A Tutorial on Discrete Event Simulation Models in R using a Cost-Effectiveness Analysis Example

Mauricio Lopez-Mendez, PhD[1]

Jeremy D. Goldhaber-Fiebert, PhD[2]

Fernando Alarid-Escudero, PhD[†]

Discrete Event Simulation (DES) is a flexible and computationally efficient approach for modeling diverse processes; however, DES remains underutilized in healthcare and medical decision-making due to a lack of reliable and reproducible implementations. We developed an open-source DES framework in R to simulate individual-level state-transition models (iSTMs) in continuous time accounting for treatment effects, time dependence on state residence, and age-dependent mortality. Our DES implementation employs a modular and easily adaptable structure, with each module corresponding to a unique transition between health states. To simulate the evolution of the process (i.e., individual state-transitions), we adapted the next-reaction algorithm from the stochastic chemical reactions literature. Simulation-time dependence (age-dependent mortality) and state residence time dependence (transition from Sick to Sicker) are seamlessly incorporated into the DES framework via validated non-parametric and parametric sampling routines (e.g., inversion method) of event times. Treatment effects are integrated as scaling factors of the hazard functions (proportional hazards). We illustrate the framework's benefits by implementing the Sick-Sicker Model and conduct a cost-effectiveness analysis and probabilistic analysis. We also obtain epidemiological outcomes of interest from the DES output, such as disease prevalence, survival probabilities, and distributions of state-specific dwell times. Our DES framework offers a reliable and accessible alternative that enables the simulation of more realistic dynamics of state-transition processes at potentially lower implementation and computational costs than discrete-time iSTMs.

Keywords: discrete event simulation (DES), individual state-transition model (iSTM), Monte Carlo, decision models, cost-effectiveness analysis (CEA), Tutorial, R

---

[1] Department of Health Policy, Stanford School of Medicine, Stanford University, Stanford, CA, USA. mlopezme@stanford.edu

[2] Department of Health Policy, Stanford School of Medicine, Center for Health Policy, Freeman-Spogli Institute for International Studies, Stanford University, Stanford, CA, USA

Higlights:

- Discrete event simulation (DES) is a flexible and efficient approach to simulate diverse processes in model-based decision analysis.
- The tutorial presents an open-source DES framework in R to simulate individual-level State-Transition Models (iSTMs) in continuous time.
- The modular structure of our DES framework accommodates treatment effects, time-dependent transitions, and age-dependent mortality using validated sampling methods.
- The coded example uses the Sick-Sicker model to compute a cost-effectiveness analysis (CEA), epidemiological outcomes, probabilistic analysis (PA), and value of information analysis (VOI).

1. Introduction

Discrete event simulation (DES) is a general framework for simulating dynamic systems stochastically [1, 2, 3] . The salient feature is that the variables of interest are represented by "events" and the system's state is only updated when events occur. The DES framework can represent dynamic processes that evolve over continuous or discrete time, and can include multiple sources of within- and between-person variation[4], as well as different types of dependencies on time and previous states of the system (i.e., event history).

DES models apply to many situations as they are highly general. They are commonly used to analyze queuing systems, and more recently, they have been applied to model epidemiological and clinical phenomena, such as disease progression and transmission (e.g., [5, 6]. They are particularly valuable for modeling complex dynamics that can inform medical, pharmacological, and healthcare decision-making (e.g., [7, 8, 9]).

One advantage of DES is its computational and memory efficiency, even when representing systems involving large numbers of heterogeneous individuals facing complex dynamics. Discrete-time individual state-transition simulation models (iSTMs) (i.e., microsimulations) are frequently used to model systems with complex dynamics and heterogeneity. However, such models can require large amounts of computational power to generate sufficient simulated trajectories to make stable estimates of the key outcomes. This is because each trajectory in an iSTM requires evaluation of potential transitions at fixed time intervals, generating a pseudorandom number for each transition in each interval for each trajectory. However, the evolution of trajectories in a DES is "event-driven", meaning that evaluation is only necessary as events occur, greatly reducing the computation and memory requirements.

This tutorial aims to present a modular DES framework for model-based CEA and motivate its use with a healthcare example. Our framework emphasizes modeling principles that support a general implementation of DES that readers will find helpful for building their own implementation from first principles and without necessarily depending on specialized software. First, we provide a theoretical description of the DES framework. Second, we illustrate how to simulate time-dependent state-transition models (STM) using DES. Third, we conceptualize how to include treatment effects as well as covariates that can potentially be time-varying. DES yields equivalent outputs to those from cohort STMs and iSTMs that can be used in cost-effectiveness analysis (CEA) and health-technology assessments (HTAs). Finally, we provide a modular example in the R programming language [10] with the code to generate economic and epidemiological outcomes of interest in CEA and HTA, which can be adapted to more complex iSTMs following our DES framework. Readers will find that DES can be used to implement models in a straightforward manner similar to their prior experience with other techniques while enabling them to access its advantages without substantial additional programming efforts.

2. Theoretical Framework and Notation

A stochastic process is a collection of random variables, typically indexed by time. The domain of these random variables can be continuous or discrete [2].Mathematically, a DES corresponds to a continuous-time stochastic process over a discrete, finite event space. Compared to discrete-time STMs [11, 12, 13], continuous-time STMs require a different set of mathematical tools to analyze the simulated individuals' dynamics and behaviors.

2.1 Event-driven Dynamics in an STM

The evolution of a stochastic process with a discrete, finite state space can be described by the occurrence of "events" over time, defined by three components in an STM:

- An "event" is a change of state, i.e., a change in the discrete values of a variable X_t, and the set of all possible events is denoted as the "event-space". In the context of STMs, we can think of the set of unique transitions between states as the set of all possible events.
- The random variable T_n denotes the nth event time ($n = 1,2, \ldots < \infty$), i.e., the time at which the nth change of state occurs. An example of an event time is the time at which a patient transitions from healthy to sick.
- Inter-event time (i.e., dwell time), denotes the time spent in a given state before transitioning to another state, and is described mathematically as the time increment between consecutive events, $\tau_{n-1} = (T_n - T_{n-1})$.

Since $(T_n,\ T_{n-1})$ are random variables, the inter-event time, τ_{n-1}, is a random variable with an associated cumulative distribution function (CDF), $F(\tau)_{r,s}$, where (r) indexes the origin state (value of X_t immediately before the nth event) and (s) indexes the destination state (value of X_t after the nth event).

Figure 1 illustrates these concepts for a simple three-state iSTM for one simulated person that starts in state S_1 at time zero, T_0 and can transition between S_1 and S_2, and is also at risk of transitioning to the absorbing state S_3, i.e., once reached, individuals remain in S_3. Each transition is an event, and the four unique transitions in the state-transition diagram represent the event-space. A realization of a single trajectory of the 3-state model can be represented either as a sequence of states visited over time, or as a sequence of events occurring over time. In a DES, we focus on the latter, and keep track of the time points when the events occur. Consider the following realization of a single trajectory: $\{S_1, S_2, S_1, S_3\}$. Discrete-time STMs would evaluate the state a person occupies at all points on a finite grid of time points, requiring evaluations $X_{t=0}, X_{t=1}, \ldots, X_{t=k}; k < \infty$. Equivalently, we can encode the same information as a sequence of events, i.e., transitions between states: $(S_1 \rightarrow S_2,\ S_2 \rightarrow S_1,\ S_1 \rightarrow S_3)$. With this event-driven representation, we only need to keep track of the three times at which the events occur (T_1, T_2, T_3); for which we only need three evaluations $X_{t=T_1}, X_{t=T_2}, X_{t=T_3}$.

2.2 Intensity Functions and Cumulative Intensity Functions

A key concept to describe state-transition processes in continuous time is the notion of transition-specific intensity functions (also referred to as hazard functions in survival analysis)[14]. In the context of STMs, the intensity function describes the rate or instantaneous probability at which a person would transition from a state r, to a state s, $r \neq s$ [15, 16].

Let $\mathbb{S}$ denote the set of possible states that a person can visit throughout the simulation time-horizon, e.g., lifetime.

$$\mathbb{S} = [S_0,\ S_1,\ \ldots, S_n] \quad ; n < \infty$$

Let $X_t \in \mathbb{S}$, $t \leq 0$, be a collection of random variables denoting the state at time t that can only take values from the set $\mathbb{S}$. For example, consider an individual at time t who is in state $X_t = r,\ t \geq 0$, then, the intensity of transitioning to the next state, $X_{t+\delta} = s$, is represented by the intensity function $\lambda_{r,s}(t, \mathbf{z}(t), \mathcal{H}_t)$:

$$\lambda_{r,s}(t, \mathbf{z}(t), \mathcal{H}_t) = \lim_{\delta \to 0} \frac{\mathbb{P}[X_{t+\delta} = s | X_t = r, \mathbf{z}(t), \mathcal{H}_t]}{\delta} \qquad (1)$$

$$\text{for } r, s \in \mathbb{S},\ r \neq s.$$

Where $\lambda_{r,s}$ may depend on time t, covariates $\mathbf{z}(t)$, and the history of states visited before the time of the current event T_n, $\mathcal{H}_t = \{X_{t=0}, X_{t=T_1}, \ldots, X_{t=T_{n-1}}\}$. Additionally, let

$$\Lambda_{r,s}(t, \mathbf{z}(t), \mathcal{H}_t) = \int_0^t \lambda_{r,s}\,(u, \mathbf{z}(u), \mathcal{H}_u) du \qquad (2)$$

denote the cumulative intensity function that describes the accumulated rate of transition between state r and state s up to time t.

3. Evolution of a Continuous Time Process with Discrete and Finite Event-space (Simulating Sequences of Events)

3.1 Inter-event Time Sampling

If the process of transitioning between state r to state s follows a constant intensity function, the CDF of inter-event times $F(\tau)_{r,s}$ follows an exponential distribution. This corresponds to a homogeneous Poisson point process (HPPP) and has an analytic solution. However, if the intensity function depends on time, e.g., the time elapsed in state r, the distribution $F(\tau)_{r,s}$ could correspond to a non-homogeneous Poisson point process (NHPPP), which does not necessarily have a known analytic solution.

To simulate state-transition processes with an event-driven approach, validated routines are recommended for sampling the inter-event times, especially when the intensity function depends on time. In our DES implementation, we provide submodules for sampling inter-

event times following either the inversion method(e.g., [17, 18]) or the non-parametric approach [19].

