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

Mauricio Lopez-Mendez, PhD* Jeremy D. Goldhaber-Fiebert, PhD†
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 sensitivity 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.

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

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

1 Introduction

Discrete event simulation (DES) is a flexible, general framework for simulating dynamic systems stochastically (Caro et al., 2015; Cassandras & Lafortune, 2021; Degeling et al., 2019; Karnon et al., 2012). The salient feature is that the variables of interest are represented by "events" and the state of the system is only updated when events occur. The DES framework enables the representation of dynamic processes that evolve over continuous or discrete time, and the inclusion of multiple sources of within- and between-person variation (Degeling et al., 2017), as well as different types of dependencies on time and previous states of the system (i.e., event history).

DES models apply to many modeling situations as they are highly general. For instance, 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 disease transmission (e.g., Arrospide et al., 2022; Masucci et al., 2024). They are particularly valuable for modeling complex dynamics that can inform medical, pharmacological, and healthcare decision-making (e.g., Glover et al., 2018; Sadatsafavi et al., 2019; Van Den Broek et al., 2018).

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 pseudo random 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. 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 extend the DES to represent more complex dynamics that 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 DES implementation in the R programming language (Jalal et al., 2017; R-Core-Team, 2022) with the code to generate economic and epidemiological outcomes of interest in CEA and HTA. Readers of the tutorial 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, e.g. real numbers, or discrete, e.g. natural numbers (Cassandras & Lafortune, 2021). Mathematically, a DES corresponds to a continuous time stochastic process over a discrete, finite event space. Compared to discrete time simulation models (Alarid-Escudero et al., 2023b; Krijkamp et al., 2018), continuous time simulation requires a different set of mathematical tools to analyze the simulated individual's dynamics and behaviors.

2.1 Event-driven Dynamics

The evolution of a stochastic process with a discrete, finite state space can be described by the occurrence of "events" over time.

- A change of state, i.e., a change in the discrete values of a variable X_t, is denoted as an "event", and the set of all possible events is denoted as the "event-space". In the context of state-transition models, we can think of the set of unique transitions between states that can occur as the set of all possible events.
- Let the random variable T_n denote the nth event time ($n = 1, 2, ... < \infty$), i.e., the time at which the nth change of state occurs. An example of an event time would be the recorded time at which a patient transitions from being healthy to becoming sick.
- Inter-event time (also referred to as 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 itself with an associated cumulative distribution function (CDF), $F(\tau)_{r,s}$, where (r) indexes the origin state (value of X_t immediately prior to the nth event) and (s) indexes the destination state (value of X_t after the nth event). We index the distributions of inter-event times by the origin and destination states to highlight that the passage of time between pairs of distinct states may follow different distributions.

Figure 1 illustrates these concepts for a simple three-state iSTM for one simulated person. At time zero, T_0, the person starts in state S_1. Note that state S_3 is absorbing, i.e., once one reaches S_3 one cannot transition to any other state. Each transition is considered an event, for example the first transitions from state S_1 to state S_2 is the first event. The four possible 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. Suppose that the sequence of states of a single realization of the 3-state model is the following: (S_1, S_2, S_1, S_3). Discrete time

state-transition models would evaluate the state a person occupies at all points on a finite grid of time points, for example, requiring evaluations $X_{t=0}, X_{t=1}, ..., X_{t=k}; k < \infty$. Equivalently, we can encode the same information as a sequence of events, in this case transitions between states: $(S_1 \rightarrow S_2,\ S_2 \rightarrow S_1,\ S_1 \rightarrow S_3)$. Note that 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 $X_{t=T_1}, X_{t=T_2}, X_{t=T_3}$.

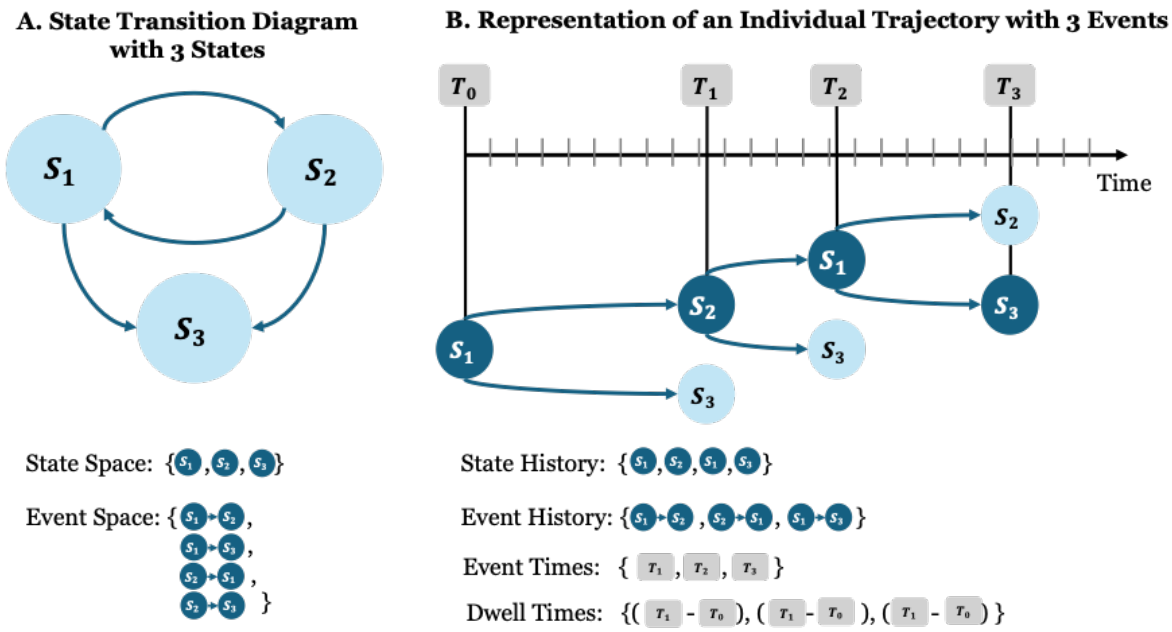

Figure 1: Panel A. State transition diagram with three States. State space refers to the set of mutually exclusive and collectively exhaustive states that a simulated person can visit during the simulation. Event space refers to the set of unique transitions between pairs of states that a simulated person 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.

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 and epidemiology) (Lee & Wang, 2003). In the context of state-transition models, the intensity function describes the rate or instantaneous probability at which a person would

transition from a state r, to a state s, $r \neq s$ (Jackson, 2016; Welton & Ades, 2005).

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

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

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 or infinitesimal probability ($\delta \to 0$) of transitioning to the next state, $X_{t+\delta} = s$, is represented by the intensity function $\lambda_{r,s}(t, \boldsymbol{z}(t), \mathcal{H}_t)$:

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

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

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

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

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, it is convenient to use validated routines 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., Johnson et al., 1994; Trikalinos & Sereda, 2024) or the non-parametric approach (Garibay-Treviño et al., 2025). In particular, we use the inversion method to sample inter-event times governed by intensity functions that depend on the time spent in the origin state, and the non-parametric approach to sample age-dependent mortality.

