A Tutorial on Time-Dependent Cohort State-Transition Models in R using a Cost-Effectiveness Analysis Example

Fernando Alarid-Escudero, PhD[*] Eline Krijkamp, MSc[†] Eva A. Enns, PhD[‡]
Alan Yang, MSc[§] Myriam G.M. Hunink, PhD[†¶] Petros Pechlivanoglou, PhD[‖]
Hawre Jalal, MD, PhD[**]

2021-08-30

Abstract

This tutorial shows how to implement time-dependent cohort state-transition models (cSTMs) to conduct cost-effectiveness analyses (CEA) in R, where transition probabilities and rewards vary by time. We account for two types of time dependency: time since the start of the simulation (simulation-time dependency) and time spent in a health state (state residence dependency). We illustrate how to conduct a CEA of multiple strategies based on a time-dependent cSTM using a previously published cSTM, including probabilistic sensitivity analyses. We also demonstrate how to compute various epidemiological outcomes of interest from the outputs generated from the cSTM, such as survival probability and disease prevalence. We demonstrate both the mathematical notation and the R code to execute the calculations. This tutorial builds upon an introductory tutorial that introduces time-independent cSTMs using a CEA example in R. We provide an up-to-date public code repository for broader implementation.

[*]Division of Public Administration, Center for Research and Teaching in Economics (CIDE), Aguascalientes, AGS, Mexico
[†]Department of Epidemiology, Erasmus University Medical Center, Rotterdam, The Netherlands
[‡]Division of Health Policy and Management, University of Minnesota School of Public Health, Minneapolis, MN, USA
[§]The Hospital for Sick Children, Toronto
[¶]Center for Health Decision Sciences, Harvard T.H. Chan School of Public Health, Boston, USA
[‖]The Hospital for Sick Children, Toronto and University of Toronto, Toronto, Ontario, Canada
[**]University of Pittsburgh, Pittsburgh, PA, USA

1 Introduction

Cohort state-transition models (cSTMs), commonly known as Markov models, are decision models that simulate disease dynamics over time. cSTMs are widely used to evaluate various health policies and clinical strategies, such as screening and surveillance programs,[1,2] diagnostic procedures,[3] disease management programs,[4] and interventions.[5,6]. The simplest cSTMs are time-independent (time-homogeneous), meaning that the transition probabilities and other model parameters remain fixed over the simulated time horizon. In an introductory tutorial, we described the implementation of time-independent cSTMs.[7] In many applications, a time-independent cSTM is limited because key parameters may vary over time. For example, background mortality changes as a cohort ages, the risk of cancer recurrence might change as a function of time since diagnosis, or the incidence of vector-borne diseases may have temporal or seasonal trends. The costs and utility of residing in a particular health state might also vary over time. For example, cancer-related health care costs and utility might depend on whether a patient is in their first year of remission or their fifth. Similarly, a person's last year of life is often their most expensive in health care spending. Allowing for flexibility in capturing time-varying dynamics is often essential in realistic models.

This tutorial expands the cSTM framework described in the introductory time-independent cSTM tutorial by allowing transition probabilities, costs, and utilities to vary over time and for one-time costs or utilities when the cohort experiences events when transitioning between states, often called transition rewards. We also demonstrate how to compute various epidemiological measures from cSTMs needed for model calibration or validation.

We distinguish between two types of time-dependency that require different approaches: (1) Simulation time-dependency, which represents the time since the start of the simulation, and (2) state residence time-dependency, representing time spent in a health state. Simulation time-dependency affects parameters that vary with time for the entire cohort in the same way. The most common example of simulation time-dependency is age-specific background mortality over time as the cohort ages. Since all members of the cohort age at the same rate, we can implement this dependency by changing the mortality parameters as the simulation progresses.[8] Similarly, in a model simulating a cohort starting from disease diagnosis, any dependence on time since diagnosis can be implemented as a dependence on the time since simulation start. Seasonal or temporal variation in disease incidence can also be reflected through simulation-time dependency, where the risk of developing a disease can vary based on the current season or year in the simulation.

State-residence time-dependency captures time dependence on events that members of the cohort could experience at different times. For example, in a model simulating a cohort of healthy individuals, they may experience disease onset at different times. Thus, parameters that depend on the time since disease onset cannot be implemented based on simulation time; instead, we need to track cohort members *from their disease onset time*. We implement this type of time-dependency by expanding the model state space to include disease states that encode the time since an event has occurred. For example, instead of a single "Sick" state, individuals would transition first to the "Sick - cycle 1" state at disease onset, then "Sick - cycle 2" at the next cycle, and so on. In this way, each replicate of the "Sick" state can have different transition probabilities (e.g., mortality, risk of complications, etc.), costs, or utilities. We should note that we can also use this state expansion approach to time-dependency to model simulation-time dependency. However, as we see below, substantial state expansion can be cumbersome and potentially computationally burdensome. For simplicity, modelers should always consider simulation time-dependency first.

Besides describing the two forms of time-dependency, we also illustrate the concept and implementation of transition rewards, such as one-time costs or utilities applied when individuals experience events when transitioning between certain states. These transition rewards reflect event-driven outcome impacts, such as a higher cost for the first cycle of disease onset due to increased diagnostic and management costs or the increased cost of transition to the dead state incurred from end-of-life interventions or management.[9]

In this tutorial, we describe how to implement simulation-time dependent cSTMs and then add state residence dependency to account for both time dependencies. We illustrate the use of these cSTMs to conduct a cost-effectiveness analysis (CEA) in R, a statistical software with increasing use in health decision sciences.[10] We also illustrate the calculation of various epidemiological measures beyond the simple cohort trace, including survival, life expectancy, prevalence, and incidence. Some of these calculations leverage the state-transition array implementation that we will introduce, while others are manipulations of the cohort trace. Such epidemiological measures may be of interest for specific analyses but are also important for calibrating and validating the model against real-world data.

Readers can find the most up-to-date model code and code to create the tutorial graphs in the accompanying GitHub repository (https://github.com/DARTH-git/cohort-modeling-tutorial-timedep). We encourage the reader first to review the basics of decision modeling and how to develop time-independent cSTMs in R, as described in the introductory tutorial.

2 Simulation-time dependency

As described above, simulation-time dependency represents the time since the model starts and can be represented by defining the transition probability matrix as a function of time, P_t. The elements of P_t are the transition probabilities of moving from state i to state j in time t, $p_{[i,j,t]}$, where $\{i, j\} = 1, \ldots, n_S$ and $t = 0, \ldots, n_T$, n_S is the number of health states of the model and n_T is the number of cycles that represent total simulation time.

$$P_t = \begin{bmatrix} p_{[1,1,t]} & p_{[1,2,t]} & \cdots & p_{[1,n_S,t]} \\ p_{[2,1,t]} & p_{[2,2,t]} & \cdots & p_{[2,n_S,t]} \\ \vdots & \vdots & \ddots & \vdots \\ p_{[n_S,1,t]} & p_{[n_S,2,t]} & \cdots & p_{[n_S,n_S,t]} \end{bmatrix}.$$

Note that in each cycle t all rows of the transition probability matrix must sum to one, $\sum_{j=1}^{n_S} p_{[i,j,t]} = 1$ for all $i = 1, \ldots, n_S$ and $t = 0, \ldots, n_T$.