3.2 Simulation Algorithm (Next-reaction Method)

There are at least two approaches to simulate realizations of the stochastic process underlying a DES of an iSTM in continuous time [20]. The first approach relies on choosing the transition that occurs at a faster rate immediately after an event occurs, also referred to as "next reaction" methods [21]or as an 'event-specific distribution' approach [22]. These algorithms require knowledge of the set of transition-specific intensity functions for each state, which allows sampling the latent times for each competing transition to determine the next event. The second approach consists of first determining the time of the next event of any type and then, conditional on that time, determining the type of event that will occur. Our DES implementation is based specifically on Anderson's modified next reaction method [23] and adaptated for general non-exponential inter-event times [24]. This approach involves setting the transition-specific cumulative intensity functions, their corresponding inverse functions, and the adjacency matrix A that encodes the set of directly connected states, as represented by the state-transition diagram (i.e., for each state, determine which direct transitions the model permits).

At a given event time, T_n, in order to determine the next transition for each person, we first determine the set of possible states that a person can transition to given their current state X_{T_n}. This can be done using an adjacency matrix A. In the example with three states (Figure 1), one person starts at S1 ($X_{T_n} = s_1$) and faces two possible transitions, to S2 or to S3. Second, we sample a set of latent inter-event times $\tau_{s_1,s}$, for $s \in \{s_2, s_3\}$ via the inversion or non-parametric methods (see Section 3.1). With samples of the time from S1 to S2, τ_{s_1,s_2}, and from S1 to S3, τ_{s_1,s_3}, we can determine the first event out of S1. We compute the minimum inter-arrival time $\tau^*_{s_1,s}$ among the latent inter-event times, i.e.:

$$\tau^*_{s_1,s} = \min_{s \in \{s_2,s_3\}} \left(\tau_{s_1,s}\right) \qquad (3)$$

With the realized inter-event time $\tau^*_{s_1,s}$, we update the current state determined by its second index s^*.

$$s^* = \arg\min_{s \in \{s_2,s_3\}} \left(\tau_{s_1,s}\right) \qquad (4)$$

Finally, the simulation time is updated to $T_{n+1} = T_n + \tau^*_{s_1,s}$, and the state at the next event time is updated to $X_{T_{n+1}} = s^*$.

Figure 2 illustrates these steps for the more general case of one person in a model with m states, starting from state S1 and facing $(m-1)$ possible transitions out of S1.

4. Adding Complexity to a DES

The next-reaction method relies on transition-specific intensity functions, capturing time-dependence, treatment effects and individual-level heterogeneity influencing transition rates. These complexities would require specialized techniques in a cSTM, like tunnel states, multidimensional transition probability arrays, and stratified health states [11, 12]. In contrast, DES does not require such techniques, and can incorporate treatment, covariates, time dependence, and state-history directly via transition-specific intensity functions, $\lambda(t)_{r,s}$.

4.1 Time Dependence

When intensities depend on simulation time or time since state-residency started, the distribution of time in state r before transitioning to state s, $F(\tau)_{r,s}$, may be unknown analytically. However, the inversion [18] and non-parametric methods [19] offer reliable solutions for sampling from an unknown $F(\tau)_{r,s}$ via the cumulative intensity function $\Lambda(t)_{r,s}$.

4.2 Treatment and Covariate Effects

The effects of treatments and covariates on transition-specific intensities can be modeled using the proportional hazards assumption, which provides an interpretable way to encode these effects, like hazard ratios in clinical trials. Other formulations, such as accelerated failure time, are also possible.

To illustrate how treatment and covariate effects influence transition rates under proportional hazards, consider a treatment that only affects the transition rate from state r to s. The intensity function for the ith individual describing the transition from state r to s is then expressed as:

$$\lambda\left(t, z_i\big(g(t)\big), \mathbb{I}_i(t)\right)_{r,s} = \lambda_0(t;\theta)_{r,s}\ e^{[\beta\ z_i(g(t))+\gamma \mathbb{I}_i(t)]} \qquad (5)$$

Where $\lambda_0(t;\theta)_{r,s}$ is the baseline intensity function which in our framework can belong to any flexible parametric model parameterized by the vector (θ) (note that non-parametric specifications of $\lambda_0(t)_{r,s}$ are also possible); (β) is a vector of time-fixed covariate effects; $z\big(g(t)\big)$ is a vector of covariates (possibly time-varying) and $g(t)$ is a function of time; (γ) is a time-fixed scalar treatment effect; and $\mathbb{I}_i(t)$ is an indicator function of receiving the treatment, possibly time-varying. Note that this formulation makes treatment effects enter the intensity function like a covariate.

4.3 State-history Dependence

If the simulation stores previous simulated events, the information of previously visited states, $\mathcal{H}_t$, can enter the intensity function as a covariate as well; as shown in Equation 5.

4.4 Capacity and Resource Constraints

Depending on the application, it may be necessary to explicitly model delays due to resource or capacity constraints. Queues are well-studied models that represent continuous-time processes, compatible with DES [25, 26]. In healthcare, queues can represent capacity constraints – for instance, limited appointments for diagnostic imaging can delay diagnosis and treatment, allowing disease severity to progress.

5. Case study: a Cost-effectiveness Analysis using a Time-dependent Sick-Sicker model

We demonstrate the implementation of our DES framework by simulating in continuous time the time-dependent 4-state Sick-Sicker model [11, 12, 13, 27].

A cohort of 25-year-olds in the "Healthy" state (denoted "H") is simulated until they reach age 100. Cohort members face an age-specific background mortality rate, $\lambda(t)_{H,D}$ (`v_r_HDage`). Healthy individuals are also at risk of a hypothetical disease with 2 stages, "Sick" and "Sicker", denoted by "S1" and "S2", respectively. The disease process is described as follows:

- Healthy individuals face a constant annual rate of getting sick, $\lambda_{H,S1}$ (`r_HS1` in R notation). The time from H to S1, $\tau_{H,S1}$, is exponentially distributed, $\tau_{H,S1} \sim Exp(\lambda_{H,S1})$.
- Sick individuals are at risk of progressing to the more severe Sicker health state, as a function of time spent in the Sick state following a Weibull intensity parameterized as proportional hazards, i.e., $\lambda(t)_{S_1,S_2} = \beta k t^{k-1}$, where t is the time spent in S1, and β (`r_S1S2_scale`) and k (`r_S1S2_shape`) are the scale and shape parameters for the Weibull hazard function, respectively. The time from S1 to S2, $\tau_{S1,S2} \sim Weibull(\beta, k)$. Both β and k are greater than zero. Sick individuals can recover and become Healthy again at a constant rate $\lambda_{S1,H}$ (`r_S1H`), i.e., $\tau_{S1,H} \sim Exp(\lambda_{S1,H})$. However, once individuals reach S2, they cannot recover; i.e., the transition rate to S1 or H from S2 is zero.
- Individuals in S1 and S2 face an increased hazard of death compared with healthy individuals, which is modeled as a relative increase to the age-specific background mortality rate with a hazard ratio (HR) of 3 (`hr_S1`) and 10 (`hr_S2`), respectively.

Individuals in S1 and S2 also experience increased health care costs and reduced quality of life (QoL) compared with individuals in H. When individuals die, they transition to the absorbing "Dead" state (denoted by "D"), with no further transitions. We use the model to compute the cohort's expected costs and quality-adjusted life-years (QALYs) over their lifetime. We discount costs and QALYs at a 3% annual rate.

We assess the cost-effectiveness of 4 strategies: standard of care (strategy SoC); strategy A; strategy B; and a combination of strategies A and B (strategy AB).

- Strategy A administers treatment A, increasing the QoL of individuals in S1 from 0.75 (utility without treatment, `u_S1`) to 0.95 (utility with treatment A, `u_trtA`). Treatment A costs $12,000 per year (`c_trtA`). This strategy does not affect the QoL of individuals in S2, nor changes the risk of becoming sick or progressing through the sick states.
- Strategy B implements treatment B, reducingthe rate of progressing to Sicker by 40%, costs $13,000 per year (c_trtB), and does not affect QoL. The effect of strategy B is encoded as a hazard ratio $HR = 0.6$ (`hr_S1S2_trtB`).
- Strategy AB involves administering both treatments A and B with both effects and additive costs.

We assume that it is not possible to distinguish between Sick and Sicker patients; therefore, individuals in both disease states receive treatment(s) under all treatment strategies. After comparing the 4 strategies in terms of expected QALYs and costs, we calculate the incremental cost per QALY gained for non-dominated strategies.

Model parameters and the corresponding R variable names are presented in Table 1 . We follow the notation conventions described in the DARTH coding framework [28]. These conventions are proposed for clarity and consistency; analysts are welcome to adopt their own coding and naming and adjust the accompanying R code accordingly. Our implementation mainly relies on the `data.table` R package for memory efficiency and computational speed for handling large data sets [29]. In the appendix, we show execution time distributions of our code and describe package dependencies.

5.2 Useful Data Objects

Our simulation framework uses three distinct `data.table (dt)` objects:

a. `dt_baseline`: stores baseline characteristics, initial states, and initial times for all simulated people. This `dt` is only needed to initialize the simulation.

b. `dt_next_event`: stores one instance of the latent transitions that can occur for each simulated person at a given time. This `dt` will be overwritten several times as new events occur, only storing relevant information to determine the next event for each person in the simulation.

c. `dt_event_history`: stores the history for each simulated person's transitions and the times they occurred. This `dt` stores all instances of `dt_next_event` in long format.

Additionally, a numbered adjacency matrix, A (labeled as `m_num_adj`), is used to determine which are the possible transitions from each state. For the Sick-Sicker model the adjacency matrix is informally defined as:

$$\mathbf{A} = \begin{array}{c|cccc} & H & S1 & S2 & D \\ \hline H & 0 & 1 & 0 & 2 \\ S1 & 3 & 0 & 4 & 5 \\ S2 & 0 & 0 & 0 & 6 \\ D & 0 & 0 & 0 & 0 \end{array} \qquad (6)$$

A conventional adjacency matrix is a square matrix with only 0 or 1 values, where 1 indicates a possible transition (i.e., the states are connected in the state-transition diagram), and 0 otherwise. We enumerate nonzero entries to easily identify health state transitions. Alternative implementation styles in DES often include, "event-oriented" or "process-oriented" approaches. This tutorial focuses on the continuous-time stochastic process underlying an iSTM and organizes the code around the state-transition structure defined by the adjacency matrix.