3.2 Simulation Algorithm (Next-reaction Method)

There are at least two approaches to simulate realizations of the stochastic process underlying a DES in continuous time. 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. These algorithms require knowledge of the set of transition-specific intensity functions between every state, which permits sampling of the latent times to 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 knowing the time of the next event determining the type of event that occurs.

These two computational approaches are extension of the next reaction and direct methods proposed by Gillespie (Gillespie, 2007) in the context of exact stochastic simulation of well-stirred chemical reactions with constant intensity functions. Our DES implementation is based specifically on Anderson's modified next reaction method (Anderson, 2007) and Boguña's adaptation for general non-exponential inter-event times (Boguñá et al., 2014).

The computational algorithm we use to evolve the process, i.e., determine for each person at each event time which transition will occur next, makes use of the set of 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 are the direct transitions that 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 while being in a given state X_{T_n}. This can be done with 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 can use the cumulative intensity functions $\Lambda(t)_{r=s_1,s}$ and the corresponding inverse cumulative intensity functions $\Lambda(t)^{-}_{s_1,s}$ to 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). In the three-state example (Figure 1), we would sample the time from S1 to S2 and the time from S1 to S3 to determine the first event out of S1.

Finally, we compute the minimum inter-arrival time $\tau^{*}_{s_1,s}$ among the latent inter-event times, i.e.:

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

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

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

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 update 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 the S1 state and facing $(m-1)$ possible transitions out of S1.

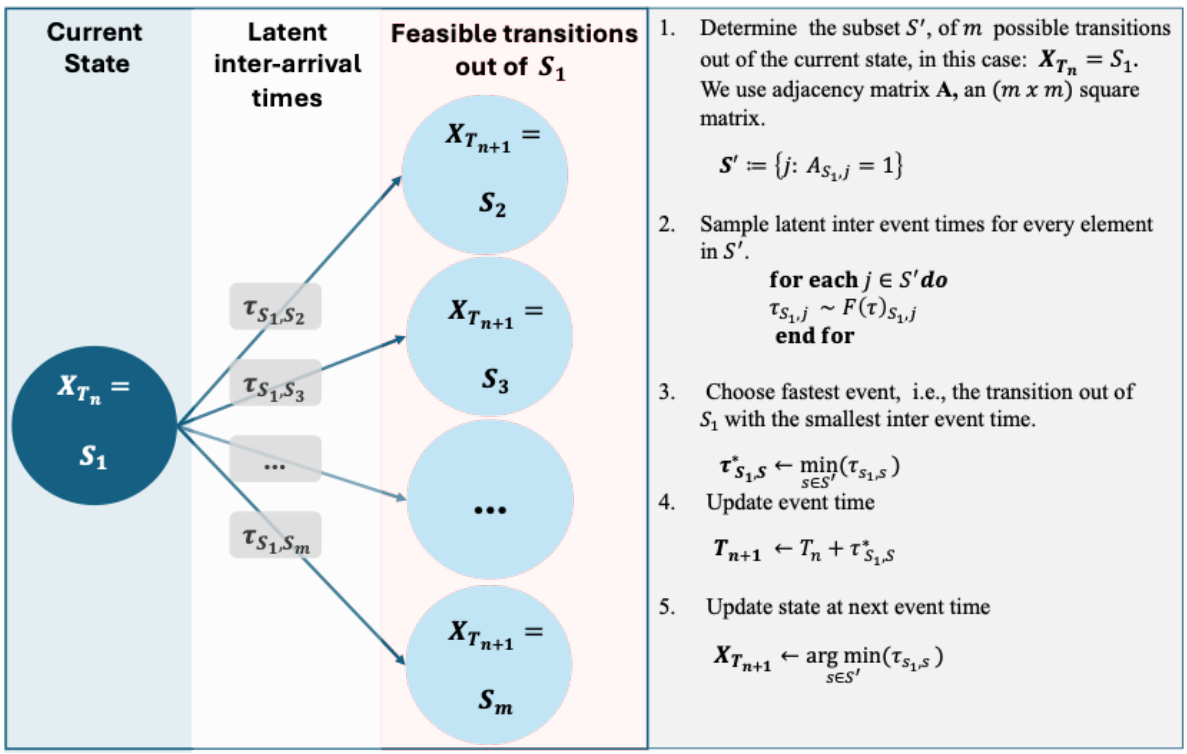

Figure 2: Left. Diagram of $(m-1)$ feasible transitions out of S_1. Right. Pseudocode for determining the next transition and the inter-event time following the next reaction method.

4 Adding Complexity to a DES

As mentioned previously, the next-reaction method for stochastic simulation of sequences of events relies on knowledge of the set of transition-specific intensity functions, $\lambda(t)_{r,s}$. We chose this approach because the intensity functions offer a direct way to incorporate time-dependence, and a flexible framework for modeling treatment effects as well as individual-level heterogeneity that act on transitions rates between states. These complexities would require specialized techniques if we were using a cSTM. For example we would need tunnel states and multidimensional transition probability arrays to incorporate time-dependence, and stratification of health states to model heterogeneity (Alarid-Escudero et al., 2023a, 2023b).

In contrast, DES does not require additional techniques to do this, and can incorporate treatment and covariate information, time dependence, and even state-history dependence in a straight-forward and transparent manner through the transition-specific intensity functions, $\lambda(t)_{r,s}$.

4.1 Time Dependence

As previously mentioned, only when the intensity functions, $\lambda(t)_{r,s}$, does not depend on time, the corresponding distribution function of the inter-event times follows an exponential distribution. 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 (e.g., Trikalinos & Sereda, 2024) and non-parametric methods (e.g., Garibay-Treviño et al., 2025) offer reliable solutions allowing us to sample from an unknown $F(\tau)_{r,s}$ via the cumulative intensity function $\Lambda(t)_{r,s}$.

4.2 Treatment and Covariate Effects

The effect of treatments and covariates (possibly time-dependent) on the transition-specific intensities can be modeled by assuming proportional hazards. We chose to follow proportional hazards for this implementation since this assumption offers an interpretable way to encode treatment and covariate effects and aligns with how these effects are typically estimated and reported in clinical trials (e.g., as hazard ratios). When the situation warrants, other formulation are also possible, e.g., the accelerated failure time formulation.

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

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

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; $z(g(t))$ is a vector of covariates (possibly time-varying) and $g(t)$ is a function of time with respect to global time (t), or with respect to the time elapsed since entering state r, (τ); (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 under this formulation, the effect of treatment enters the intensity function in the same way as the effect of a covariate would.

4.3 State-history Dependence

If the simulation keeps track of 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.

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

We demonstrate the implementation of the DES framework in R by simulating the time-dependent 4-state Sick-Sicker model (Enns et al., 2015) described in previous tutorials (Alarid-Escudero et al., 2023a, 2023b; Krijkamp et al., 2018). The DES model simulates a cohort of 25-year-olds in the "Healthy" state (denoted "H") in continuous time until they reach a maximum age of 100 years.