Next, we specify the initial distribution of the cohort at $t = 0$. We then define $\mathbf{m}_0$ as the vector that captures the distribution of the cohort among the states at $t = 0$ (i.e., the initial state vector). As illustrated in the introductory tutorial, we iteratively compute the cohort distribution among the health states in each cycle from the transition probability matrix and the cohort's distribution from the prior cycle. The state vector at the next cycle $t + 1$ ($\mathbf{m}_{t+1}$) is then calculated as the matrix product of the state vector at cycle t, $\mathbf{m}_t$, and the transition probability matrix that the cohort faces in cycle t, P_t,

$$\mathbf{m}_{t+1} = \mathbf{m}_t P_t \qquad \text{for} \qquad t = 0, \ldots, (n_T - 1). \tag{1}$$

Equation (1) is iteratively evaluated until $t = n_T$.

The cohort trace matrix, M, is a matrix of dimensions $(n_T + 1) \times n_S$ where each row is a state vector $(-\mathbf{m}_t-)$, such that

$$M = \begin{bmatrix} -\mathbf{m}_0- \\ -\mathbf{m}_1- \\ \vdots \\ -\mathbf{m}_{n_T}- \end{bmatrix}.$$

M stores the output of the cSTM, which we can use to compute various epidemiological measures, such as prevalence and survival probability over time, and economic measures, such as cumulative resource use and costs.

3 Time dependency on state residence

The implementation of state residence time dependency is slightly more involved than simulation time dependency. As described above, dependence on state residence occurs in applications where transition probabilities or rewards depend on the time spent in a given state. To account for state residence dependency, we expand the number of states with as many transient states as the number of cycles required for state residency. These transient states are often called *tunnel* states, where the cohort stays for only one cycle in each tunnel state and either transitions to the next tunnel state or completely exits the tunnel. The total number of states for a cSTM with tunnels is $n_{S_{\text{tunnels}}}$ and equals $n_S + n_{\text{tunnels}} - 1$, where n_{tunnels} is the total number of times a health state needs to be expanded for. We subtract one because one of the original states is expanded to a tunnel state. The transition probability matrix also needs to be expanded to incorporate these additional transient states, resulting in a transition probability matrix of dimensions $n_{S_{\text{tunnels}}} \times n_{S_{\text{tunnels}}}$. If the transition probabilities are also dependent on simulation time, as described in the previous section, we add a third dimension such that the dimensions will become $n_{S_{\text{tunnels}}} \times n_{S_{\text{tunnels}}} \times n_T$. As state residence time dependency extends in more health states, the total number of health states will increase, causing what has been referred to as "state explosion". Table 1 describes the core components of time-dependent cSTMs and their suggested R code names.

Table 1: Core components of time-dependent cSTMs with their R name.

Element	Description	R name
n_S	Number of states	`n_states`
$\mathbf{m}_0$	Initial state vector	`v_s_init`
$\mathbf{m}_t$	State vector in cycle t	`v_mt`
M	Cohort trace matrix	`m_M`
$\mathbf{P}$	Time-dependent transition probability array	`a_P`
$\mathbf{A}$	Transition-dynamics array	`a_A`
n_{tunnels}	Number of tunnel states	`n_tunnel size`
$n_{S_{\text{tunnels}}}$	Number of states including tunnel states	`n_states_tunnels`
$\mathbf{m}_{\text{tunnels}_0}$	Initial state vector for the model with tunnel states	`v_s_init_tunnels`

For a more detailed description of these components with their variable types, data structure, and R name, please see the Supplementary Materials.

4 Case study: A Cost-effectiveness analysis using a time-dependent Sick-Sicker model

We demonstrate simulation-time and state-residence time-dependency in R by expanding the time-independent 4-state "Sick-Sicker" model described in the introductory tutorial.[7] The state-transition diagram of the Sick-Sicker model is presented in Figure 1. We first modify this cSTM to account for simulation-time dependency by incorporating age-dependent background mortality and then expand it to account for state-residence dependency. We will use this cSTM to conduct a CEA of different treatment strategies accounting for transition rewards.

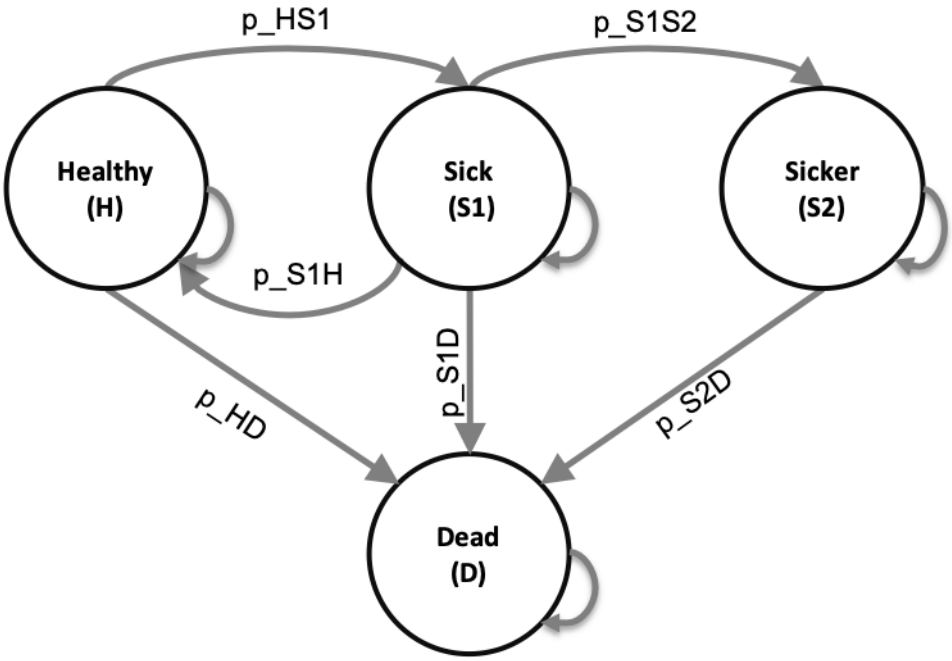

Figure 1: State-transition diagram of the Sick-Sicker cohort state-transition model. The circles represent the health states, and the arrows represent the possible transition probabilities. The labels next to the arrows represent the variable names for these transitions.

As described in the introductory tutorial, this model simulates the health care costs and quality-adjusted life-years (QALYs) of a cohort of 25-year-old individuals at risk of developing a hypothetical disease with two stages: "Sick" and "Sicker".[11] We simulate the cohort of individuals who all start in the "Healthy" state (denoted by "H") over their lifetime, which means that we will simulate the cohort for 75 cycles. The total number of cycles is denoted as n_T and defined in R as `n_cycles`.

In the introductory tutorial, healthy individuals face a constant risk of death. To illustrate simulation time dependency, here we assume that healthy individuals face age-specific background mortality. Healthy individuals who survive might become ill over time and transition to the "Sick" state (denoted by "S1"). In this tutorial, we consider that once healthy individuals get sick, they incur a one-time utility decrement of 0.01 (`du_HS1`, a disutility of transitioning from H to S1) and a transition cost of $1,000 (`ic_HS1`) that reflects the immediate costs of developing the illness. We demonstrate how to include these in the section transition rewards.