5.3 Sequences of Events and Event Times

The provided code generates the sequence and timing of events for each person in two modules (See eFigure 1 in the appendix).

5.3.1 Module 1: Currently Occupied States

The first module reads `dt_main`, supplied as `dt_baseline` or `dt_next_event`, identifies the occupied states by simulated individuals, and creates subsets for each occupied state stored in a list `l_subsets_dt_main`. For example, if 50% start the simulation Healthy and 50% in the Sick state, the module splits `dt_baseline` into two `dt_subsets`, one with people in the Healthy state and one with people in the Sick state. It also reads the life table, `dt_bckgrd_mortality` and produces sick-state-specific life tables using the provided hazard ratios, `dt_mortality_Sick` and `dt_mortality_Sicker`.

```
#-----------------------------------------------#
# Module 1: Currently Occupied Health States ----
#-----------------------------------------------#
## Read dt
dt_main  <- copy(dt_baseline)       # For first event
# dt_main <- copy(dt_next_event)    # For events after the first one

## Determine set of current states ----
# set of states currently occupied by at least one simulated person, "current
states"
v_current_state       <- unique(dt_main$from)
# names of "current  states"v_current_state_names <- paste("from", v_current_
state, sep = "_")

## Subset dt_main based on current state(s) ----
l_subsets_main_dt <- list()
for( i in seq_along(v_current_state)){
l_subsets_main_dt[[i]]   <-  as.data.table(
  dt_main[from == v_current_state[i], ])
}
names(l_subsets_main_dt) <- v_current_state_names

## Mortality life tables ----
### Mortality from Sick
dt_mortality_Sick  <- copy(dt_bckgrd_mortality)
# apply hazard ratio for Sick
```

```
dt_mortality_Sick[, death_rate := death_rate*hr_S1]
### Mortality from Sicker
dt_mortality_Sicker <- copy(dt_bckgrd_mortality)
# apply hazard ratio for Sicker
dt_mortality_Sicker[, death_rate := death_rate*hr_S2]
```

5.3.2 Module 2: Transitions between Health States

This module is the core of the simulation, divided into submodules for each non-absorbing state. The Sick-Sicker model has three submodules, corresponding to the Healthy, Sick and Sicker states. Adding more submodules requires minimal extra work since submodules are standardized with four tasks, even in models with a larger state-space. The tasks for the ith submodule are summarized in Table 2.

5.3.2.1 Submodule 1: Transitions from Healthy

Task 1: Identify possible transitions from Healthy and expand `dt_subset` to include rows for the latent event times from H, creating `dt_subset_long`.

```
# ------------------------------------------------ #
# MODULE 2: Transitions between Health States ----
# ------------------------------------------------ #
## 2.1 Transitions from the Healthy State (H) ----

# ---------------------------------#
# Task 1: Possible Transitions ----
# ---------------------------------#
# Temporary copy of dt_subset for H
dt_subset      <- l_subsets_main_dt$from_H
current_state  <- unique(dt_subset$from) # current state
# Number of possible transitions out of Healthy (using adjacency matrix m_Tr_
r)
transition  <- as.numeric(m_Tr_r[current_state, m_Tr_r[current_state,] !=0])
# Expand temp to have one row for each possible transition for each ID
ID  <- unique(dt_subset$ID)# unique IDs
setkey(dt_subset, ID)
## CJ is like expand.grid  for data.tables
dt_subset_long <- merge(dt_subset,CJ(ID, transition), all.x = F)
dt_subset_long[, c("from","to") := NULL]

## Assign "from" and "to" states based on table version of m_Tr_r, dt_trans_k
eys
# Indexed join in data.table using "on" sets the Key automatically
dt_subset_long <- dt_subset_long[dt_trans_keys, on = "transition", nomatch =
0L]
dt_subset_long <- dt_subset_long[order(ID, transition)]
```

Task 2: Sample latent inter-event times (τ), for transitions out of Healthy, following two routines from Section 3.1.

- Subtask 2.1 H -> S : Healthy individuals face a constant annual rate of getting sick, implying that $\tau_{H,S1} \sim exp(\lambda_{H,S1})$. We use R's base function `'rexp()'` to sample this latent inter-event time $\tau_{H,S1}$, employing the inversion method.

```
# ------------------------------------ #
# Task 2: Sample Latent Event Times ----
# ------------------------------------ #
## Transition #1  (H -> S1) ----
# ----------------------------------- #
# Constant annual rate of becoming Sick when Healthy
# Sample from an exponential distribution
if(r_HS1 != 0){   # Case when the transition is allowed
 dt_subset_long[transition == 1, T_stop := rexp(.N, rate = r_HS1) + T_n]
  } else {        # Case when transition is not allowed, assign infinite time
 dt_subset_long[transition == 1, T_stop := Inf]
  }
```

- Subtask 2.2 H -> D : Healthy individuals face an age-specific background mortality. We sample latent times to death from `dt_bckgrd_mortality` using non-parametric sampling (Section 3.1).

```
# ---------------------------- #
## Transition #2  (H -> D) ----
# ----------------------------#
# Obtain vector of probabilities of dying from life table
dt_time2death_probs <- as.data.table(obtain_probs_des(dt_bckgrd_mortality))
# Only keep  rows for which the transition is possible,i.e., trans #2
dt_trans2  <- dt_subset_long[transition == 2]

# Set key in dt_trans2
setkey(dt_trans2, Age, Sex)
# Set key in dt_time2death_probs
setkey(dt_time2death_probs, Age, Sex)
# Indexed merge (all.x == T performs left-join)
dt_2sample_time2death <- merge(
  dt_trans2[, Age := ceiling(Age)],
  dt_time2death_probs, all.x = TRUE)

# Generate a matrix of probabilities from rows that matched and merged.
m_probs <- as.matrix(dt_2sample_time2death[, .SD, .SDcols = patterns("^DP")])

# Sample time to death using nonparametric sampling
# Using `nps_nhppp` with a uniform correction
dt_trans2[, T_stop := nps_nhppp(m_probs = m_probs, correction = "uniform")]

# Consistent ordering
setkey(dt_subset_long, ID, T_n)
setkey(dt_trans2, ID, T_n)
```

```
# Write into dt_subset_long
dt_subset_long[transition == 2, T_stop := dt_trans2$T_stop]
```

Task 3: Use samples of latent event times for all possible transitions per individual in the Healthy state to identify each individual's next transition following the algorithm in Section 3.2.

```
# ---------------------------------------- #
# Task 3: Predict the Next Event from H ----
# ---------------------------------------- #
# Create an indicator for minimum value in T_stop by ID
dt_subset_long[, status := as.numeric(T_stop == min(T_stop)), by = ID]
# Recover the inter-event time, tau
dt_subset_long[,tau := T_stop - T_start]
```

Task 4: Update each individual's simulation time and age, which are needed for consistent sampling time to death. Add the predicted inter-event time from **Task 3** to the current simulation time.

```
# ---------------------------------------- #
### Task 4: Update Important Variables ----
# ---------------------------------------- #
# Update simulation time
# We use years for event times (T_n) and age (Age).
dt_subset_long[status == T, T_n := T_stop]
# Update age
dt_subset_long[status == T, Age := round(T_n)]
# round(T_n) to match age-specific row in lifetables
```

We track events histories, appending the new event information to `dt_event_history` in long format.

```
# ---------------------------------- #
# Update dt_event_history
dt_event_history  <<- rbind(dt_event_history, dt_subset_long)
# Guarantee consistent ordering
setkey(dt_event_history, ID, T_n, Event_num, transition)
# Double arrow assignment (<<-) used here to update dt_event_history outside
the function environment (see function dt_next_event in Section 5.3.3)
```

5.3.2.2 Submodule 2: Transitions from Sick

Recall that Sick individuals's hazard of progressing to the Sicker state increases as a function of time spent Sick. They also face a constant recovery rate to Healthy and increased age-dependent mortality while Sick.

We follow the same structure as in the submodule for Healthy but modify **Task 2** to sample the latent inter-event times from Sick to Sicker using a Weibull distribution with parameters `r_S1S2_scale` and `r_S1S2_shape`, latent inter-event times from Sick to Healthy from an exponential distribution with rate `r_S1H`, and time to death while Sick from the life table and

hazard ratio. We omit the code for other Tasks, as they are mostly unchanged (see the appendix and GitHub repository).

Task 2: Sample latent inter-event times for transitions out of Sick following the two routines in Section 3.1. To include the effect of strategies "B" and "AB" on reducing progression to Sicker, we scale the baseline scale parameter of the Weibull distribution by `hr_S1S2_trtB` , assuming proportional hazards (Section 4.2).

```
## 2.2 Transitions from the Sick State (S1) ----
# ------------------------------------- #
# Task 2: Sample latent Event Times ----
# ------------------------------------- #

# ---------------------------------- #
## Transition #3 (S1 -> H) ----
# ---------------------------------- #
# Sample from exponential distribution
if(r_S1H != 0){ # case when the transition is  allowed
 dt_subset_long[transition == 3, T_stop := rexp(.N, rate = r_S1H) + T_n]
}else {         # case when the transition is not  allowed
 dt_subset_long[transition == 3, T_stop := Inf]
}
# ------------------------------#
## Transition #4 (S1 -> S2) ----
# ------------------------------#
# State-residence time-dependent hazard of transition from Sick to Sicker
## Apply the treatment effect of strategy B or strategy AB
if (strategy == "B" || strategy == "AB"){
  r_S1S2_scale_ph <- r_S1S2_scale_ph * hr_S1S2_trtB
}
## Reparameterize to match accelerated failure time specification
r_S1S2_scale_aft <- r_S1S2_scale_ph ^(-1 / r_S1S2_shape)

## Sample inter-event time from Weibull distribution
if(r_S1S2_shape*r_S1S2_scale_aft != 0){  # when  transition is  allowed
 dt_subset_long[transition == 4,
                T_stop := rweibull(.N, shape = r_S1S2_shape,
                                   scale = r_S1S2_scale_aft) + T_n]
 } else {                                 # when transition is not  allowed
 dt_subset_long[transition == 4, T_stop := Inf]
 }
```

For sampling the time to death while Sick, we use the same non-parametric routine as in **Submodule 1,** but replace `dt_bckgrd_mortality` with the health-state-specific life table `dt_mortality_Sick` generated in **Module 1**.

5.3.2.3 Submodule 3: Transitions from Sicker

This submodule follows the same structure as before but restricts the only transition from Sicker to Dead, so **Task 3** is unnecessary. Like **Submodule 2**, **Task 2** uses the life table for the Sicker state from **Module 1**, `dt_mortality_Sicker`.