The cohort faces 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 evolution of 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), meaning the time from H to S1, $\tau_{H,S1}$, is exponentially distributed, i.e., $\tau_{H,S1} \sim Exp(\lambda_{H,S1})$.
- Sick individuals are at risk of further progressing to a more severe disease stage, the 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. This means 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 in the form of a hazard ratio (HR) of 3 (`hr_S2`) 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"), meaning they remain there for the rest of the simulation. We use the model to compute the cohort's expected costs and quality-adjusted life-years (QALYs) over their lifetime. We discount both costs and QALYs at an annual rate of 3%.

We are interested in evaluating the cost-effectiveness of strategies: the standard of care (strategy SoC), strategy A, strategy B, and a combination of strategies A and B (strategy AB).

- Strategy A involves administering treatment A that increases 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 the advanced stage of the disease, S2, nor changes the risk of becoming sick or progressing through the sick states.
- Strategy B uses treatment B, which reduces the rate of Sick individuals progressing to Sicker by 40%, costs $\$13,000$ per year (c_trtB), and does not affect QoL. Under proportional hazards, the effect of strategy B is encoded as a covariate with corresponding 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 described in the DARTH coding framework (Alarid-Escudero et al., 2019).

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	-

Parameter	R name	Base-case	Distribution
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($\mu = (0.08, 1.10)$, $\sigma = (0.02, 0.05)$, $\rho = 0.5$)
Shape (γ)	`r_S1S2_shape`	1.10	
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			
- 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			

Parameter	R name	Base-case	Distribution
- 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 when transitioning to S1	`du_HS1`	0.01	beta(11,1088)
- Cost of healthy individuals when transitioning to S1	`ic_HS1`	$1,000	gamma(25, 40)
- Cost of dying when transitioning to D	`ic_D`	$2,000	gamma(100, 20)

5.1 Useful Data Objects

We use `data.table` (`dt`), an R object optimized for memory efficiency and computational speed for handling large data sets with millions of rows, and offers faster grouping, summarizing, and aggregation operations with fewer memory requirements than `data.frame` and `tibble` for the case example (Barrett et al., 2006). For an introduction to `data.table` and its syntax we refer the reader to open-source resources (e.g., see online *Introduction to Data.table*, 2025).

Our simulation framework uses three distinct `data.tables`:

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

Additionally, a numbered adjacency matrix will be useful to determine which are the possible transitions from each state. We call this matrix A, and label it as `m_num_adj` in the code. For the Sick-Sicker model the adjacency matrix is informally defined in the following way:

$$\mathbf{A} = \begin{array}{cccccc} & H & S1 & S2 & D \\ H & 0 & 1 & 0 & 2 \\ S1 & 3 & 0 & 4 & 5 \\ S2 & 0 & 0 & 0 & 6 \\ D & 0 & 0 & 0 & 0 \end{array} \tag{6}$$

An adjacency matrix is traditionally a square matrix that only contains values of 0 or 1, where an entry of 1 is interpreted as the transition between row and column states being possible (i.e., the states are connected in the state-transition diagram), and 0 otherwise. However, we choose to enumerate nonzero entries to easily distinguish possible transitions between health states.

5.2 Sequences of Events and Event Times

We structure the code to determine the sequence of events for each person in two sequential modules.

5.2.1 Module 1: Setup

The first module reads `dt_main`, supplied either as `dt_baseline` or `dt_next_event`, determines the set of states currently occupied by at least one simulated person, and creates subsets of the `dt_main` such that each `dt_subset` only contains the observations of individuals currently occupying the same state. All `dt_subsets` are stored in a list `l_subsets_dt_main.` For example, if we decided to initialize the simulation with 50% of people in the Healthy and 50% in the Sick states, this module reads `dt_baseline` and produces two equally sized `dt_subsets`, one with people in the Healthy state and one with people in the Sick state.

Additionally, this module reads the provided life table, `dt_bckgrd_mortality`, and produces state-specific life tables using the provided hazard ratios, in this case, producing the `dt_mortality_Sick` and `dt_mortality_Sicker` life tables.

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

```r
## Determine set of current states
v_current_state       <- unique(dt_main$from)
v_current_state_names <- paste("from", v_current_state, sep = "_")

## Subset dt_main based on origin 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)
dt_mortality_Sick[, death_rate := death_rate*hr_S1]
### Mortality from Sicker
dt_mortality_Sicker <- copy(dt_bckgrd_mortality)
dt_mortality_Sicker[, death_rate := death_rate*hr_S2]
```

5.2.2 Module 2: Transitions between Health States

This module is the core of the simulation. It is divided into k submodules, corresponding to k non-absorbing states. For our case example, the Sick-Sicker model only requires three submodules, corresponding to the Healthy, Sick and Sicker states. An advantage of our framework is that adding more submodules does not amount to substantial additional work since each submodule is standardized and follows the same four tasks (e.g., when working with models with a larger state-space). The tasks for the ith submodule are summarized in Table 2:

Table 2: Structure and Tasks of Submodules in Module 2

$Submodule_i$: Determine Transitions from the i^{th}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.

$Submodule_i$: Determine Transitions from the i^{th}State
Task4: Update `dt_main` with new current state and time for each person, and append rows to `dt_event_history`.

5.2.2.1 Submodule 1: Transitions from Healthy

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

```
# ------------------------------------------ #
# MODULE 2: Transitions between Health States
# ------------------------------------------ #
# ----------------------------------- #
# Task 1: Possible Transitions from H
# ----------------------------------- #
# 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_keys
# 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 of the routines described in Section 3.1.

- H -> S : Healthy individuals face a constant annual rate of getting sick, meaning that the time spent in Healthy before transitioning to Sick, $\tau_{H,S1}$, is exponentially distributed, i.e., $\tau_{H,S1} \sim exp(\lambda_{H,S1})$. We use the base R function `'rexp()'` to sample the latent inter-event time $\tau_{H,S1}$, which uses the inversion method to sample latent event times, as follows:

```r
# ------------------------------------ #
# 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]
  }
```

- H -> D : Healthy individuals face an age-specific background mortality rate. Here we follow the non-parametric sampling routine to sample the latent times to death from the life table `dt_bckgrd_mortality`, as described in Section 3.1. Note that here we sample times to death since the start of the simulation and not the inter-event time as with the transition from Healthy to Sick.

```r
# ------------------------------------ #
## 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
```

```r
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: With a set of samples of latent event times for all possible transitions for each simulated person in the Healthy state, we now identify the transition that will occur next for each person by following the algorithm described in Section 3.2.

```r
# ----------------------------------- #
# 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 the variables tracking the simulation time and age (required for sampling time to death consistently) for each person in the simulation. Simulation time is updated by adding the inter-event time of the transition that was predicted to occur in **Task 3** to the existing simulation time. Note that we use years as the unit of time for both event times (`T_n`) and age of simulated people (`Age`); however, `Age` is rounded to integer values to use nonparametric sampling from life tables.