The S1 state is associated with higher mortality, higher health care costs, and lower quality of life than the H state. Once in the S1 state, individuals may recover (returning to the H state), die (move to the D state), or progress further to the "Sicker" state (denoted by "S2"), with further increases in mortality risk and health care costs and reduced quality of life. We assume that it is not clinically possible to distinguish between individuals in S1 and S2. Individuals in S1 and S2 face an increased hazard of death, compared to healthy individuals, in the form of a hazard ratio (HR) of 3 and 10, respectively, relative to the background age-specific mortality hazard rate. In the state-residence time-dependent model, the risk of progressing from S1 to S2 depends on the time spent in S1. Once simulated individuals die, they transition to the absorbing D state, where they remain, and incur a one-time cost of $2,000 (`ic_D`) for the expected acute care received

before dying. All transitions between non-death states are assumed to be conditional on surviving each cycle. We simulated the evolution of the cohort in one-year discrete-time cycles.

We use this cSTM to evaluate the cost-effectiveness of four strategies: Strategy A, strategy B, a combination of A and B (strategy AB), and the standard of care (strategy SoC). Strategy A involves administering treatment A to individuals in S1 and S2. Treatment A increases the QoL of individuals only in S1 from 0.75 (utility without treatment, `u_S1`) to 0.95 (utility with treatment A, `u_trtA`) and costs $12,000 per year (`c_trtA`). This strategy does not impact the QoL of individuals in S2, nor does it change the risk of becoming sick or progressing through the sick states. Strategy B uses treatment B to reduce the rate of Sick individuals progressing to the S2 state with a hazard ratio (HR) of 0.6 (`hr_S1S2_trtB`) and costs $13,000 per year (`c_trtB`). Treatment B does not affect QoL. Strategy AB involves administering both treatments A and B, resulting in benefits in increased QoL in patients in the S1 state and reduced risk of progression from S1 to S2, accounting for the costs of both treatments. We discount both costs and QALYs at an annual rate of 3%. Model parameters and the corresponding R variable names are presented in Table 2 and follow the notation described in the DARTH coding framework.[12]

Note that for strategy A, the model has identical transition probabilities to SoC. The only difference is the added cost of the treatment for S1 and S2, and QoL increases for S1. After comparing the four strategies in terms of expected QALYs and costs, we calculate the incremental cost per QALY gained between non-dominated strategies as explained below.

Table 2: Description of parameters, their R variable name, base-case value and distribution.

Parameter	**R name**	**Base-case**	**Distribution**
Number of cycles (n_T)	`n_cycles`	75 years	constant
Names of health states	`v_names_states`	H, S1, S2, D	constant
Annual discount rate for costs	`d_c`	3%	constant
Annual discount rate for QALYs	`d_e`	3%	constant
Number of PSA samples (K)	`n_sim`	1,000	constant
Annual transition probabilities conditional on surviving			
- Disease onset (H to S1)	`p_HS1`	0.15	beta(30, 170)
- Recovery (S1 to H)	`p_S1H`	0.5	beta(60, 60)
- Time-independent disease progression (S1 to S2)	`p_S1S2`	0.105	beta(84, 716)
- Time-dependent disease progression (S1 to S2)	`v_p_S1S2_tunnels`		
Weibull parameters			
Scale (λ)	`p_S1S2_scale`	0.08	lognormal(log(0.08), 0.02)
Shape (γ)	`p_S1S2_shape`	1.10	lognormal(log(1.10), 0.05)
Annual mortality			

Parameter	R name	Base-case	Distribution
- Age-dependent background mortality rate (H to D)	`v_r_HDage`	age-specific	constant
- 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	-
- Cost of treatment A as an additional costs on individuals treated in S1 or S2	`c_trtA`	$12,000	gamma(576.0, 20.8)
- Cost of treatment B as an additional costs on individuals treated in S1 or S2	`c_trtB`	$13,000	gamma(676.0, 19.2)
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	constant
Treatment A effectiveness			
- Utility for treated individuals in S1	`u_trtA`	0.95	beta(300, 15)
Treatment B effectiveness			
- Reduction in rate of disease progression (S1 to S2) as hazard ratio (HR)	`hr_S1S2_trtB`	log(0.6)	lognormal(log(0.6), 0.1)
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)

The R code below describes the initialization of the input parameters.

```r
## General setup
cycle_length <- 1 # cycle length equal one year
n_age_init <- 25  # age at baseline
n_age_max  <- 100 # maximum age of follow up
n_cycles <- n_age_max - n_age_init # number of cycles
# The 4 health states of the model:
v_names_states <- c("H",  # Healthy (H)
                    "S1", # Sick (S1)
```

```
                    "S2", # Sicker (S2)
                    "D")  # Dead (D)
n_states <- length(v_names_states) # number of health states
d_e <- 0.03 # discount rate for QALYs of 3% per cycle
d_c <- 0.03 # discount rate for costs of 3% per cycle
v_names_str <- c("Standard of care", # store the strategy names
                 "Strategy A",
                 "Strategy B",
                 "Strategy AB")

## Transition probabilities (per cycle), hazard ratios and odds ratio (OR)
p_HS1   <- 0.15  # probability of becoming Sick when Healthy
p_S1H   <- 0.5   # probability of becoming Healthy when Sick
p_S1S2  <- 0.105 # probability of becoming Sicker when Sick
hr_S1   <- 3     # hazard ratio of death in Sick vs Healthy
hr_S2   <- 10    # hazard ratio of death in Sicker vs Healthy

# Effectiveness of treatment B
hr_S1S2_trtB <- 0.6 # hazard ratio of becoming Sicker when Sick under treatment B

## State rewards
## Costs
c_H    <- 2000  # cost of being Healthy for one cycle
c_S1   <- 4000  # cost of being Sick for one cycle
c_S2   <- 15000 # cost of being Sicker for one cycle
c_D    <- 0     # cost of being dead for one cycle
c_trtA <- 12000 # cost of receiving treatment A for one cycle
c_trtB <- 13000 # cost of receiving treatment B for one cycle
# Utilities
u_H    <- 1     # utility of being Healthy for one cycle
u_S1   <- 0.75  # utility of being Sick for one cycle
u_S2   <- 0.5   # utility of being Sicker for one cycle
u_D    <- 0     # utility of being dead for one cycle
u_trtA <- 0.95  # utility when receiving treatment A for one cycle

## Transition rewards
du_HS1 <- 0.01  # one-time utility decrement when transitioning from Healthy to Sick
ic_HS1 <- 1000  # one-time cost when transitioning from Healthy to Sick
ic_D   <- 2000  # one-time cost when dying
```