5.3.3 Module 3: Dealing with Recurrent Events

In a purely progressive STM without transitions from Sick to Healthy, the maximum number of iterations of Modules 1 and 2 are fixed, ensuring all simulated individuals die or reach 100 years. However, allowing a transition back to Healthy implies creating loops, making the maximum iterations of Modules 1 and 2 (now a random variable) unknown.

This module requires two components. First, wrap Modules 1 and 2 in `'sim_next_event()'`. Here, we only show the structure, see the appendix and the GitHub repository for details.

```
# ----------------------------------- #
# Module 3: Simulation with Recurrence to Healthy
# ----------------------------------- #

# ----------------------------------- #
## Structure of the Function 'sim_next_event()'
# ----------------------------------- #
sim_next_event <- function(l_params){

    # Module 1: Setup
      ## Identify current state for each person in dt_next_event
    ## Subset dt_next_event based on current state

    # Module 2: Simulate next event
    ## Submodule 1: Transitions from H
       ### Tasks 1-4 that apply
    ## Submodule 2: Transitions from Sick
       ### Tasks 1-4 that apply
    ## Submodule 3: Transitions from Sicker
       ### Tasks 1-4 that apply

    # Reset and update variables of interest in dt_next_event
    ## Keep simulated transitions that occurred
    ## Drop simulated people who died (-> 1 row per alive simulated person)
    ## Reset and update variables of interest

return(sim_next_event)
}
```

Second, we run a **`while`** loop that executes `sim_next_event` until everyone in the simulation has died or reached 100 years of age. We also set a **`warning`** for exceeding the maximum number of events as a sanity check.

```
# ---------------------------------- #
## Simulation with Recurrence
# ---------------------------------- #
# Initialize dt_next_event
dt_next_event  <- dt_baseline
# Initialize dt_event_history
dt_event_history <- dt_baseline
# Set seed for reproducibility
n_seed             <- 2
set.seed(n_seed)
# Max. number of events before triggering warning
# e.g., if more than 1,000 have occurred, trigger a warning message
max_num_events <- 1e3

# Run while loop until everyone in the simulation has died
          while(
            # stopping criteria: each row in dt_next_event represents
            # the number of people still alive in the simulation
            nrow(l_params$dt_next_event) > 0
            ){

            # Function `sim_next_event` updates dt_next_event for everyone wh
o is still alive
            l_params$dt_next_event <- sim_next_event(l_params)
            # Warning:
            if(unique(l_params$sim_next_event$Event_num) > max_num_events){
             warning(
               paste0("Within a simulation time-horizon of"
               ,l_params$time_horizon[2]-l_params$time_horizon[1], "years,
               the max number of events (i.e., transitions between health
               states) for at least one person in the simulation has
               exceeded", max_num_events,".
               Reconsider if the simulation should continue since these
               dynamics might not reflect what you intended."
               ))
              }
            }
```

5.4 Post-processing and Auxiliary Modules

Below are code snippets illustrating the main components used to process the DES results and calculate outcomes. For the complete code to generate cost-effectiveness and epidemiological outcomes, refer to the appendix and GitHub repository.

5.4.1 Post-processing Module A: Compute a Cost-effectiveness Analysis

This module computes a CEA of the four strategies. First, it generates the event history, `dt_event_history`, for each strategy using a **`for`** loop over `v_strategies`, executing **Module 3** each time.

We set a common random number (CRN) stream across all strategies to reduce variance in Monte Carlo estimates of incremental costs and benefits. [30] The event history tables are stored in `l_event_history_strategies`, which feeds into **Module A** for the CEA.

```
# ---------------------------------------------------------------- #
# Post-processing Module A: Compute a cost-effectiveness analysis
# ---------------------------------------------------------------- #
# Set reference list of parameters
l_params_ref              <- l_params
# ---------------------------------------- #
## Run Simulation for all Strategies ----
# ---------------------------------------- #
# Names of Strategies
v_strategies_names <- c( "SoC", "A", "B", "AB" )
# Since the effect on transition dynamics for SoC == A, and B == AB,
# we can execute the simulation for "SOC" and "B" only.
v_strategies       <- c("SoC","B")
# Set seed for reproducibility
n_seed             <- 2
#Initialize list of event histories
l_event_history_strategies   <- list()

# For-loop running simulation under each strategy
for(i in seq_along(v_strategies)){
  # Initialize dt_event_history for each strategy
  dt_event_history    <- data.table()
  # Reset initial params for each iteration
  l_params            <- l_params_ref
  # Update strategy
  l_params$strategy   <- v_strategies[i]
  # Common random number stream across scenarios
  set.seed(n_seed)
  # Run simulation under i^th strategy
  while(nrow(l_params$dt_next_event) > 0){
    l_params$dt_next_event <- sim_next_event(l_params)
  }
  # Store dt_event_history for i^th strategy
  l_event_history_strategies[[i]] <- as.data.table(dt_event_history)
}
names(l_event_history_strategies) <- v_strategies

#  Copying event histories for equivalent strategies
# "SoC" and "A"
l_event_history_strategies$A  <- l_event_history_strategies$SoC
# "B" and "AB"
l_event_history_strategies$AB <- l_event_history_strategies $B
```

Next, we compute the expected discounted total costs and utilities under each strategy by appending each `dt` in `l_event_history_strategies` and creating `dt_all`, containing the

four event histories. Costs and utilities are assigned to the time spent in each non-absorbing state, as well as for each treatment and transition between states (see Section 5).

```
# -------------------------------#
# Assign Costs and Utilities -----
# ------------------------------ #
# Indicator of strategy, and subset events that occurred
dt_SoC <- l_event_history_strategies$SoC[status == T][, Trt := "SoC"]
dt_A   <- l_event_history_strategies$A[status == T][, Trt := "A"]
dt_B   <- l_event_history_strategies$B[status == T][, Trt := "B"]
dt_AB  <- l_event_history_strategies$AB[status == T][, Trt := "AB"]
# Append event histories of all strategies
dt_all <- rbind(dt_SoC,
                dt_A  ,
                dt_B  ,
                dt_AB  )
# Assign costs
## for episodes
dt_all[, Annual_Cost_ep:= fifelse(from == "H", c_H,
                         fifelse(from == "S1", c_S1,
                         fifelse(from == "S2", c_S2, NA)))]
## for treatments
dt_all[, Annual_Cost_trt:= fifelse(Trt=="A"&(from=="S1"|from=="S2"), c_trtA,
                          fifelse(Trt=="B"&(from=="S1"|from=="S2"), c_trtB,
                          fifelse(Trt=="AB"&(from=="S1"|from=="S2"),
                                  dc_trtA + dc_trtB, 0)))]
## for transitions
dt_all[, Cost_trans:= fifelse(from == "H" & to == "S1", c_HS1,
                      fifelse(to == "D", c_D, 0))]
# --- #
# Assign utilities
## for episodes
dt_all[, Annual_Utility_ep := fifelse(from == "H", u_H,
                              fifelse(from == "S1", u_S1,
                              fifelse(from == "S2", u_S2, NA)))]
## for treatments
dt_all[, Annual_Utility_trt:= fifelse(Trt == "A" & from =="S1", u_trtA,
                              fifelse(Trt == "AB" & from =="S1", u_trtA,0))]
## for transitions
dt_all[, Utility_trans := fifelse(from == "H" & to =="S1", u_HS1,0)]

# --- #
# Total costs and utilities per event
dt_all[, Total_cost_ep :=
          Annual_Cost_ep    + Annual_Cost_trt + Cost_trans]
dt_all[, Total_utility_ep :=
          Annual_Utility_ep + Annual_Utility_trt + Utility_trans]
```

We exponentially discount all outcomes in continuous time. Let O_i be a cumulative outcome for the ith individual for the interval $[t_1, t_2]$, and ρ the exponential discount rate, then the discounted cumulative outcome over $[t_1, t_2]$ for the ith individual, $\widehat{O_i}$, is:

$$\widehat{O_i} = \int_{t_1}^{t_2} O_i \, e^{-\rho t} dt = O_i \frac{(e^{-\rho t_1} - e^{-\rho t_2})}{\rho} \qquad (7)$$

For discounting, the initial time is 0. We shift all inter-event times by subtracting the start age of the cohort (25 years of age), i.e., the time after birth when the simulation starts for each individual.

```
# ------------------------------ #
# Discounted Cost/Utilities -----
# ------------------------------ #
## Discounted costs
### Exponential discounting, lower and upper bounds per event
dt_all[,v_dwc_t1  := exp(-(d_c) * (T_start - 25)), by = .(ID,Trt)]
dt_all[,v_dwc_t2  := exp(-(d_c) * (T_stop  - 25)), by = .(ID,Trt)]
### Exponential discounting over the event  interval
dt_all[,v_dwc_t12  := (v_dwc_t1-v_dwc_t2)/d_c, by = .(ID,Trt)]
## Discounted effects
### Exponential discounting, lower and upper bounds per event
dt_all[,v_dwe_t1  := exp(-(d_e) * (T_start - 25)), by = .(ID,Trt)]
dt_all[,v_dwe_t2  := exp(-(d_e) * (T_stop  - 25)), by = .(ID,Trt)]
### Exponential discounting over the event interval
dt_all[,v_dwe_t12  := (v_dwe_t1-v_dwe_t2)/d_e, by = .(ID,Trt)]
## Apply discounting
dt_all[ , Disc_total_cost    := Total_cost_ep    * v_dwc_t12 ]
dt_all[ , Disc_total_utility := Total_utility_ep * v_dwe_t12 ]
# ------------------------------------------ #
# Lifetime Discounted Costs and Utilities ----
# ------------------------------------------ #
# ----- #
# Cumulative costs and utilities at each event for each ID
dt_all[ , Cum_total_cost   := cumsum(Disc_total_cost   ), by = .(ID,Trt)]
dt_all[ , Cum_total_utility:= cumsum(Disc_total_utility), by = .(ID,Trt)]
#-------#
#Lifetime costs and utilities per ID
dt_all[ , Lifetime_total_cost   := sum(Disc_total_cost   ), by = .(ID,Trt)]
dt_all[ , Lifetime_total_utility:= sum(Disc_total_utility), by = .(ID,Trt)]
#-------#
# Collapse to a dt with Lifetime costs/utilities per row
dt_all_lifetime <- dt_all[, .(Lifetime_total_cost = sum(Disc_total_cost),
                              Lifetime_total_utility = sum(Disc_total_utility
                              )), by = .(ID, Trt)]
#-------#
# Lifetime expected costs/utilities per strategy
dt_all_mean_lifetime <- dt_all_lifetime[,
                        .(Mean_total_cost = mean(Lifetime_total_cost),
```

```
                          Mean_total_utility = mean(Lifetime_total_utility)),
                        by = Trt]
# ------------------------------------------------- #
## Incremental Cost-effectiveness Ratios (ICERs) ----
# ------------------------------------------------- #
# ICER table
df_cea <- calculate_icers(cost     = dt_all_mean_lifetime$Mean_total_cost,
                          effect   = dt_all_mean_lifetime$Mean_total_utility,
                          strategies = dt_all_mean_lifetime$Trt)
```