```r
# ----------------------------------- #
### Task 4: Update Important Variables
# ----------------------------------- #
# Update simulation time
dt_subset_long[status == T, T_n := T_stop]
# Update age
dt_subset_long[status == T, Age := round(T_n)]
```

At the same time, we keep track of the events' histories by appending the information of the new event to `dt_event_history` in long format.

```r
# ----------------------------------- #
# 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)
```

Note the double arrow assignment `<<-`, which is needed inside a function to update `dt_event_history` outside the function environment (see Section 5.2.3).

5.2.2.2 Submodule 2: Transitions from Sick

Recall that Sick individuals are at risk of further progressing to a more severe disease stage, i.e., the Sicker health state, as a function of time spent in the Sick state. Additionally, Sick individuals face a constant rate of recovering to Healthy along with an increased age-dependent mortality while Sick.

To encode these aspects of the dynamics in the simulation, we proceed with the same structure as in the submodule dealing with transitions out of Healthy but modify **Task 2** to sample the latent inter-event times from Sick to Sicker following a Weibull distribution with parameters (`r_S1S2_scale`) and (`r_S1S2_shape`), the latent inter-event times from Sick to Healthy from an exponential distribution with rate parameter `r_S1H`, and finally sample the time to death while Sick from the corresponding life table and hazard ratio. We omit the code for Tasks other than **Task 2** since they are mostly unchanged in this submodule (the complete code is available in the Appendix titled `DES_Sick_Sicker_progressive`, and Github repository).

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

```
# ----------------------------------- #
# 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 the same non-parametric routine is used as in **Submodule 1,** but instead of using the background mortality life table, `dt_bckgrd_mortality` , we use the state-specific life table generated in **Module 1**, `dt_mortality_Sick`.

5.2.2.3 Submodule 3: Transitions from Sicker

The last submodule follows the same structure as before but with the modification that the only possible transition out of Sicker is to Dead. This means that we do not need to execute **Task 3** to determine which transition will occur since by definition of this case example, the only possible transition from the Sicker state is to the Dead state. As in **Submodule 2**, the relevant modification in **Task 2** involves the correct use of the life table generated in **Module 1** that corresponds to mortality while Sicker, `dt_mortality_Sicker`.

5.2.3 Module 3: Dealing with Recurrent Events

If the state-transition model was purely progressive, i.e. the model did not include a transition from Sick back to Healthy, the maximum number of iterations of Modules 1 and 2 that guarantee that every person in the simulation has either died or reached the 100 years threshold is a fixed quantity, i.e., three iterations corresponding to the number of transitions between states of the longest possible path: Health -> Sick -> Sicker -> D. However, allowing a transition back to Healthy implies that paths of arbitrary length are possible since we have a loop from Healthy to Sick and back to Healthy, and therefore, the maximum number of iterations to run Modules 1 and 2 (i.e., now a random variable itself) is unknown.

Two components are necessary for this module. First, we wrap Modules 1 and 2 in the function `'sim_next_event'`. Below, we only show the structure of this function making

reference to the previously mentioned Modules. See the detailed function in the Appendix titled `DES_Sick_Sicker_recurrence` and GitHub repository.

```
# ----------------------------------- #
# 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 will execute `sim_next_event` until everyone in the simulation has died or reached the follow-up limit of 100 years of age. As a sanity check, we also automate a **`warning`** in the code by specifying the maximum number of events, that when exceeded by an individual simulated trajectory would trigger a warning message.

```
# ----------------------------------- #
## 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
```

```r
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 updates dt_next_event for everyone who 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."
               ))
              }
            }
```

5.3 Post-processing and Auxiliary Modules

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

This module computes a cost-effectiveness analysis of the four strategies described above. First, we need to generate the event history, `dt_event_history`, for each of the four strategies that we are evaluating. We use a **`for`** loop that iterates over a vector containing the names of the strategies, `v_strategies`, and executes **Module 3** under each strategy.

Following best practices we include common random numbers for simulating event histories under all strategies. The set of event history tables is stored in `l_event_history_strategies`, which serves as an input for **Module A** that computes the cost-effectiveness analysis.

```r
# ---------------------------------- #
# Post-processing Module A: Compute a cost-effectiveness analysis
# ---------------------------------- #
#  Reference l_params
l_params_ref              <- l_params
# ---------------------------------- #
## Run Simulation for all Strategies
# ---------------------------------- #
v_strategies_names <- c( "SoC", "A", "B", "AB" )
#* since the effect on transition dynamics from SoC == A, and B == AB,
#* we can just execute the simulation for "SOC" and "B"
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$strategy   <- v_strategies[i]
  # Common random numbers
  set.seed(n_seed)
  # Run simulation under i^th strategy
  while(nrow(l_params$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

#  Adding 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. We append each `dt` in `l_event_history_strategies` and create a `dt` containing the four event histories, `dt_all`. We assign costs and utilities per episode, i.e. the time spent in each non-absorbing state, as well as for each treatment and for each transition between pairs of states as described in 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)]
```

To obtain total costs and benefits per event, we multiply the annual costs and utilities incurred in a given state by the time (in years) spent in each state, denoted as `tau`.

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

We apply exponential discounting of outcomes over each period of time in each state in continuous time. Let O_i be a cumulative outcome for the ith individual over a period of time $[t_1, t_2]$, and ρ be the exponential discounting rate, then the exponentially discounted cumulative outcome over period $[t_1, t_2]$ for the ith individual, $\hat{O}_i$, is:

$$\hat{O}_i = \int_{t_1}^{t_2} O_i e^{-\rho t} dt = O_i \frac{(e^{-\rho t_1} - e^{-\rho t_2})}{\rho} \tag{7}$$

For purposes of discounting, the initial time is considered to be 0. Consequently, in the case example we shift all inter-event times by subtracting the start age of the cohort (25 years of age), i.e., we subtract the time in years after birth where the simulation starts for each simulated person. The complete code and functions to run this module can be found in the Appendix titled `Module_A_CEA` and GitHub repository.

```
# ---------------------------------- #
# 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)]
```

```r
## 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)
# Formatted ICER table
df_cea_format <- format_table_cea(df_cea)
```

5.3.2 Post-processing Module B : Compute Epidemiological Outcomes

Survival probabilities and disease prevalence are epidemiological outcomes that can be obtained directly from cSTMs and DES. Another informative outcome is the dwell time in a specific health state, e.g., the time spent in the recurrence-free state in cancer models. Due to the time-to-event formulation of the DES approach, dwell times at each alive state for each individual

over the entire time horizon, as well as the distribution of dwell times for the entire cohort are directly obtained from the output of the DES.

In this module we provide functions to compute survival probabilities, prevalence of disease, and distribution of dwell times for the entire cohort at each health state for the Sick-Sicker Model. Below we only present the specific lines of the code used to compute the epidemiological outcomes. Functions for performing auxiliary computations and plotting epidemiological outcomes can be found in the Appendix title `Module_B_Epi_Outcomes` and the GitHub repository.