4.1 Incorporating simulation-time dependency

To illustrate simulation-time dependency in the Sick-Sicker cSTM, we model all-cause mortality as a function of age. We obtain all-cause mortality from life tables in the form of age-specific mortality hazard rates, $\mu(a)$, where a refers to age. For this example, we create a vector `v_r_mort_by_age` to represent age-specific background mortality hazard rates for 0 to 100 year-olds obtained from US life tables.[13] To compute the transition probability from state H to state D, corresponding to the cohort's age at each cycle, we transform the rate $\mu(a)$ to a transition probability assuming a constant exponential hazard rate within each year of age

$$p_{[H,D,t]} = 1 - \exp\left\{-\mu(a_0 + t)\right\},$$

where $a_0 = 25$ is the starting age of the cohort. Instead of iterating through the mortality hazard rates, we obtain a vector of background mortality hazard rates for the ages of interest between 25 through 100 by subsetting $\mu(a)$ (R variable name `v_r_mort_by_age`) for these ages. We transform the resulting R variable, `v_r_HDage`, to a probability.

```
# Age-specific mortality rate in the Healthy state (background mortality)
v_r_HDage <- v_r_mort_by_age[(n_age_init + 1) + 0:(n_cycles - 1)]
# Transform to age-specific background mortality risk
v_p_HDage  <- 1 - exp(-v_r_HDage)
```

Because mortality in S1 and S2 are relative to background mortality which depends on age, mortality in S1 and S2 will also be age-dependent. To generate the age-specific mortality in S1 and S2, we multiply the age-specific background mortality rate, `v_r_HDage`, by the constant hazard ratios `hr_S1` and `hr_S2`, respectively. We then convert the resulting age-specific mortality rates to probabilities ensuring that the transition probabilities to D are bounded between 0 and 1.

```
## Age-specific mortality rates in the Sick and Sicker states
v_r_S1Dage <- v_r_HDage * hr_S1 # when Sick
v_r_S2Dage <- v_r_HDage * hr_S2 # when Sicker
## Age-specific probabilities of dying in the Sick and Sicker states
v_p_S1Dage <- 1 - exp(-v_r_S1Dage) # when Sick
v_p_S2Dage <- 1 - exp(-v_r_S2Dage) # when Sicker
```

To incorporate simulation-time dependency into the transition probability matrix, we expand the dimensions of the matrix and create a 3-dimensional transition probability array, $\mathbf{P}$ and named `a_P` in R, of dimensions $n_S \times n_S \times n_T$. The first two dimensions of this array correspond to transitions between states and the third dimension to time. The t-th element in the third dimension corresponds to the transition probability matrix at cycle t. A visual representation of `a_P` is shown in Figure 2.

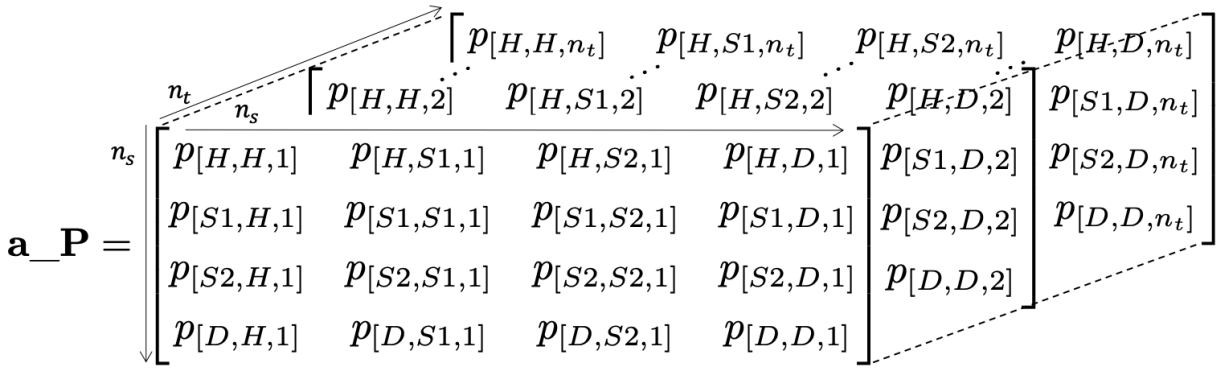

Figure 2: A 3-dimensional representation of the transition probability array of the Sick-Sicker model with simulation-time dependency.

First, we initialize the transition probability array for SoC, `a_P_SoC`, with a default value of zero for all transition probabilities.

```
# Initialize the transition probability array
a_P_SoC <- array(0, dim = c(n_states, n_states, n_cycles),
                 dimnames = list(v_names_states, v_names_states, 0:(n_cycles - 1)))
```

Filling `a_P_SoC` with the corresponding transition probabilities of the cohort under the SoC strategy is comparable with filling a transition probability matrix for a time-independent cSTM. However, this requires a slight modification of the code from the time-independent cSTM. Accounting for the time dimension is represented by the third dimension of the array. The code below illustrates how to assign age-dependent transition probabilities in the third dimension of the array. For constant transitions over time, we only need to provide one value for the transition probability. R replicates the value of such transitions as many times as the number of cycles ($n_T + 1$ times in our example). We create the transition probability array for strategy A as a copy of SoC's because treatment A does not alter the cohort's transition probabilities.

```
### Fill in array
## From H
a_P_SoC["H", "H", ]   <- (1 - v_p_HDage) * (1 - p_HS1)
a_P_SoC["H", "S1", ]  <- (1 - v_p_HDage) * p_HS1
a_P_SoC["H", "D", ]   <- v_p_HDage
## From S1
a_P_SoC["S1", "H", ]  <- (1 - v_p_S1Dage) * p_S1H
a_P_SoC["S1", "S1", ] <- (1 - v_p_S1Dage) * (1 - (p_S1H + p_S1S2))
a_P_SoC["S1", "S2", ] <- (1 - v_p_S1Dage) * p_S1S2
a_P_SoC["S1", "D", ]  <- v_p_S1Dage
## From S2
a_P_SoC["S2", "S2", ] <- 1 - v_p_S2Dage
a_P_SoC["S2", "D", ]  <- v_p_S2Dage
## From D
```

```r
a_P_SoC["D", "D", ]   <- 1

## Initialize transition probability matrix for strategy A as a copy of SoC's
a_P_strA <- a_P_SoC
```

As mentioned above, each slice along the third dimension of `a_P_SoC` corresponds to a transition probability matrix. For example, the transition matrix for 25-year-olds in the Sick-Sicker model under the SoC strategy can be retrieved by indexing the first slice of the array using:

```r
a_P_SoC[, , 1]
```

```
##            H        S1        S2           D
## H  0.8491385 0.1498480 0.0000000 0.001013486
## S1 0.4984813 0.3938002 0.1046811 0.003037378
## S2 0.0000000 0.0000000 0.9899112 0.010088764
## D  0.0000000 0.0000000 0.0000000 1.000000000
```

For strategy B, we first initialize the three-dimensional array of transition probabilities, `a_P_strB` as a copy of `a_P_SoC` and update only the probability of remaining in S1 and the transition probability from S1 to S2 (i.e., `p_S1S2` is replaced with `p_S1S2_trtB`). Next, we create the transition probability array for strategy AB, `a_P_strAB`, as a copy of `a_P_strB` since the cSTMs for strategies B and AB have identical transition probabilities.

```r
## Initialize transition probability array for strategy B
a_P_strB <- a_P_SoC
## Update only transition probabilities from S1 involving p_S1S2
a_P_strB["S1", "S1", ] <- (1 - v_p_S1Dage) * (1 - (p_S1H + p_S1S2_trtB))
a_P_strB["S1", "S2", ] <- (1 - v_p_S1Dage) * p_S1S2_trtB

## Initialize transition probability matrix for strategy AB as a copy of B's
a_P_strAB <- a_P_strB
```