5.4.2 Post-processing Module B : Compute Epidemiological Outcomes

Survival probabilities and disease prevalence are epidemiological outcomes obtainable directly from cSTMs and DES. Another outcome is dwell time in a specific health state, like time spent in the recurrence-free state in cancer models. Due to the time-to-event formulation of DES, dwell times at each alive state for each individual and for the cohort are directly obtained from the DES output. In this module, we compute survival probabilities, disease prevalence, and the distribution of dwell times for the entire cohort at each health state in the Sick-Sicker Model.

```
# ---------------------------------------------------------------- #
# Post-processing Module B: Compute Epidemiological Outcomes -----
# ---------------------------------------------------------------- #

# ------------------------------------- #
## Dwell Times across Health States ----
# ------------------------------------- #
# Individual average dwell time at each alive state under each strategy.
dt_dwell <- dt_all
dt_dwell <- dt_dwell[, .(dwell_mean  =  mean(tau)),
                     by = c("ID", "from", "Strategy")]
# Distribution of the average dwell time at each alive state under each strat
egy.
dt_dwell_pop        <- dt_dwell[, .(
  mean              =  mean(dwell_mean),
  quantile_2_5pct   =  quantile(dwell_mean, probs = c(.025)),
  quantile_5pct     =  quantile(dwell_mean, probs = c(.05)),
  quantile_25pct    =  quantile(dwell_mean, probs = c(.25)),
  quantile_50pct    =  quantile(dwell_mean, probs = c(.50)),
  quantile_75pct    =  quantile(dwell_mean, probs = c(.75)),
  quantile_95pct    =  quantile(dwell_mean, probs = c(.95)),
  quantile_97_5pct  =  quantile(dwell_mean, probs = c(.975))),
                     by = c("from", "Strategy")]

# ------------------------------------------------- #
## Cohort survival probabilities and prevalence ----
# ------------------------------------------------- #
# Generate a continuous time trace matrix for each strategy
l_dt_trace_strategies <- list()
```

```r
for (i in seq_along(l_event_history_strategies)) {
  l_trace_strategies[[i]] <- cont_time_trace_sick_sicker(
    dt_event_history = l_event_history_strategies[[i]],
    v_names_states   = l_init_params$v_names_states,
    time_init        = l_init_params$time_horizon[1],
    proportions      = T ,
    rounding_factor  = 5 ,
    long             = F
  )
  # Compute survival and prevalence from the trace
  l_dt_trace_strategies[[i]][,':='(
    # Check everything sums to 1
    Total      = H + S1 + S2 + D,
    # Sum of the proportions of all alive states
    Survival   = H + S1 + S2,
    # (Sum proportions of sick-states) / (sum proportions of alive-states)
    Prevalence = (S1+S2)/(H+S1+S2)
  )]
}
# Assign names of strategies
names(l_dt_trace_strategies) <- c("SoC", "B", "A", "AB")
```

5.4.3 Post-processing Module C: Compute a Probabilistic Analysis

This Module computes a probabilistic analysis (PA) to assess how model parameter uncertainty affects cost-effectiveness outcomes. It samples parameter sets using Latin Hypercube sampling[31] and user-specified distributions. Each set is used to simulate strategies and calculate their cost-effectiveness outcomes, stored in a PA data set consistent with previous tutorials [11, 12]. This compatibility allows use of validated functions for sampling parameter sets, and outcome calculations.

Figure 4 illustrates the CEA of the Sick-Sicker Model, with the scatter plot (See Panel A), cost-effectiveness acceptability curves (CEACs) and cost-effectiveness frontier (CEAF) (See Panel B), the expected loss curves (ELCs) (See Panel C), and the expected value of perfect information (EVPI). We sampled 1,000 parameter sets for PA results.

5.4.4 Auxiliary Module: Continuous Time Trace Matrix and State Occupancy Plot

In the appendix, we include a module with functions that map the DES output to a cohort trace matrix (i.e., the proportions of simulated people in each state over time). This function enables quick calculations of epidemiological outcomes like survival probabilities and prevalence, similar to a discrete-time cSTM. Analysts can compare the proportions at each state over time from the DES with those simulated using discrete-time cSTMs. We also generate state occupancy plots to visually assess how the DES captures transition dynamics and the convergence of Monte Carlo estimators of state proportions over time.

5.5 Convergence of Monte Carlo Estimator

Finally, we explore how many individuals to simulate with the DES to produce estimates that converge to those obtained from an equivalent cSTM (mean-process model). In this example, we focus on DES-based estimators of the proportions of people in each state over time, along with their variability expressed as bootstrap confidence bounds at the 2.5^th and 97.5^th percentiles around the median. We showed that the variability of these state-occupancy estimates declines rapidly as the number of simulated individual by the DES increases (Figure 5). An implementation of this analysis in a distributed setting is available in the appendix and Github repository.

6. Discussion

This tutorial introduces a novel, open-source DES framework designed for modeling realistic continuous-time, individual-level STMs. It supports features such as simulation and state-residence time dependence, treatment effects and other individual-level characteristics. We demonstrate the framework's versatility and advantages within HTAs and medical decision making by providing a case example of a CEA in R using the previously published Sick-Sicker model.

DES offers computational advantages in terms of simulating models in which transition dynamics depend on state-residence time (i.e., the type of dynamics for which cSTMs expand their state space using "tunnel" states, see the tutorial paper by *Alarid-Escudero et al., 2023)*. Alternative approaches to this for multi-state cohort models with semi-Markov dynamics include the use of Kolmogorov's forward equations to calculate state-occupancy probabilities [32].Our individual-level DES samples event times from hazard functions, handling path dependence, competing risks, and time-varying covariates without solving integro-differential equations, providing a flexible alternative.

We demonstrate how treatment effects and individual-level characteristics influencing health-state transitions can be incorporated via a proportional-hazards approach, by scaling intensity functions. Incorporating heterogeneity in cSTMs adds computational load since the model assumes cohort homogeneity, requiring to to run as many simulations as the number of subpopulations. Other discrete-time methods like iSTMs may incorporate heterogeneity and individual-level factors without multiple runs, but DES offers a computational advantage by evaluating health states only at the times events occur. Our DES framework also captures within- and between-person variability in dwell times without complex calculations. For example, to assess personalized cancer treatments prolonging recurrence-free states, recovering dwell time distribution is crucial. cSTMs only provide mean dwell times, while discrete-time iSTMs recover both the population and individual variability. With DES, individual dwell times are directly obtained and can be summarized for the entire cohort in a few code lines.

When implementing a DES, there are important analytic choices. Intensity functions must accurately reflect the joint distribution of time-to-event for each transition, which can be achieved by tracking individuals' times and states. A multistate model can then estimate

transition intensities considering competing events [16, 33]. Using estimates from different studies and populations can misrepresent dynamics, potentially biasing estimates of epidemiological and economic outcomes. Calibration is necessary to match key epidemiological outcomes of the target population. Additionally, the probabilistic analysis should quantify how uncertainty in the transition rates, e.g., uncertainty on the hazard ratios that affect mortality, affects decision making.

Deciding whether to use specialized software (like 'simmer' [34, 35], 'quecomputer' [36], 'microsimualtion' [37] and 'hesim' [38]) or lower-level code to implement DES is important. Our DES implementation offers a transparent, domain-specific reference for individual-level, continuous-time state-transition modeling in health economics, focusing on accessibility, auditability, modifiability, and integration with survival analysis and probabilistic analysis workflows in R. It complements general DES packages suited for resource/queue management or high performance. This tutorial uses a specific coding and naming convention for readability, consistency, and ease of extension, but choosing any suitable convention is acceptable and should align with team practices.

iSTMs, whether simulated in discrete or continuous time like DES, may bias estimates of decision uncertainty and value of information. Goldhaber-Fiebert et al 2025 provide a closed-form and numerical characterization of this bias, showing the bias is asymptotically consistent and can be minimized by increasing the number of simulated individuals [39]. Analysts using DES should assess the number of simulated individuals needed to reduce bias to tolerably low levels. To keep the tutorial focused, we used a single random number seed generating a common random number (CRN) stream across strategies as a variance-reduction technique, requiring smaller samples [30]. Analysts conducting comparative experiments may further improve precision by mapping random number sub-streams (RNGS) (e.g., MRG32k3a) to distinct processes and synchronizing RNGS across scenarios, but requires careful design to avoid unintended correlations and maintain event-level alignment across scenarios.

This tutorial provides an open-source and transparent framework for stochastic simulation of individual, state-transition models in continuous time via an event-driven approach. It explains the key mathematical concepts and provides a detailed guide to perform a CEA from first principles, generate epidemiological outcomes, and conduct a probabilistic analysis, including EVPI, required by CEA panels and multiple HTA agencies. Decision scientists using modeling approaches like DES can benefit from understanding the mathematical concepts, coding techniques, and design that our framework illustrates, regardless of whether they use specialized software. The concepts and techniques in DES are closely related to those in simulating cSTMs and iSTMs in discrete time, with significant gains in computational efficiency and flexibility in modeling realistic dynamics.

References

[1] J. Caro, J. Möller, J. Karnon, J. Stahl, and J. Ishak, *Discrete Event Simulation for Health Technology Assessment*, 1st edition. Erscheinungsort nicht ermittelbar: Chapman and Hall/CRC, 2015.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-72274-6.

[3] J. Karnon, J. Stahl, A. Brennan, J. J. Caro, J. Mar, and J. Möller, "Modeling Using Discrete Event Simulation: A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force–4," *Med. Decis. Making*, vol. 32, no. 5, pp. 701–711, Sept. 2012, doi: 10.1177/0272989X12455462.