```
# ------------------------------------ #
# Post-processing Module B: Compute Epidemiological Outcomes
# ------------------------------------ #
# A.1 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")]
# A.2 Distribution of the average dwell time at each alive state under each strategy.
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")]

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

5.3.3 Post-processing Module C: Compute a Probabilistic Sensitivity Analysis (PSA)

The objective of this Module is to compute a probabilistic sensitivity analysis to quantify the effect of model parameter uncertainty on cost-effectiveness outcomes. First, the module generates a `dt` with sampled parameter sets following a Latin Hypercube sampling approach (McKay et al., 1979) and user-specified parameter uncertainty distributions. Then iterates over each parameter set simulating all strategies and computes cost-effectiveness outcomes corresponding to each parameter set. The economic outcomes and each parameter set are stored in a PSA data set that follows the structure presented in other tutorials (Alarid-Escudero et al., 2023a, 2023b; Krijkamp et al., 2018). The compatibility with previous tutorial has the advantage of using previously validated functions for generating samples of parameter sets, and compute economic outcomes directly from the PSA data set generated for the DES. See Appendix titled `Module_C_PSA`.

To showcase this functionality, Figure 4 shows for the CEA of the time-dependent Sick Sicker Model, the CEA 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). To generate the PSA-based results we sampled $1e3$ parameter sets.

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

In the Appendix titled `Aux_Module_Cont_Time_Trace` we include another useful module with functions that map the output of the DES into a cohort trace matrix (i.e., the proportions of simulated people in each state at times specified by a fixed grid of time points). This function enables us to quickly produce epidemiological outcomes of interest such as survival probabilities and prevalence in the same way as one would with a discrete time cSTM. Using this module, analysts can also compare the proportions at each state over time recovered from the event histories from the DES against the proportions at each state over time simulated

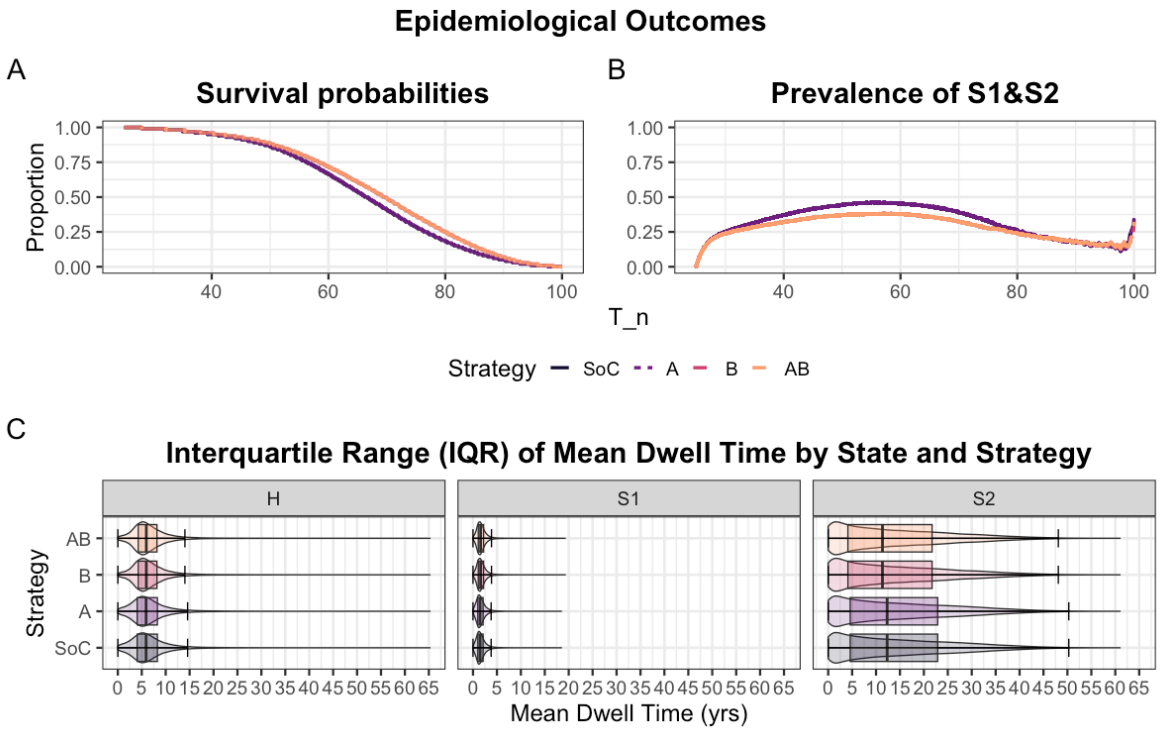

Figure 3: Panel A.Comparison of survival probabilities under each Strategy. Strategies "SoC" and "A" do not affect transition rates hence probabilities of surviving up to time t are unaffected. Strategies "B"and "AB" reduce the rate of transition from Sick to Sicker states increasing the probabilities 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.