Once we create the transition probability arrays, we check they are valid (i.e., ensuring transition probabilities are between 0 and 1, and transition probabilities from each state sum to 1) using the functions `check_sum_of_transition_array` and `check_transition_probability`, provided in the `darthtools` package (https://github.com/DARTH-git/darthtools).

```r
### Check if transition probability matrices are valid
## Check that transition probabilities are [0, 1]
check_transition_probability(a_P_SoC)
check_transition_probability(a_P_strA)
check_transition_probability(a_P_strB)
check_transition_probability(a_P_strAB)
## Check that all rows sum to 1
check_sum_of_transition_array(a_P_SoC, n_states = n_states, n_cycles = n_cycles)
check_sum_of_transition_array(a_P_strA, n_states = n_states, n_cycles = n_cycles)
check_sum_of_transition_array(a_P_strB, n_states = n_states, n_cycles = n_cycles)
```

```
check_sum_of_transition_array(a_P_strAB, n_states = n_states, n_cycles = n_cycles)
```

In the Sick-Sicker model, the entire cohort starts in the Healthy state. Therefore, we create the $1 \times n_S$ initial state vector `v_s_init` with all of the cohort assigned to the H state:

```
v_s_init <- c(H = 1, S1 = 0, S2 = 0, D = 0) # initial state vector
v_s_init
#  H S1 S2  D
#  1  0  0  0
```

We use the variable `v_s_init` to initialize M represented by `m_M` for the cohort under SoC strategy. We also create a trace for each of the other treatment-based strategies. Note that the initial state vector, `v_s_init`, can be modified to account for the distribution of the cohort across the states at the start of the simulation and might vary by strategy. To simulate the cohort over the n_T cycles for the simulation-time-dependent cSTM, we initialize four cohort trace matrices, one for each strategy.

```
## Initialize cohort trace for age-dependent cSTM under SoC
m_M_SoC <- matrix(NA,
                  nrow = (n_cycles + 1), ncol = n_states,
                  dimnames = list(0:n_cycles, v_names_states))
# Store the initial state vector in the first row of the cohort trace
m_M_SoC[1, ] <- v_s_init
## Initialize cohort trace for strategies A, B, and AB
# Structure and initial states are the same as for SoC
m_M_strA  <- m_M_SoC # Strategy A
m_M_strB  <- m_M_SoC # Strategy B
m_M_strAB <- m_M_SoC # Strategy AB
```

We then use the matrix product to get the state vector of the cohort's distribution at each cycle t. This equation is similar to the one described for the time-independent model. The only modification required is to index the transition probability arrays by t to obtain each strategy's cycle-specific transition probability matrices.

```
# Iterative solution of age-dependent cSTM
for(t in 1:n_cycles){
  # For SoC
  m_M_SoC[t + 1, ] <- m_M_SoC[t, ] %*% a_P_SoC[, , t]
  # For strategy A
  m_M_strA[t + 1, ] <- m_M_strA[t, ] %*% a_P_strA[, , t]
  # For strategy B
  m_M_strB[t + 1, ] <- m_M_strB[t, ] %*% a_P_strB[, , t]
  # For strategy AB
  m_M_strAB[t + 1, ] <- m_M_strAB[t, ] %*% a_P_strAB[, , t]
}
```

A graphical representation of the cohort trace for all cycles of the age-dependent cSTM under SoC is shown in Figure 3.

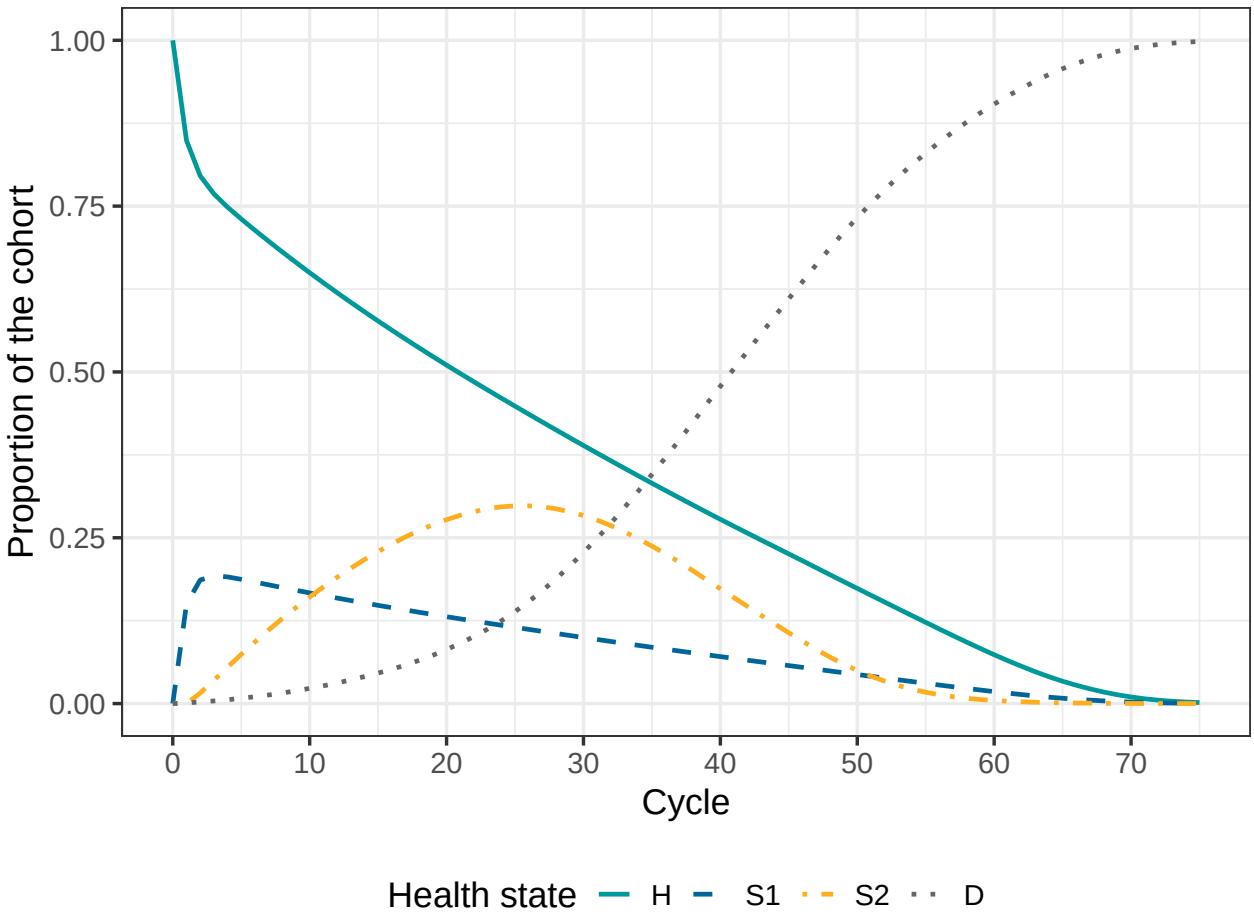

Figure 3: Cohort trace of the age-dependent cSTM under SoC.