[4] K. Degeling, M. J. IJzerman, M. Koopman, and H. Koffijberg, "Accounting for parameter uncertainty in the definition of parametric distributions used to describe individual patient variation in health economic models," *BMC Med. Res. Methodol.*, vol. 17, no. 1, p. 170, Dec. 2017, doi: 10.1186/s12874-017-0437-y.

[5] A. Arrospide, O. Ibarrondo, I. Castilla, I. Larrañaga, and J. Mar, "Development and Validation of a Discrete Event Simulation Model to Evaluate the Cardiovascular Impact of Population Policies for Obesity," *Med. Decis. Making*, vol. 42, no. 2, pp. 241–254, Feb. 2022, doi: 10.1177/0272989X211032964.

[6] L. Masucci *et al.*, "CAR T-cell Therapy for Diffuse Large B-cell Lymphoma in Canada: A Cost-Utility Analysis," *Med. Decis. Making*, vol. 44, no. 3, pp. 296–306, 2024, doi: 10.1177/0272989X241234070.

[7] M. J. Glover *et al.*, "Discrete Event Simulation for Decision Modeling in Health Care: Lessons from Abdominal Aortic Aneurysm Screening," *Med. Decis. Making*, vol. 38, no. 4, pp. 439–451, May 2018, doi: 10.1177/0272989X17753380.

[8] M. Sadatsafavi *et al.*, "Development and Validation of the Evaluation Platform in COPD (EPIC): A Population-Based Outcomes Model of COPD for Canada," *Med. Decis. Making*, vol. 39, no. 2, pp. 152–167, Feb. 2019, doi: 10.1177/0272989X18824098.

[9] J. J. Van Den Broek, N. T. Van Ravesteyn, E. A. Heijnsdijk, and H. J. De Koning, "Simulating the Impact of Risk-Based Screening and Treatment on Breast Cancer Outcomes with MISCAN-Fadia," *Med. Decis. Making*, vol. 38, no. 1_suppl, pp. 54S-65S, Apr. 2018, doi: 10.1177/0272989X17711928.

[10] H. Jalal, P. Pechlivanoglou, E. Krijkamp, F. Alarid-Escudero, E. Enns, and M. G. M. Hunink, "An Overview of R in Health Decision Sciences," *Med. Decis. Making*, vol. 37, no. 7, pp. 735–746, Oct. 2017, doi: 10.1177/0272989X16686559.

[11] F. Alarid-Escudero *et al.*, "An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example," *Med. Decis. Making*, vol. 43, no. 1, pp. 3–20, Jan. 2023, doi: 10.1177/0272989X221103163.

[12] F. Alarid-Escudero *et al.*, "A Tutorial on Time-Dependent Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example," *Med. Decis. Making*, vol. 43, no. 1, pp. 21–41, Jan. 2023, doi: 10.1177/0272989X221121747.

[13] E. M. Krijkamp, F. Alarid-Escudero, E. A. Enns, H. J. Jalal, M. G. M. Hunink, and P. Pechlivanoglou, "Microsimulation Modeling for Health Decision Sciences Using R: A Tutorial," *Med. Decis. Making*, vol. 38, no. 3, pp. 400–422, Apr. 2018, doi: 10.1177/0272989X18754513.

[14] E. T. Lee and J. W. Wang, *Statistical methods for survival data analysis*, Third edition. in Wiley series in probability and statistics. Hoboken, N.J: Wiley, 2003. doi: 10.1002/0471458546.

[15] N. J. Welton and A. E. Ades, "Estimation of Markov Chain Transition Probabilities and Rates from Fully and Partially Observed Data: Uncertainty Propagation, Evidence Synthesis, and Model Calibration," *Med. Decis. Making*, vol. 25, no. 6, pp. 633–645, Nov. 2005, doi: 10.1177/0272989X05282637.

[16] C. Jackson, "**flexsurv** : A Platform for Parametric Survival Modeling in *R*," *J. Stat. Softw.*, vol. 70, no. 8, 2016, doi: 10.18637/jss.v070.i08.
[17] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions*, 2nd ed. in Wiley series in probability and mathematical statistics. New York: Wiley, 1994.
[18] T. A. Trikalinos and Y. Sereda, "The nhppp package for simulating non-homogeneous Poisson point processes in R," *PLOS ONE*, vol. 19, no. 11, p. e0311311, Nov. 2024, doi: 10.1371/journal.pone.0311311.
[19] D. U. Garibay-Treviño, H. Jalal, and F. Alarid-Escudero, "A Fast Nonparametric Sampling Method for Time to Event in Individual-Level Simulation Models," *Med. Decis. Making*, vol. 45, no. 2, pp. 205–213, Feb. 2025, doi: 10.1177/0272989X241308768.
[20] A. M. Law, *Simulation Modeling and Analysis*, Fifth Edition. New York: McGraw-Hill US Higher Ed USE Legacy, 2015.
[21] D. T. Gillespie, "Stochastic Simulation of Chemical Kinetics," *Annu. Rev. Phys. Chem.*, vol. 58, no. 1, pp. 35–55, May 2007, doi: 10.1146/annurev.physchem.58.032806.104637.
[22] K. Degeling, H. Koffijberg, M. D. Franken, M. Koopman, and M. J. IJzerman, "Comparing Strategies for Modeling Competing Risks in Discrete-Event Simulations: A Simulation Study and Illustration in Colorectal Cancer," *Med. Decis. Making*, vol. 39, no. 1, pp. 57–73, Jan. 2019, doi: 10.1177/0272989X18814770.
[23] D. F. Anderson, "A modified next reaction method for simulating chemical systems with time dependent propensities and delays," *J. Chem. Phys.*, vol. 127, no. 21, p. 214107, Dec. 2007, doi: 10.1063/1.2799998.
[24] M. Boguñá, L. F. Lafuerza, R. Toral, and M. Á. Serrano, "Simulating non-Markovian stochastic processes," *Phys. Rev. E*, vol. 90, no. 4, p. 042108, Oct. 2014, doi: 10.1103/PhysRevE.90.042108.
[25] J. Dong and O. Perry, "Queueing Models for Patient-Flow Dynamics in Inpatient Wards," *Oper. Res.*, vol. 68, no. 1, pp. 250–275, Jan. 2020, doi: 10.1287/opre.2019.1845.
[26] X. Wu, J. Li, and M. T. Khasawneh, "Rule-based task assignment and scheduling of medical examinations for heterogeneous MRI machines," *J. Simul.*, vol. 14, no. 3, pp. 189–203, July 2020, doi: 10.1080/17477778.2019.1623988.
[27] E. A. Enns, L. E. Cipriano, C. T. Simons, and C. Y. Kong, "Identifying Best-Fitting Inputs in Health-Economic Model Calibration: A Pareto Frontier Approach," *Med. Decis. Making*, vol. 35, no. 2, pp. 170–182, Feb. 2015, doi: 10.1177/0272989X14528382.
[28] F. Alarid-Escudero *et al.*, "A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling," *PharmacoEconomics*, vol. 37, no. 11, pp. 1329–1339, Nov. 2019, doi: 10.1007/s40273-019-00837-x.
[29] T. Barrett *et al.*, "data.table: Extension of `data.frame`." p. 1.17.0, Apr. 15, 2006. doi: 10.32614/CRAN.package.data.table.
[30] P. Glasserman and D. D. Yao, "Some Guidelines and Guarantees for Common Random Numbers," *Manag. Sci.*, vol. 38, no. 6, pp. 884–908, June 1992, doi: 10.1287/mnsc.38.6.884.
[31] M. D. McKay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, p. 239, May 1979, doi: 10.2307/1268522.
[32] J. Worthington *et al.*, "Evaluating Semi-Markov Processes and Other Epidemiological Time-to-Event Models by Computing Disease Sojourn Density as Partial Differential Equations," *Med. Decis. Making*, vol. 45, no. 5, pp. 569–586, July 2025, doi: 10.1177/0272989X251333398.
[33] P. K. Andersen and N. Keiding, "Multi-state models for event history analysis," *Stat. Methods Med. Res.*, vol. 11, no. 2, pp. 91–115, Apr. 2002, doi: 10.1191/0962280202SM276ra.
[34] K. Degeling, J. Karnon, M. Van De Ven, A. Brennan, and H. Koffijberg, "Discrete Event Simulation in R using the 'Simmer' Package for Health Economic Modelling: A Tutorial and Illustration in Colon Cancer," *Appl. Health Econ. Health Policy.*, vol. 23, no. 6, pp. 961–975, Nov. 2025, doi: 10.1007/s40258-025-00983-8.

[35] I. Ucar, B. Smeets, and A. Azcorra, "**simmer** : Discrete-Event Simulation for *R*," *J. Stat. Softw.*, vol. 90, no. 2, 2019, doi: 10.18637/jss.v090.i02.
[36] A. Ebert, P. Wu, K. Mengersen, and F. Ruggeri, "Computationally Efficient Simulation of Queues: The *R* Package **queuecomputer**," *J. Stat. Softw.*, vol. 95, no. 5, 2020, doi: 10.18637/jss.v095.i05.
[37] M. Clements, A. Jauhiainen, and A. Karlsson, *Microsimulation package for R*. (2021). R. [Online]. Available: https://github.com/mclements/microsimulation
[38] D. Incerti and J. P. Jansen, "hesim: Health Economic Simulation Modeling and Decision Analysis," Mar. 09, 2021, *arXiv*: arXiv:2102.09437. doi: 10.48550/arXiv.2102.09437.
[39] J. D. Goldhaber-Fiebert, H. Jalal, and F. Alarid-Escudero, "Microsimulation Estimates of Decision Uncertainty and Value of Information Are Biased but Consistent," *Med. Decis. Making*, vol. 45, no. 2, pp. 127–142, Feb. 2025, doi: 10.1177/0272989X241305414.
[40] H. Wickham, R. Francois, L. Herny, K. Müller, and D. Vaughan, *dplyr: A Grammar of Data Manipulation*. (2025). [Online]. Available: https://CRAN.R-project.org/package=dplyr

Tables

Table 1: Description of parameters, their R variable name, base-case value and distribution