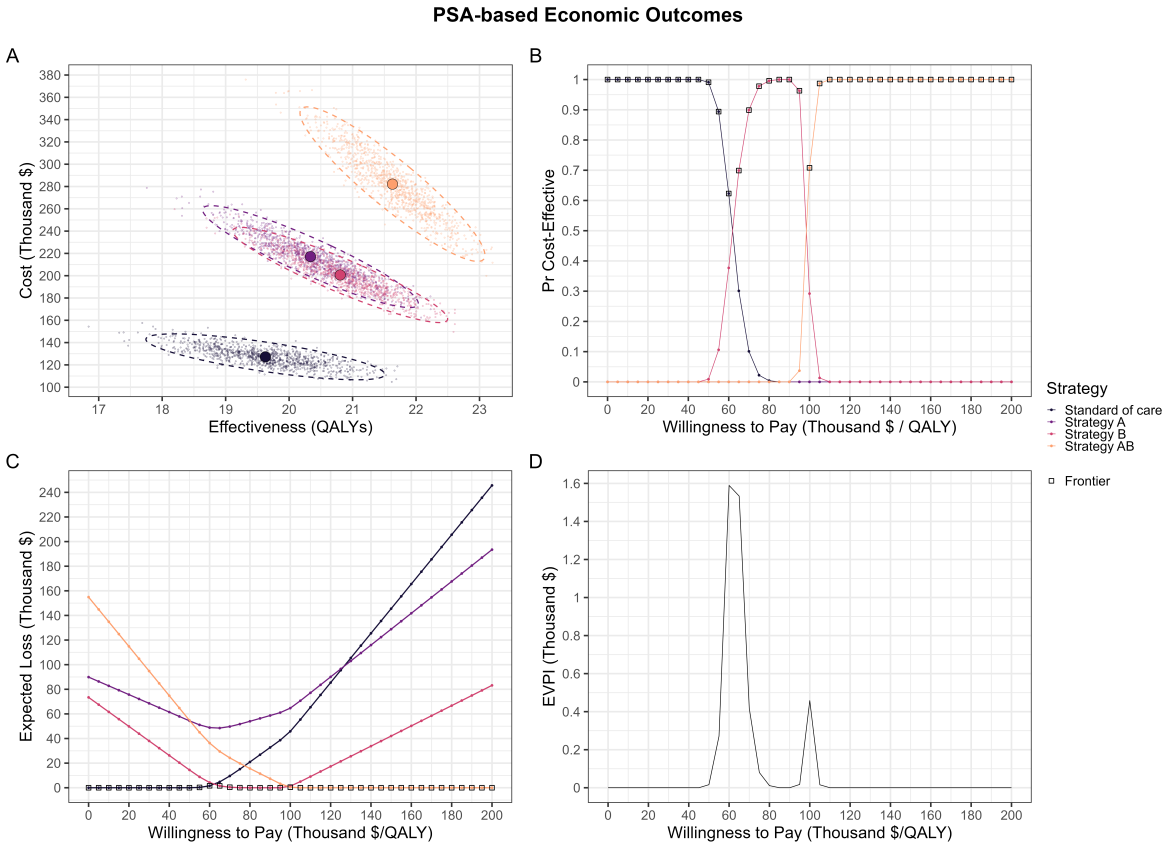

Figure 4: Panel A. Cost-effectiveness scatter plot shows pairs of expected costs and QALYs under each strategy for each parameter set. Panel B. Cost-effectiveness acceptability curves (CEACs) represent the probability that each strategy is cost-effective at each willingness to pay (WTP) threshold, and the cost-effectiveness frontier (CEAF), representing the probability of the optimal strategy being cost-effective at each WTP threshold. 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.

with discrete time approaches such as a cSTMs. We also generate the trace plots from these cohort trace matrices which can be useful to visually diagnose how the DES is representing transition dynamics over time as well as examine the convergence of the estimators of the proportions at each state over time (i.e., state occupancy over time).

5.4 Convergence of Monte Carlo Estimator

Finally, we are interested in how many people one must simulate with the DES to get estimates that converge with those made using an equivalent cSTM (mean-process model).For this case example, we consider the DES-based estimators of the proportions of people in each state (state occupancy) over time and their variability expressed as bootstrap confidence bounds at the 2.5^th and 97.5^th percentiles around the median. Figure 5 shows that the variability of the DES-based state occupancy estimates quickly declines as we increase the number of people simulated by the DES. A distributed implementation of this analysis can be found in the Appendix titled `Aux_Module_Convergence` and Github repository.

6 Discussion

In this tutorial, we present a novel, open-source implementation of DES suited for representing a realistic continuous time, individual state-transition model that may include simulation and state-residence time dependence, treatment effects and other individual-level characteristics. We highlight the versatility and benefits of our framework in the context of health-technology assessments and medical decision making by presenting the key mathematical concepts and a detailed case example of a cost-effectiveness analysis in R using the previously published Sick-Sicker model.

In our DES framework, both simulation and state-residence time dependence are seamlessly integrated into the simulation dynamics. In the case example, simulation time dependence is incorporated as age-dependent mortality using non-parametric sampling methods to sample age of death from life tables. This contrasts to other frameworks in discrete time that would require an expansion of the transition probability matrix into a three-dimensional array that tracks the transition probabilities at each cycle or requires updating transition probabilities at each cycle. Similarly, our framework offers a direct way to incorporate state-residence time dependence into the trajectories simulated by the DES by allowing fast and direct sampling of inter-event times from non-exponential distributions, e.g., via inversion methods. For example, to incorporate the dependence in the time spent in the Sick state into the intensity of transition to Sicker, an analyst need only choose the sampling distribution of the time to transition from Sick to Sicker to be a nonexponential distribution, e.g., Weibull distribution. In discrete time, state-residence time dependence would require expanding the state space with a finite number of so-called "tunnel" states. In our case example, we would need to partition the Sick state into (k) tunnel Sick states to capture the time spent at Sick before transitioning to the Sicker

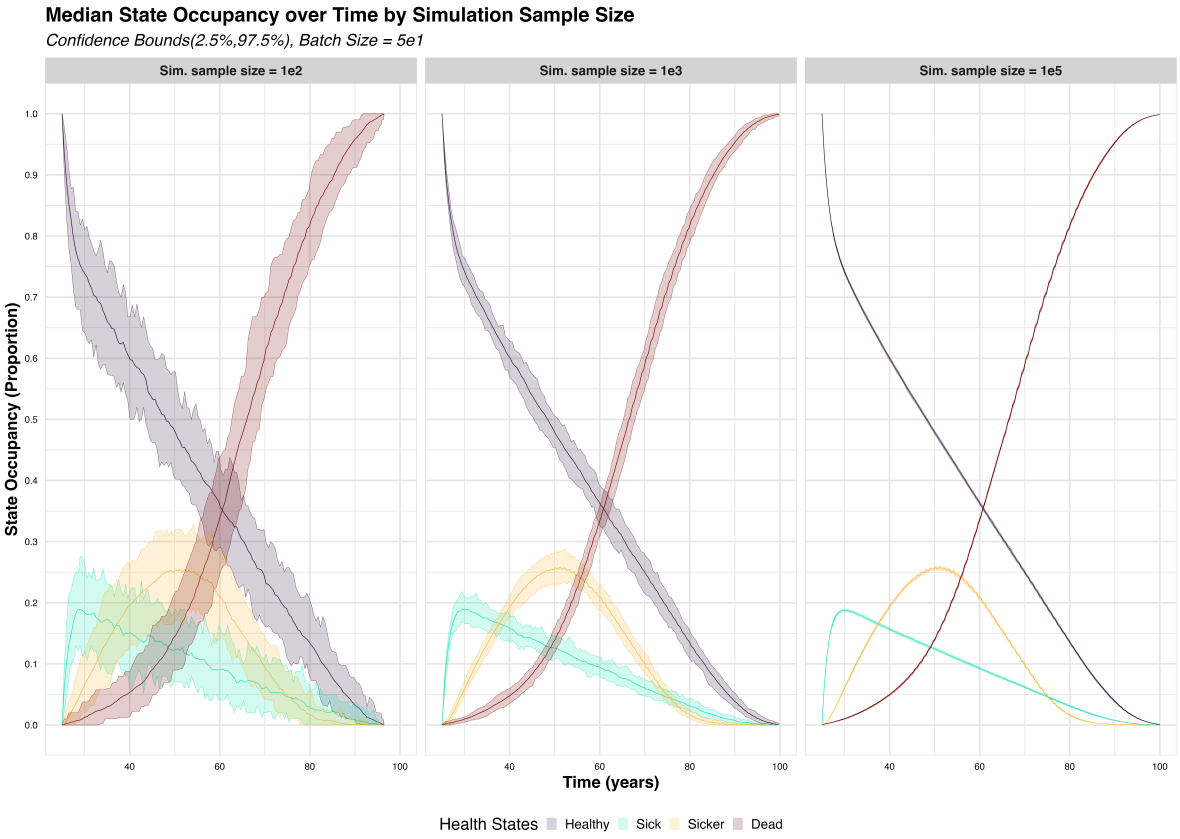

Figure 5: State occupancy is calculated as the proportion of people in a given health state at a given time. We compute median, 2.5%, and 97.5% percentiles of state occupancy estimates by running fixed-sized batches of 5e1 independent runs of the simulation for cohorts of different sample sizes: 1e2,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

state. This can impact the computational and memory requirements of the simulation. In the case of models already considering a large number of health states, adding tunnel states to account for state-residence time dependence might impose unaffordable computational costs to run the simulation.