4.2 Incorporating time dependency on state residence

Here we add the dependency on state-residence to the simulation-time-dependent Sick-Sicker model defined above. We assume the risk of progression from S1 to S2 increases as a function of the time $\tau = 1, \ldots, n_{\text{tunnels}}$ the cohort remains in the S1 state. This increase follows a Weibull hazard function, $h(\tau)$, defined as

$$h(\tau) = \gamma\lambda(\lambda\tau)^{\gamma-1},$$

with a corresponding cumulative hazard, $H(\tau)$,

$$H(\tau) = (\lambda\tau)^{\gamma}, \tag{2}$$

where λ and γ are the scale and shape parameters of the Weibull hazard function, respectively.

To derive a transition probability from S1 to S2 as a function of the time the cohort spends in S1, $p_{[S1_\tau,S2,\tau]}$, we assume constant rates within each cycle interval (i.e., piecewise exponential transition times), where the cycle-specific probability of a transition is

$$p_{[S1_\tau,S2,\tau]} = 1 - \exp\left(-\mu_{[S1_\tau,S2,\tau]}\right), \tag{3}$$

where $\mu_{[S1_\tau, S2,\tau]}$ is the rate of transition from S1 to S2 in cycle τ defined as the difference in cumulative hazards between consecutive cycles[14]

$$\mu_{[S1_\tau, S2,\tau]} = H(\tau) - H(\tau - 1). \tag{4}$$

Substituting the Weibull cumulative hazard from Equation (2) into Equation (4) gives

$$\mu_{[S1_\tau, S2,\tau]} = (\lambda\tau)^\gamma - (\lambda(\tau - 1))^\gamma, \tag{5}$$

and the transition probability

$$p_{[S1_\tau, S2,\tau]} = 1 - \exp\left(-\left((\lambda\tau)^\gamma - (\lambda(\tau - 1))^\gamma\right)\right). \tag{6}$$

We assume that state-residence dependency affects the cohort in the S1 state throughout the whole simulation (i.e., $n_{\text{tunnels}} = n_T$) and create a new variable called `n_tunnel_size` with the length of the tunnel equal to `n_cycles`. Thus, there will be 75 S1 tunnel states plus 3 more states (H, S2, D) resulting in a total of $n_{S_{\text{tunnels}}} = 78$.

Figure 4 shows the Sick-Sicker model's state-transition diagram with state-residence dependency with n_{tunnels} tunnel states for S1.

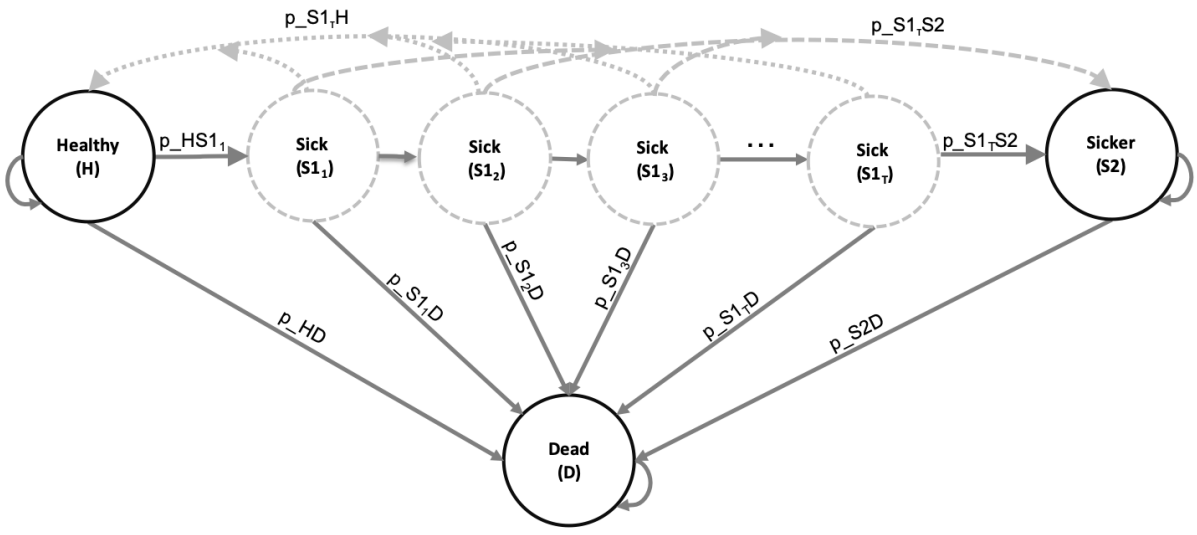

Figure 4: State-transition diagram of the Sick-Sicker model with tunnel states expanding the Sick state $(S1_1, S1_2, ..., S1_{n_{\text{tunnels}}})$.

To implement state-residence dependency in the Sick-Sicker cSTM, we create the vector variables `v_Sick_tunnel` and `v_names_states_tunnels` with the names of the Sick tunnel states' and all the states of the cSTM, including tunnels, respectively, and use the parameters listed in Table 1.

```r
## Number of tunnels
n_tunnel_size <- n_cycles
## Vector with cycles for tunnels
v_cycles_tunnel <- 1:n_tunnel_size
## Vector with the names of the Sick tunnel state
v_Sick_tunnel <- paste("S1_", seq(1, n_tunnel_size), "Yr", sep = "")
## Create variables for model with tunnels
v_names_states_tunnels <- c("H", v_Sick_tunnel, "S2", "D") # state names
n_states_tunnels <- length(v_names_states_tunnels)         # number of states
## Initialize first cycle of Markov trace accounting for the tunnels
v_s_init_tunnels <- c(1, rep(0, n_tunnel_size), 0, 0)
```

Then, the transition rate and probability dependent on state residence from Sick to Sicker, `v_r_S1S2_tunnels` and `v_p_S1S2_tunnels`, respectively, based on a Weibull hazard function are:

```r
# Weibull parameters
p_S1S2_scale <- 0.08 # scale
p_S1S2_shape <- 1.10 # shape
# Weibull function
v_r_S1S2_tunnels <- (v_cycles_tunnel*p_S1S2_scale)^p_S1S2_shape -
                    ((v_cycles_tunnel-1)*p_S1S2_scale)^p_S1S2_shape

v_p_S1S2_tunnels <- 1 - exp(-v_r_S1S2_tunnels*cycle_length)
```

To adapt the 3-dimensional transition probability array to incorporate both age and state-residence dependency in the Sick-Sicker model under SoC, we first create an expanded 3-dimensional array accounting for tunnels, `a_P_tunnels_SoC`. The dimensions of this array are $n_{S_{\text{tunnels}}} \times n_{S_{\text{tunnels}}} \times n_T$. A visual representation of `a_P_tunnels_SoC` of the Sick-Sicker model with tunnel states expanding the Sick state is shown in Figure 5.

```r
# Initialize array
a_P_tunnels_SoC <- array(0, dim = c(n_states_tunnels, n_states_tunnels, n_cycles),
                         dimnames = list(v_names_states_tunnels,
                                         v_names_states_tunnels,
                                         0:(n_cycles - 1)))
```