Parameter	R name	Base-case	Distribution
Names of health states	`v_names_states`	H, S1, S2, D	-
Annual discount rate for costs	`d_c`	3%	-
Annual discount rate for QALYs	`d_e`	3%	-
Number of PSA samples (K)	`n_sim`	1,000	-
Annual transition rates			
- Disease onset (H to S1)	`r_HS1`	0.15	gamma(30, 200)
- Recovery (S1 to H)	`r_S1H`	0.5	gamma(60, 120)
- Time-dependent disease progression (S1 to S2)	-		
Weibull parameters			
Scale (λ)	`r_S1S2_scale`	0.08	MVLN($\mu = (0.08,1.10)$,
Shape (γ)	`r_S1S2_shape`	1.10	$\sigma = (0.02,0.05)$, $\rho = 0.5$)
Annual mortality			
- Age-dependent background mortality rates (H to D)	`v_r_HDage`	age specific	-
- Hazard ratio of death in S1 vs H	`hr_S1`	3.0	lognormal(log(3.0), 0.01)
- Hazard ratio of death in S2 vs H	`hr_S2`	10.0	lognormal(log(10.0), 0.02)
Annual costs			
- Healthy individuals	`c_H`	$2,000	gamma(100.0, 20.0)
- Sick individuals in S1	`c_S1`	$4,000	gamma(177.8, 22.5)
- Sick individuals in S2	`c_S2`	$15,000	gamma(225.0, 66.7)
- Dead individuals	`c_D`	$0	-
Utility weights			
- Healthy individuals	`u_H`	1.00	beta(200, 3)
- Sick individuals in S1	`u_S1`	0.75	beta(130, 45)
- Sick individuals in S2	`u_S2`	0.50	beta(230, 230)
- Dead individuals	`u_D`	0.00	-
Treatment A: cost and effectiveness			

Parameter	R name	Base-case	Distribution
- Cost of Treatment A, additional to state-specific health care costs	`c_trtA`	$12,000	gamma(73.5, 163.3)
- Utility for treated individuals in S1	`u_trtA`	0.95	beta(300, 15)
Treatment B: cost and effectiveness			
- Cost of Treatment B, additional to state-specific health care costs	`c_trtB`	$12,000	gamma(86.2, 150.8)
- Reduction in rate of disease progression (S1 to S2) as hazard ratio (HR)	`hr_S1S2_trtB`	log(0.6)	lognormal(log(0.6), 0.02)
Transition rewards			
- Utility decrement of healthy individuals	`du_HS1`	0.01	beta(11,1088)
when transitioning to S1			
- Cost of healthy individuals	`ic_HS1`	$1,000	gamma(25, 40)
when transitioning to S1			
- Cost of dying when transitioning to D	`ic_D`	$2,000	gamma(100, 20)

Table 2: Structure and Tasks of Submodules in Module 2

$Submodule_i$: Determine Transitions from the ith State
Task1: Identify possible transitions out of current state and their corresponding intensity functions. Expand rows in current `dt_subset` to store latent event times of each possible transitions for each person.
Task2: Sample latent event times for transitions out of current state following any of the routines described in Section 3.1.
Task3: Determine next event and event time for each person following the algorithm described in Section 3.2.
Task4: Update `dt_main` with new current state and time for each person, and append rows to `dt_event_history`.

Figures

Figure 1: States Transition Diagram of a Model with three States and Example of an Individual Trajectory of three Events.
Figure 2: Schematic and Peseudocode of Simulation Algorithm
Figure 3: Epidemiological Outcomes
Figure 4: PSA-based Economic Outcomes
Figure 5: Median State Occupancy over Time by Simulation Sample Size

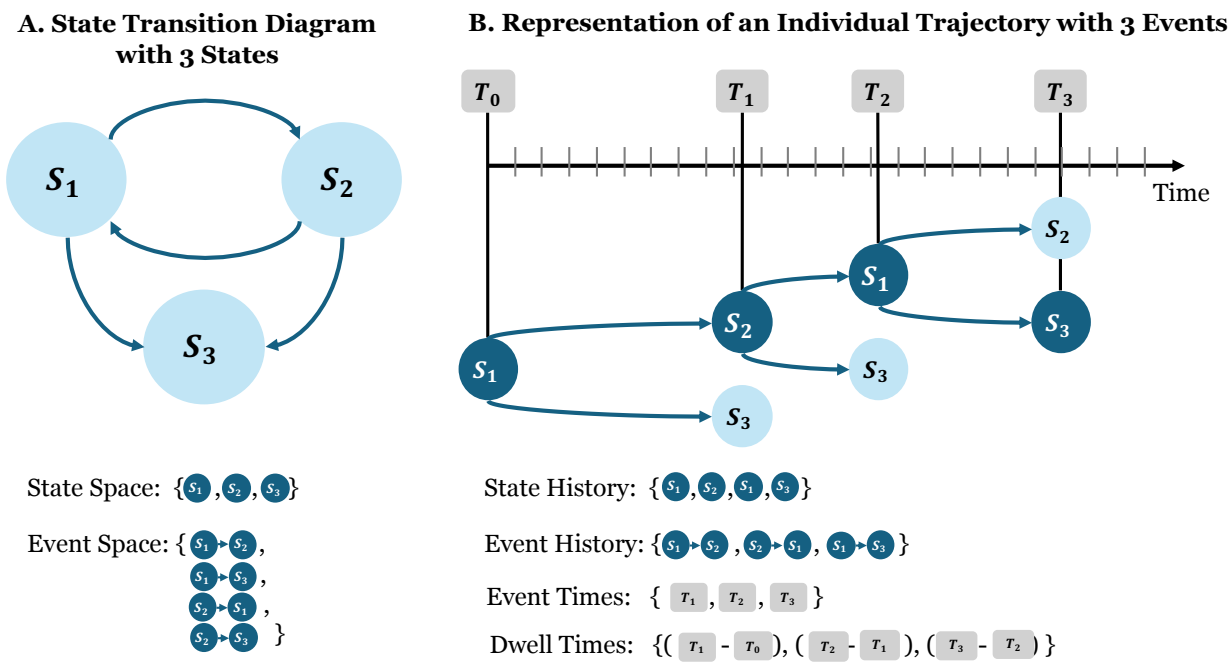

Figure 1: State-Transition Diagram of a Model with three States and Example of an Individual Trajectory of three Events.

Panel A. State-transition diagram with three states. State space refers to the set of mutually exclusive and collectively exhaustive states that a simulated person can visit during the simulation. Event space refers to the set of unique transitions between pairs of states that a simulated person can experience during the simulation. Panel B. Representation of an individual trajectory involving the occurrence of three events. State history refers to the previosuly occupied health states. Event history refers to the previosly experienced transitions between health states. Event times are the time points at which transitions between health states occured. Dwell times refer to the elapsed time in a given state before transitioning to another state.

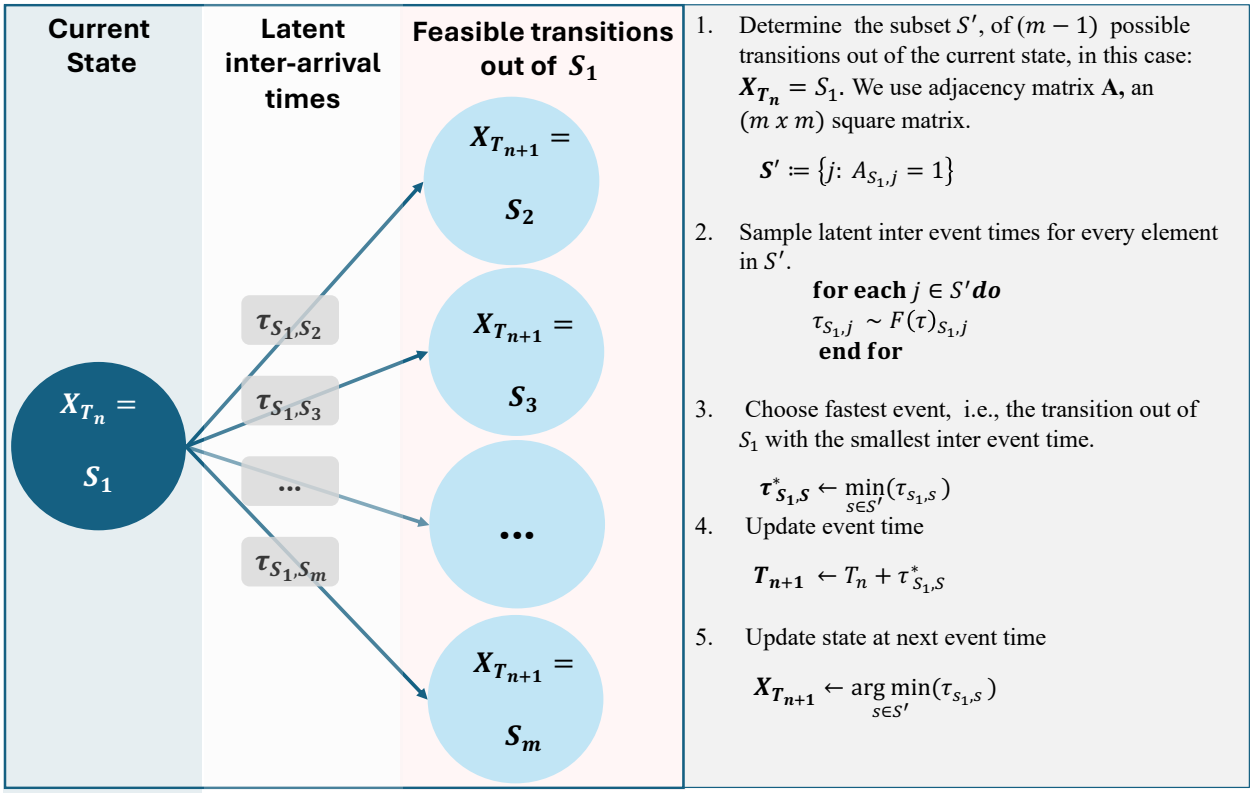

Figure 2: Schematic and Peseudocode of Simulation Algorithm

Left. Diagram of $(m-1)$ feasible transitions out of S_1. Right. Pseudocode for determining the next transition and the inter-event time following the next reaction method.