We also illustrate the ease with which treatment effects that influence specific transitions between health states can be incorporated into the simulated transition dynamics by following a proportional hazards approach (i.e. scaling the intensity functions). Although we do not show it explicitly in the case example, we conceptualize how individual-level characteristics that influence transitions between health states can be incorporated via the intensity functions, in the same way we incorporated treatments effects. In the cSTM, adding heterogeneity involves additional computational time and resources since the assumption underlying a cSTM is that the cohort is homogeneous, and to simulate the dynamics of (J) subpopulations, it requires running the cSTM (J) times, i.e., one for each subpopulation. While other approaches in discrete time may provide the option to incorporate treatment effects and individual-level factors without requiring running the simulation (J) times, such as discrete time iSTMs, DES still provides a comparative advantage in terms of computational resources. Due to its event-driven approach in continuous time, DES only requires evaluation of the health states at the times events occur, and not at every cycle as in discrete time approaches.

Finally, another important advantage of the DES framework is that we can characterize the within- and between-person variability of simulated dwell times at each state without complex computations. Dwell times are important metrics informing medical decisions and health technology assessments. For example, in the context of cancer disease progression models that include cancer recurrence, the ability to recover the distribution of dwell times in the recurrence-free state can be a crucial aspect to evaluate the effectiveness of personalized treatments that aim to prolong the time spent in recurrence-free states for specific subpopulations. From a cSTM we can only recover the population mean dwell time in a given state. In the case of discrete time iSTMs, we can recover the distribution of dwell times across the population (i.e., between-person variability) as well as the within person variability of the dwell times; e.g., how dwell times in a given health state change over time for one simulated person. However, the computation is not straightforward, to obtain the individual dwell times from a discrete time iSTM we would need to count for each simulated person how many times they occupied a given state over the whole time-horizon and then sum the number of cycles at each instance in which they occupied that state. In contrast, we get individual dwell times directly from the DES output and can be summarized for the entire cohort in a few code lines.

There are important analytic choices that should be considered when implementing a DES. First, to correctly sample from trajectories of events, the intensity functions used to inform the DES must reflect the joint distribution of the time-to event for each potential transition at each state. This can be achieved if data tracking the time and states visited by each person in a target population is available. In this case, a multistate model of this data would provide estimates of the transition intensities considering the joint likelihood of the competing events (Degeling et al., 2019; Jackson, 2016). If instead, estimates from different studies and

populations are used to inform the different transitions, there is a risk of misrepresenting the dynamics since independent estimates might not reflect the relative rates of transitioning to competing states accurately for the population of interest, potentially biasing the estimates of the time spent in each state and consequently the estimates of epidemiological and economic outcomes. In these situations, calibration procedures to match key epidemiological outcomes of the target population are a necessity. A practical alternative would be to leverage expert opinion about the relative rates of different transitions aligned with the best available knowledge about the dynamics of the process of interest in the target population. For example, to encode relative rates of transition between disease stages and mortality that guarantee that the rates of mortality are higher once more advanced stages of a disease are reached, decision modelers can encode mortality risks that are proportional across disease stages, where the hazard ratios between transitions to the dead state are informed by expert opinion. Additionally, the probabilistic sensitivity analysis must be designed to quantify the impact that uncertainty on the encoded relative rates of transitions between competing events can have on the decisions, e.g., accounting for uncertainty around the hazard ratios that affect mortality at different states.

A second consideration is that it may become harder to interpret the effect of interventions and health technologies on the simulated outcomes as we add complexity to the disease-progression model. To illustrate this, consider the case example of the Sick-Sicker model. By incorporating time-dependence as age-dependent mortality and recurrence to the Healthy state, the interpretation of a strategy that affects a transition from Sick to Sicker states (Strategy B in the example) is not as straightforward as it would be when dealing with transition dynamics that do not change over time. In this case, the effect of Strategy B on dwell times in the alive states fluctuates over time, i.e., it is a stochastic process. In the early years of the simulation, Strategy B, which reduces the intensity to transition to Sicker, has a bigger impact in increasing the time spent in the Healthy state, but as time passes, the mortality risk from the Sick state increases as well, reducing the number of people that return to Healthy, complicating the interpretation of the impact of Strategy B on increasing the time spent in Healthy. Analysts should carefully consider when complexity in the model design is justified; expert opinion and decision-makers should inform the design choices of the model to guarantee a reasonable degree of parsimony.

Finally, iSTMs whether simulated in discrete or continuous time, like DES, may introduce additional bias in estimates of decision uncertainty and value of information. Goldhaber-Fiebert et al 2025, provide a closed-form and numerical characterization of this bias, showing that the bias is asymptotically consistent and can be reduced by using a sufficiently large number of simulated individuals(Goldhaber-Fiebert et al., 2025). Consequently, modelers and analysts implementing a DES for decision analysis should take the necessary steps to assess the required number of simulated individuals per probabilistic sample to reduce the bias in estimates of decision uncertainty and value of information to tolerably low levels.

This tutorial provides an open-source and transparent framework for stochastic simulation of individual state transition models in continuous time following an event-driven approach. It

clarifies the key mathematical ingredients and provides a detailed walkthrough to implement a cost-effectiveness analysis, generate economic and epidemiological outcomes, conduct a PSA, which is required by CEA panels and multiple HTA agencies, and produce PSA-based outcomes such as EVPI. Decision scientist should find that implementing a DES does not require additional expertise, since most mathematical concepts and coding techniques are the same or closely related to the concepts and techniques required for simulating cSTMs and iSTMs in discrete time. Yet, the computational efficiency gains and flexibility to model realistic dynamics are considerable.