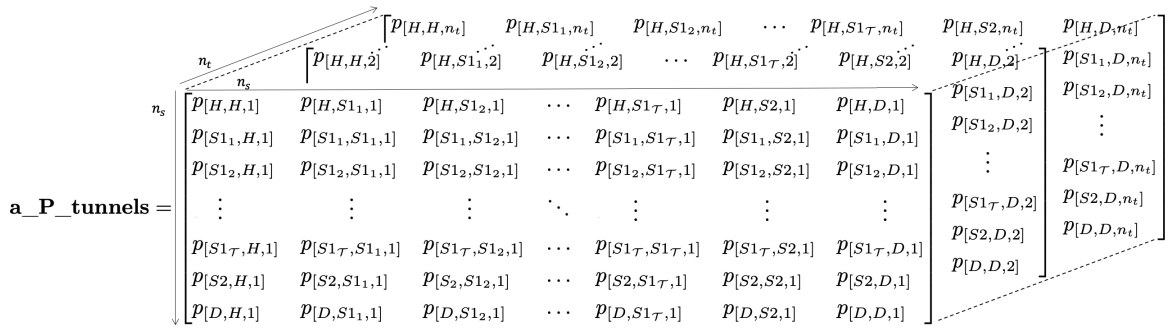

Figure 5: The 3-dimensional transition probability array of the Sick-Sicker model expanded to account for simulation-time and state-residence dependency using τ tunnel states for S1.

Filling `a_P_tunnels_SoC` with the corresponding transition probabilities is similar to how it's done with `a_P_SoC` above, with the difference being that we now fill the transition probabilities from all the tunnel states by iterating through all the tunnel states and assigning the corresponding disease progression transition probabilities for each tunnel state.

```
### Fill in array
## From H
a_P_tunnels_SoC["H", "H", ]               <- (1 - v_p_HDage) * (1 - p_HS1)
a_P_tunnels_SoC["H", v_Sick_tunnel[1], ] <- (1 - v_p_HDage) * p_HS1
a_P_tunnels_SoC["H", "D", ]               <- v_p_HDage
## From S1
for(i in 1:(n_tunnel_size - 1)){
  a_P_tunnels_SoC[v_Sick_tunnel[i], "H", ]  <- (1 - v_p_S1Dage) * p_S1H
  a_P_tunnels_SoC[v_Sick_tunnel[i],
                  v_Sick_tunnel[i + 1], ]   <- (1 - v_p_S1Dage) *
                                               (1 - (p_S1H + v_p_S1S2_tunnels[i]))
  a_P_tunnels_SoC[v_Sick_tunnel[i], "S2", ] <- (1 - v_p_S1Dage) * v_p_S1S2_tunnels[i]
  a_P_tunnels_SoC[v_Sick_tunnel[i], "D", ]  <- v_p_S1Dage
}
# Repeat code for the last cycle to force the cohort stay in the last tunnel state of Sick
a_P_tunnels_SoC[v_Sick_tunnel[n_tunnel_size], "H", ]  <- (1 - v_p_S1Dage) * p_S1H
a_P_tunnels_SoC[v_Sick_tunnel[n_tunnel_size],
                v_Sick_tunnel[n_tunnel_size], ] <- (1 - v_p_S1Dage) *
                                                   (1 - (p_S1H + v_p_S1S2_tunnels[n_tunnel_size]))
a_P_tunnels_SoC[v_Sick_tunnel[n_tunnel_size], "S2", ] <- (1 - v_p_S1Dage) *
                                                     v_p_S1S2_tunnels[n_tunnel_size]
a_P_tunnels_SoC[v_Sick_tunnel[n_tunnel_size], "D", ]  <- v_p_S1Dage
### From S2
a_P_tunnels_SoC["S2", "S2", ] <- 1 - v_p_S2Dage
a_P_tunnels_SoC["S2", "D", ]  <- v_p_S2Dage
# From D
```

```
a_P_tunnels_SoC["D", "D", ] <- 1
```

Next, we create the transition probability array for Strategy B. To implement the effectiveness of treatment B, we multiply the vector of transition rates, `v_r_S1S2_tunnels`, by the hazard ratio of treatment B, `hr_S1S2_trtB`. Then, we transform to a vector of transition probabilities that account for the duration of S1 state-residence under treatment B, `v_p_S1S2_tunnels_trtB`, following Equation (3).

```
# Apply hazard ratio to rate to obtain transition rate of becoming Sicker when
# Sick for treatment B
v_r_S1S2_tunnels_trtB <- v_r_S1S2_tunnels * hr_S1S2_trtB
# transform rate to probability to become Sicker when Sick under treatment B
# conditional on surviving
v_p_S1S2_tunnels_trtB <- 1 - exp(-v_r_S1S2_tunnels_trtB*cycle_length)
```

Then, we initialize the three-dimensional transition probability array for treatment B, `a_P_tunnels_trtB`, based on `a_P_tunnels_SoC`. The only difference here is that we update the transition probabilities from S1 involving `v_p_S1S2_tunnels` to `v_p_S1S2_tunnels_trtB` instead.

```
## Initialize transition probability array for treatment B
a_P_tunnels_trtB <- a_P_tunnels_SoC
## Update only transition probabilities from S1 involving v_p_S1S2_tunnels
for(i in 1:(n_tunnel_size - 1)){
  a_P_tunnels_trtB[v_Sick_tunnel[i], "H", ] <- (1 - v_p_S1Dage) * p_S1H
  a_P_tunnels_trtB[v_Sick_tunnel[i],
              v_Sick_tunnel[i + 1], ] <- (1 - v_p_S1Dage) *
                                         (1 - (p_S1H + v_p_S1S2_tunnels_trtB[i]))
  a_P_tunnels_trtB[v_Sick_tunnel[i], "S2", ] <- (1 - v_p_S1Dage) * v_p_S1S2_tunnels_trtB[i]
  a_P_tunnels_trtB[v_Sick_tunnel[i], "D", ]  <- v_p_S1Dage
}
# repeat code for the last cycle to force the cohort stay in the last tunnel state of Sick
a_P_tunnels_trtB[v_Sick_tunnel[n_tunnel_size], "H", ] <- (1 - v_p_S1Dage) * p_S1H
a_P_tunnels_trtB[v_Sick_tunnel[n_tunnel_size],
            v_Sick_tunnel[n_tunnel_size], ] <- (1 - v_p_S1Dage) *
                                               (1 - (p_S1H +v_p_S1S2_tunnels_trtB[n_tunnel_size]))
a_P_tunnels_trtB[v_Sick_tunnel[n_tunnel_size], "S2", ] <- (1 - v_p_S1Dage) *
                                                          v_p_S1S2_tunnels_trtB[n_tunnel_size]
a_P_tunnels_trtB[v_Sick_tunnel[n_tunnel_size], "D", ]  <- v_p_S1Dage
```

Once we create both three-dimensional transition probability arrays with tunnels, we check they are valid (i.e., between 0 and 1 and transition probabilities from each state sum to 1).

```
### Check if transition probability matrices are valid
## Check that transition probabilities are [0, 1]
check_transition_probability(a_P_tunnels_SoC)
check_transition_probability(a_P_tunnels_trtB)
## Check that all rows sum to 1
```

```
check_sum_of_transition_array(a_P_tunnels_SoC, n_states = n_states_tunnels,
                              n_cycles = n_cycles)
check_sum_of_transition_array(a_P_tunnels_trtB, n_states = n_states_tunnels,
                              n_cycles = n_cycles)
```

