mathcal{D}^*$ denotes a simulation domain extended to include missing values.

Appendix C Efficient Pseudo-random Number Generation

The built-in pseudo-random number generators of R are fairly efficient, but they can be improved upon, and significant reductions in simulation time can be observed when the default generators are replaced by more advanced methods. The importance of fast and scalable PRNGs is further highlighted for large populations and long simulation runs. The R package dqrng (***Stubner, 2019***) provides several fast PRNG generators, including C++ implementations of the permuted congruential generator (PCG) family by ***O'Neill (2014)***, and Xoroshiro128+ and Xoshiro256+ by ***Blackman and Vigna (2021)***. The Ziggurat algorithm proposed by ***Marsaglia and Tsang (2000)*** is provided for generating random numbers from normal and exponential distributions. A version of the Threefry engine by ***Salmon et al. (2011)*** is also available. While cryptographic aspects of PRNGs may be of importance in some practical applications of the simulator, our focus is mainly in the performance.

Generators of the dqrng package are used internally in the implementation of the simulator, one example being the generation of the order in which manipulation events are applied at each time point. Typically, performance is improved over a base R generator even when using uniform random variables and the inverse method to generate realizations from a distribution for which no prespecified generator is available in the dqrng package. When manipulation events and accumulation events are defined for a specific simulator instance, the choice of PRNG for the random functionality of the events themselves can take advantage of the dqrng package but are not restricted to it. However, the user has to take care in order to ensure that the simulation remains reproducible if multiple different generators are used simultaneously.

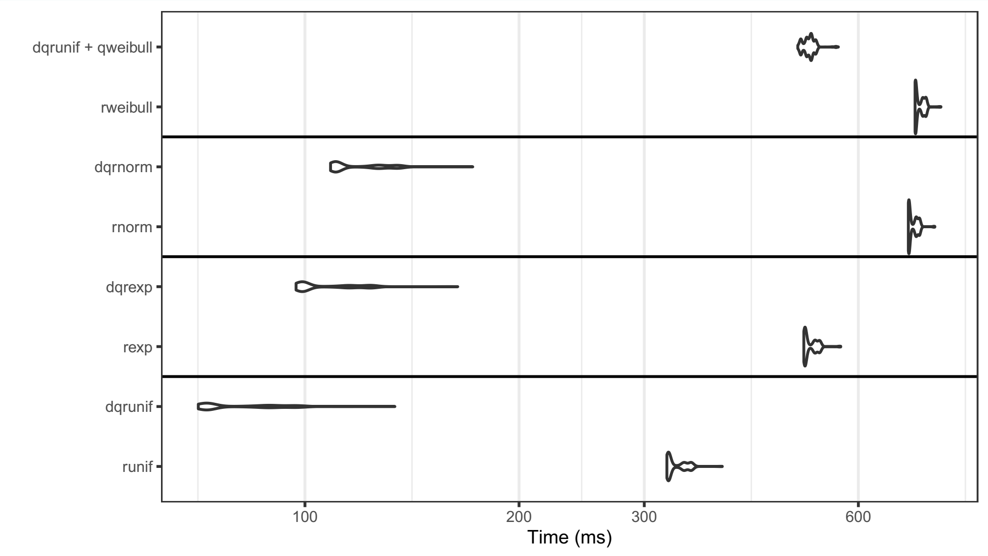

Figure C1. Density plots of time taken (in milliseconds) for 10 million random draws over 1000 replications for the Xoroshiro128+ PRNG implementation of dqrng and the 32 bit Mersenne-Twister of base R with Weibull, normal, exponential and uniform distributions. Prefix "dq" denotes the dqrng version of the function. The horizontal axis uses log10-scaling. Weibull-distributed random variables are generated using the inverse method.

We conduct a benchmark comparing Xoroshiro128+ provided by dqrng to the base R 32 bit Mersenne-Twister (***Matsumoto and Nishimura, 1998***). The distributions included in dqrng consist of exponential, gaussian and uniform distributions. In addition, we compare the generation of Weibull-distributed random variables using the inverse method and uniform random variables with dqrng against the base R rweibull function. The results are presented in Figure C1 and they show that dqrng dominates the default generators for the included distributions of the package. The difference is smaller when using the inverse method with the quantile function qweibull, but dqrng still always outperforms the base generators.