References

Alarid-Escudero, F., Krijkamp, E. M., Pechlivanoglou, P., Jalal, H., Kao, S.-Y. Z., Yang, A., & Enns, E. A. (2019). A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling. *PharmacoEconomics*, *37*(11), 1329–1339. https://doi.org/10.1007/s40273-019-00837-x

Alarid-Escudero, F., Krijkamp, E., Enns, E. A., Yang, A., Hunink, M. G. M., Pechlivanoglou, P., & Jalal, H. (2023a). A Tutorial on Time-Dependent Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. *Medical Decision Making*, *43*(1), 21–41. https://doi.org/10.1177/0272989X221121747

Alarid-Escudero, F., Krijkamp, E., Enns, E. A., Yang, A., Hunink, M. G. M., Pechlivanoglou, P., & Jalal, H. (2023b). An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. *Medical Decision Making*, *43*(1), 3–20. https://doi.org/10.1177/0272989X221103163

Anderson, D. F. (2007). A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of Chemical Physics*, *127*(21), 214107. https://doi.org/10.1063/1.2799998

Arrospide, A., Ibarrondo, O., Castilla, I., Larrañaga, I., & Mar, J. (2022). Development and Validation of a Discrete Event Simulation Model to Evaluate the Cardiovascular Impact of Population Policies for Obesity. *Medical Decision Making*, *42*(2), 241–254. https://doi.org/10.1177/0272989X211032964

Barrett, T., Dowle, M., Srinivasan, A., Gorecki, J., Chirico, M., Hocking, T., Schwendinger, B., & Krylov, I. (2006). *Data.table: Extension of 'data.frame'*. https://doi.org/10.32614/CRAN.package.data.table

Boguñá, M., Lafuerza, L. F., Toral, R., & Serrano, M. Á. (2014). Simulating non-Markovian stochastic processes. *Physical Review E*, *90*(4), 042108. https://doi.org/10.1103/PhysRevE.90.042108

Caro, J., Möller, J., Karnon, J., Stahl, J., & Ishak, J. (2015). *Discrete Event Simulation for Health Technology Assessment* (1st edition). Chapman; Hall/CRC.

Cassandras, C. G., & Lafortune, S. (2021). *Introduction to Discrete Event Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-030-72274-6

Degeling, K., IJzerman, M. J., Koopman, M., & Koffijberg, H. (2017). Accounting for parameter uncertainty in the definition of parametric distributions used to describe individual patient variation in health economic models. *BMC Medical Research Methodology*, *17*(1), 170. https://doi.org/10.1186/s12874-017-0437-y

Degeling, K., Koffijberg, H., Franken, M. D., Koopman, M., & IJzerman, M. J. (2019). Comparing Strategies for Modeling Competing Risks in Discrete-Event Simulations: A Simulation Study and Illustration in Colorectal Cancer. *Medical Decision Making*, *39*(1), 57–73. https://doi.org/10.1177/0272989X18814770

Enns, E. A., Cipriano, L. E., Simons, C. T., & Kong, C. Y. (2015). Identifying Best-Fitting Inputs in Health-Economic Model Calibration: A Pareto Frontier Approach. *Medical Decision Making*, *35*(2), 170–182. https://doi.org/10.1177/0272989X14528382

Garibay-Treviño, D. U., Jalal, H., & Alarid-Escudero, F. (2025). A Fast Nonparametric Sam-

pling Method for Time to Event in Individual-Level Simulation Models. *Medical Decision Making*, *45*(2), 205–213. https://doi.org/10.1177/0272989X241308768

Gillespie, D. T. (2007). Stochastic Simulation of Chemical Kinetics. *Annual Review of Physical Chemistry*, *58*(1), 35–55. https://doi.org/10.1146/annurev.physchem.58.032806.104637

Glover, M. J., Jones, E., Masconi, K. L., Sweeting, M. J., Thompson, S. G., SWAN Collaborators, Powell, J. T., Ulug, P., & Bown, M. J. (2018). Discrete Event Simulation for Decision Modeling in Health Care: Lessons from Abdominal Aortic Aneurysm Screening. *Medical Decision Making*, *38*(4), 439–451. https://doi.org/10.1177/0272989X17753380

Goldhaber-Fiebert, J. D., Jalal, H., & Alarid-Escudero, F. (2025). Microsimulation Estimates of Decision Uncertainty and Value of Information Are Biased but Consistent. *Medical Decision Making*, *45*(2), 127–142. https://doi.org/10.1177/0272989X241305414

Introduction to data.table. (2025). CRAN. https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html

Jackson, C. (2016). **Flexsurv** : A Platform for Parametric Survival Modeling in *r*. *Journal of Statistical Software*, *70*(8). https://doi.org/10.18637/jss.v070.i08

Jalal, H., Pechlivanoglou, P., Krijkamp, E., Alarid-Escudero, F., Enns, E., & Hunink, M. G. M. (2017). An Overview of R in Health Decision Sciences. *Medical Decision Making*, *37*(7), 735–746. https://doi.org/10.1177/0272989X16686559

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1994). *Continuous univariate distributions* (2nd ed). Wiley.

Karnon, J., Stahl, J., Brennan, A., Caro, J. J., Mar, J., & Möller, J. (2012). Modeling Using Discrete Event Simulation: A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force–4. *Medical Decision Making*, *32*(5), 701–711. https://doi.org/10.1177/0272989X12455462

Krijkamp, E. M., Alarid-Escudero, F., Enns, E. A., Jalal, H. J., Hunink, M. G. M., & Pechlivanoglou, P. (2018). Microsimulation Modeling for Health Decision Sciences Using R: A Tutorial. *Medical Decision Making*, *38*(3), 400–422. https://doi.org/10.1177/0272989X18754513

Lee, E. T., & Wang, J. W. (2003). *Statistical methods for survival data analysis* (Third edition). Wiley. https://doi.org/10.1002/0471458546

Masucci, L., Tian, F., Tully, S., Feng, Z., McFarlane, T., Chan, K. K. W., & Wong, W. W. L. (2024). CAR T-cell Therapy for Diffuse Large B-cell Lymphoma in Canada: A Cost-Utility Analysis. *Medical Decision Making*, *44*(3), 296–306. https://doi.org/10.1177/0272989X241234070

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, *21*(2), 239. https://doi.org/10.2307/1268522

R-Core-Team. (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Sadatsafavi, M., Ghanbarian, S., Adibi, A., Johnson, K., FitzGerald, J. M., Flanagan, W., Bryan, S., Sin, D., & for the Canadian Respiratory Research Network. (2019). Development and Validation of the Evaluation Platform in COPD (EPIC): A Population-Based Outcomes Model of COPD for Canada. *Medical Decision Making*, *39*(2), 152–167.

https://doi.org/10.1177/0272989X18824098

Trikalinos, T. A., & Sereda, Y. (2024). The nhppp package for simulating non-homogeneous Poisson point processes in R. *PLOS ONE*, *19*(11), e0311311. https://doi.org/10.1371/journal.pone.0311311

Van Den Broek, J. J., Van Ravesteyn, N. T., Heijnsdijk, E. A., & De Koning, H. J. (2018). Simulating the Impact of Risk-Based Screening and Treatment on Breast Cancer Outcomes with MISCAN-Fadia. *Medical Decision Making*, *38*(1_suppl), 54S–65S. https://doi.org/10.1177/0272989X17711928

Welton, N. J., & Ades, A. E. (2005). Estimation of Markov Chain Transition Probabilities and Rates from Fully and Partially Observed Data: Uncertainty Propagation, Evidence Synthesis, and Model Calibration. *Medical Decision Making*, *25*(6), 633–645. https://doi.org/10.1177/0272989X05282637