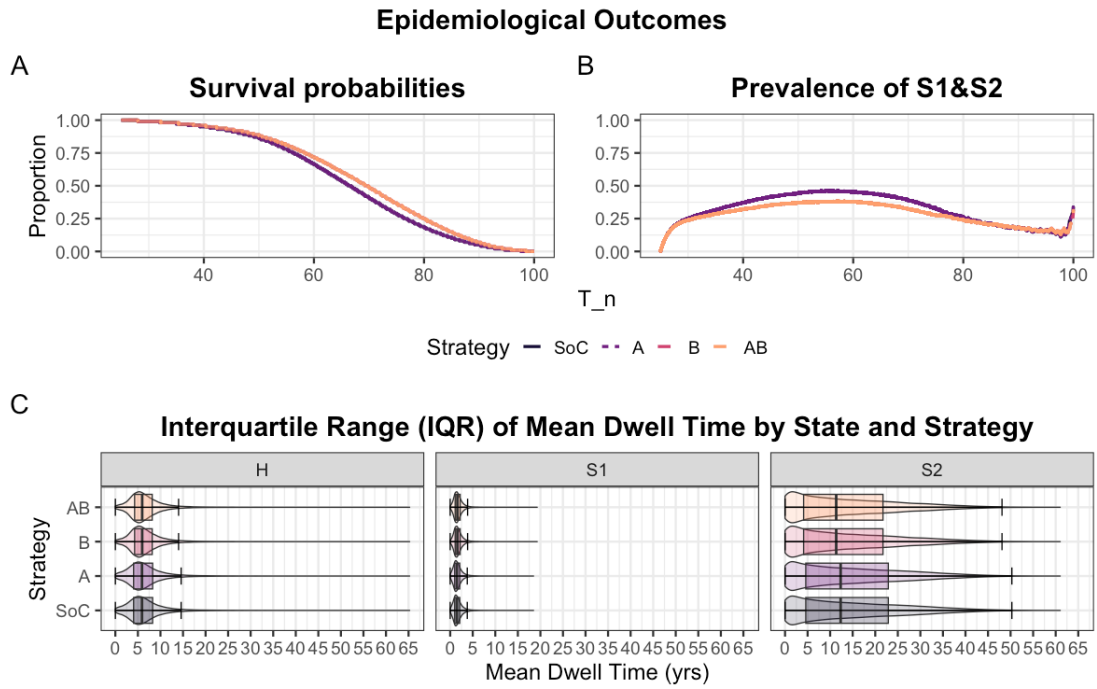

Figure 3: Epidemiological Outcomes

Panel A.Comparison of survival probabilities under each Strategy. Strategies "SoC" and "A" do not affect transition rates hence probabilities of surviving up to time t are unaffected. Strategies "B"and "AB" reduce the rate of transition from Sick to Sicker states increasing the probabilities of surviving up to time t. Panel B. Comparison of prevalence curves under each strategy. Strategies "SoC" and "A" do not affect transition rates hence probabilities of being in a Sick state (Sick or Sicker) at time t are unaffected. Strategies "B" and "AB" affect transition from Sick to Sicker states reducing the probabilities in a Sick state (Sick or Sicker) at time t. Panel C. Comparison of the interquartile range of the distribution of individual mean dwell times in each alive state under each strategy . Strategies "B" "and"AB" reduce transition from Sick to Sicker states reducing the individual median dwell time in the Sicker state compared to strategies "SoC" and "A" that do not affect transitions between health states.

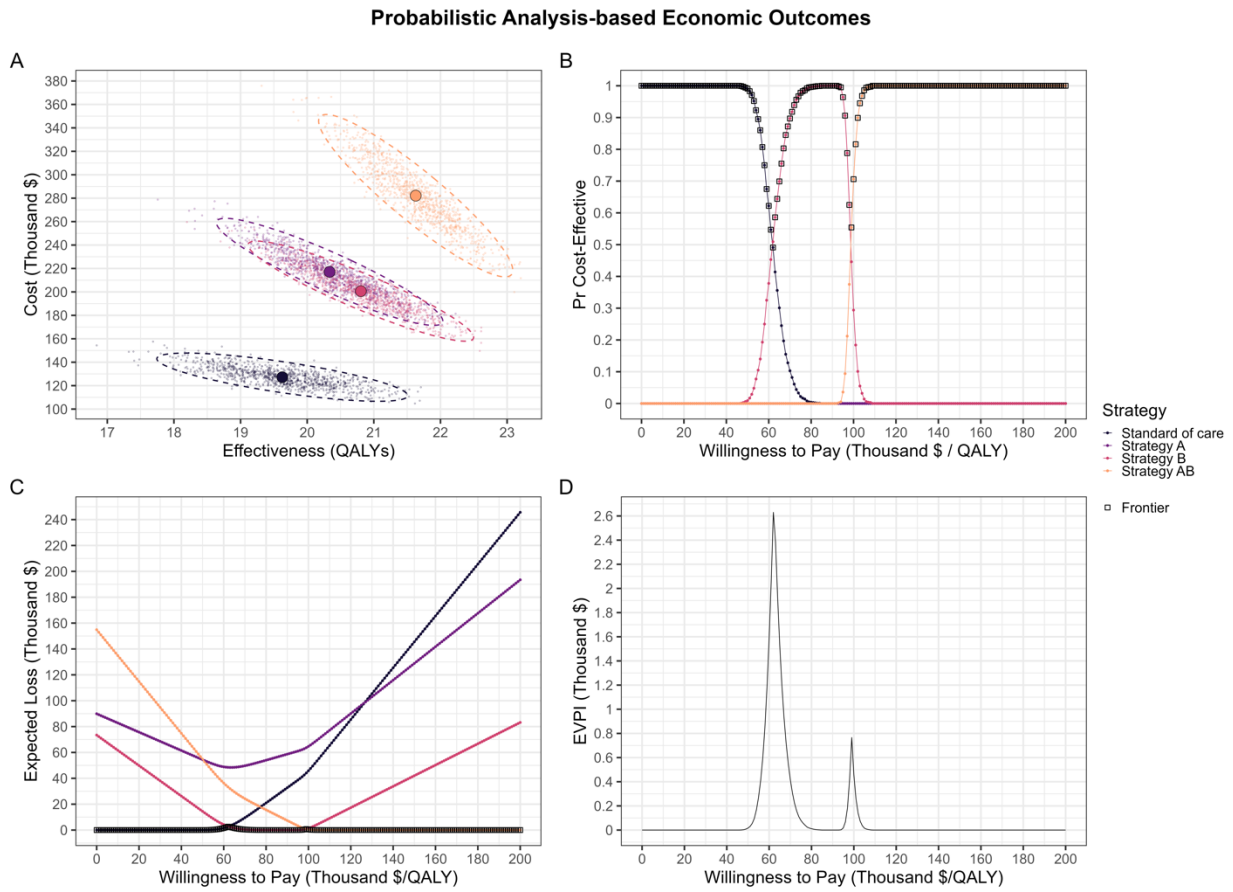

Figure 4: Probabilistic Analysis-based Economic Outcomes

Panel A. Cost-effectiveness scatter plot shows pairs of expected costs and QALYs under each strategy for each parameter set. Panel B. Cost-effectiveness acceptability curves (CEACs) represent the probability that each strategy is cost-effective at each willingness to pay (WTP) threshold, and the cost-effectiveness frontier (CEAF), representing the probability of the optimal strategy being cost-effective at each WTP threshold ($1,000 intervals). Panel C. Expected loss curves (ELCs) represent the foregone benefits of choosing a suboptimal strategy at each WTP threshold. Panel D. Expected value of perfect information (EVPI) represent the maximum amount a decision-maker is willing to pay to obtain perfect information on all parameters of the decision model.

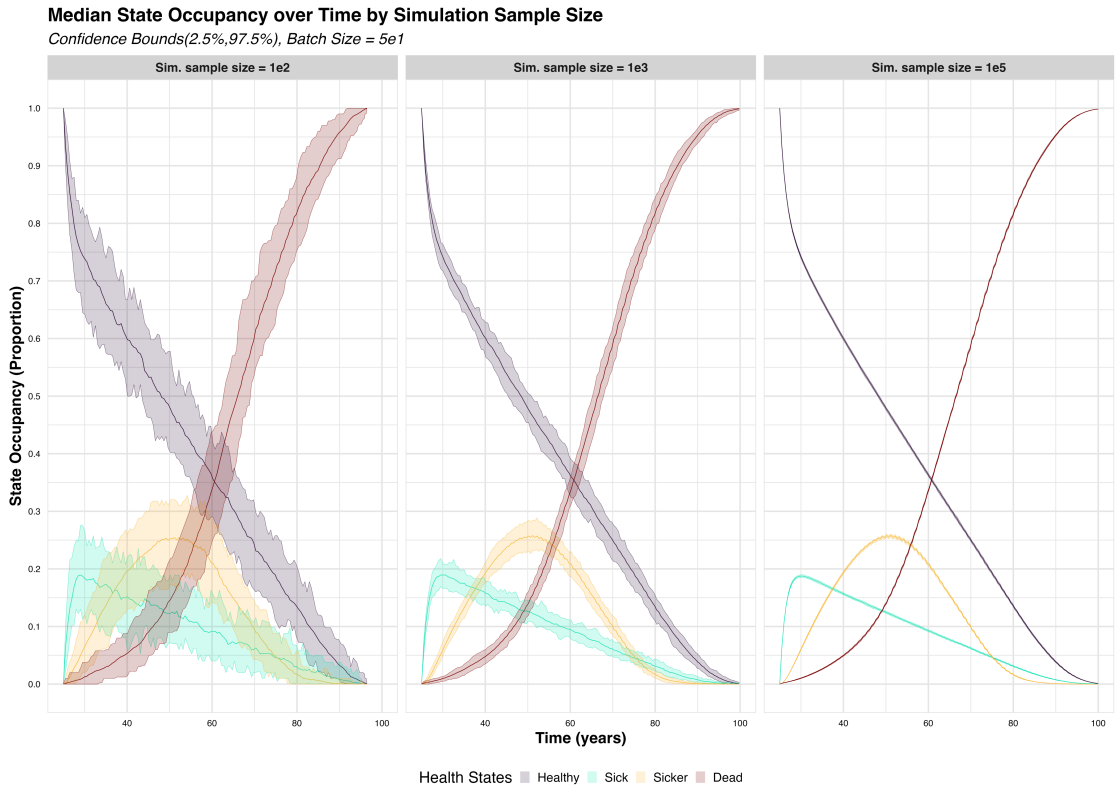

Figure 5: Median State Occupancy over Time by Simulation Sample Size

State occupancy is calculated as the proportion of people in a given health state at a given time. We compute median, 2.5%, and 97.5% percentiles of state occupancy estimates by running fixed-sized batches of 5e1 independent runs of the simulation for cohorts of different sample sizes: 1e2, 1e3, 1e5. The variability of the Monte Carlo estimate of state occupancy decreases as we increase the simulation sample size.