To simulate the cohort and store its state occupation over the n_T cycles for the cSTM accounting for state-residence dependency, we initialize two new cohort trace matrices for the SoC and treatment B, `m_M_tunnels_SoC` and `m_M_tunnels_trtB`, respectively. The dimensions of both matrices are $(n_T + 1) \times n_{S_{\text{tunnels}}}$.

```
# Initialize cohort for state-residence cSTM under SoC
m_M_tunnels_SoC <- matrix(0,
                      nrow = (n_cycles + 1), ncol = n_states_tunnels,
                      dimnames = list(0:n_cycles, v_names_states_tunnels))
# Store the initial state vector in the first row of the cohort trace
m_M_tunnels_SoC[1, ] <- v_s_init_tunnels
## Initialize cohort trace under treatment B
m_M_tunnels_trtB <- m_M_tunnels_SoC
```

We then use the matrix product, similar to the simulation-time-dependent cSTM, to generate the full cohort trace.

```
# Iterative solution of state-residence-dependent cSTM
for(t in 1:n_cycles){
  # For SoC
  m_M_tunnels_SoC[t + 1, ] <- m_M_tunnels_SoC[t, ] %*% a_P_tunnels_SoC[, , t]
  # Under treatment B
  m_M_tunnels_trtB[t + 1,] <- m_M_tunnels_trtB[t, ] %*% a_P_tunnels_trtB[, , t]
}
```

To compute a summarized cohort trace to capture occupancy in the H, S1, S2, D states under SoC, we aggregate over the tunnel states in each cycle(Figure 6).

```
# Create aggregated trace
m_M_tunnels_SoC_sum <- cbind(H = m_M_tunnels_SoC[, "H"],
                             S1 = rowSums(m_M_tunnels_SoC[, which(v_names_states=="S1"):
                                                            (n_tunnel_size +1)]),
                             S2 = m_M_tunnels_SoC[, "S2"],
                             D = m_M_tunnels_SoC[, "D"])
```

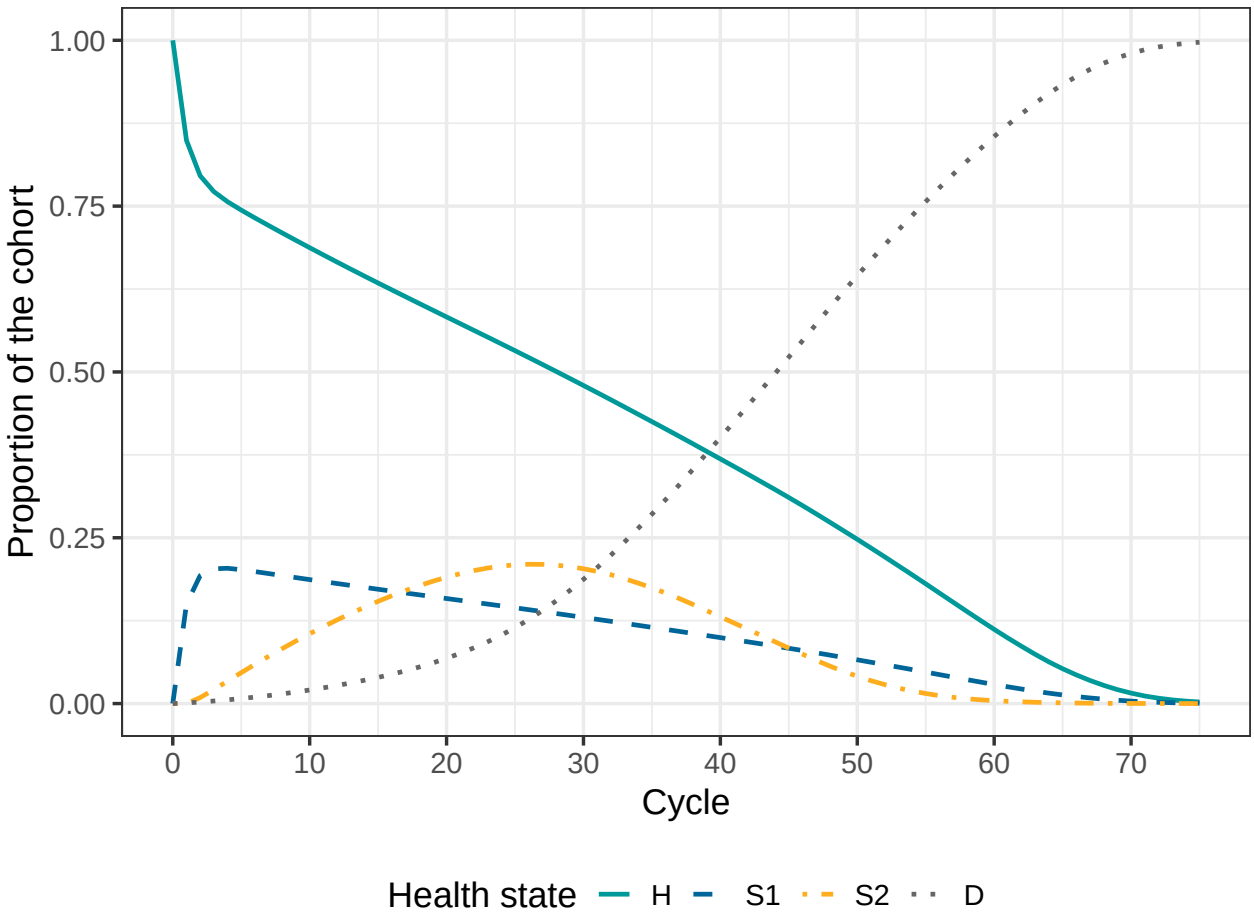

Figure 6: Cohort trace of the age-dependent cSTM accounting for state-residence dependency under SoC.

5 Epidemiological and economic measures

cSTMs can be used to generate different epidemiological and economic outputs. In a CEA, the main outcomes are typically the total discounted expected QALYs and total costs accrued by the cohort over the predefined time horizon. However, epidemiological outcomes can be helpful for calibration and validation. Some common epidemiological outcomes include survival, prevalence, incidence, the average number of events, and lifetime risk of events.[15] We show how to obtain some of these outcomes from the trace and transition probability objects.

5.1 Epidemiological measures

We provide the epidemiological definition of some of these outcomes and how they can be generated from a cSTM using the simulation time-dependent Sick-Sicker cSTM under SoC. In the GitHub repository, we provide the code to generate these outcomes from the state-residence-dependent cSTM.

5.1.1 Survival probability

The survival probability, $S(t)$, captures the proportion of the cohort remaining alive by cycle t. To estimate $S(t)$ from the simulated cohort of the simulation-time-dependent Sick-Sicker model, shown in Figure 7, we

sum the proportions of the non-death states for all n_T cycles in `m_M_SoC`.

```
v_S_SoC <- rowSums(m_M_SoC[, -which(v_names_states == "D")]) # vector with survival curve
```

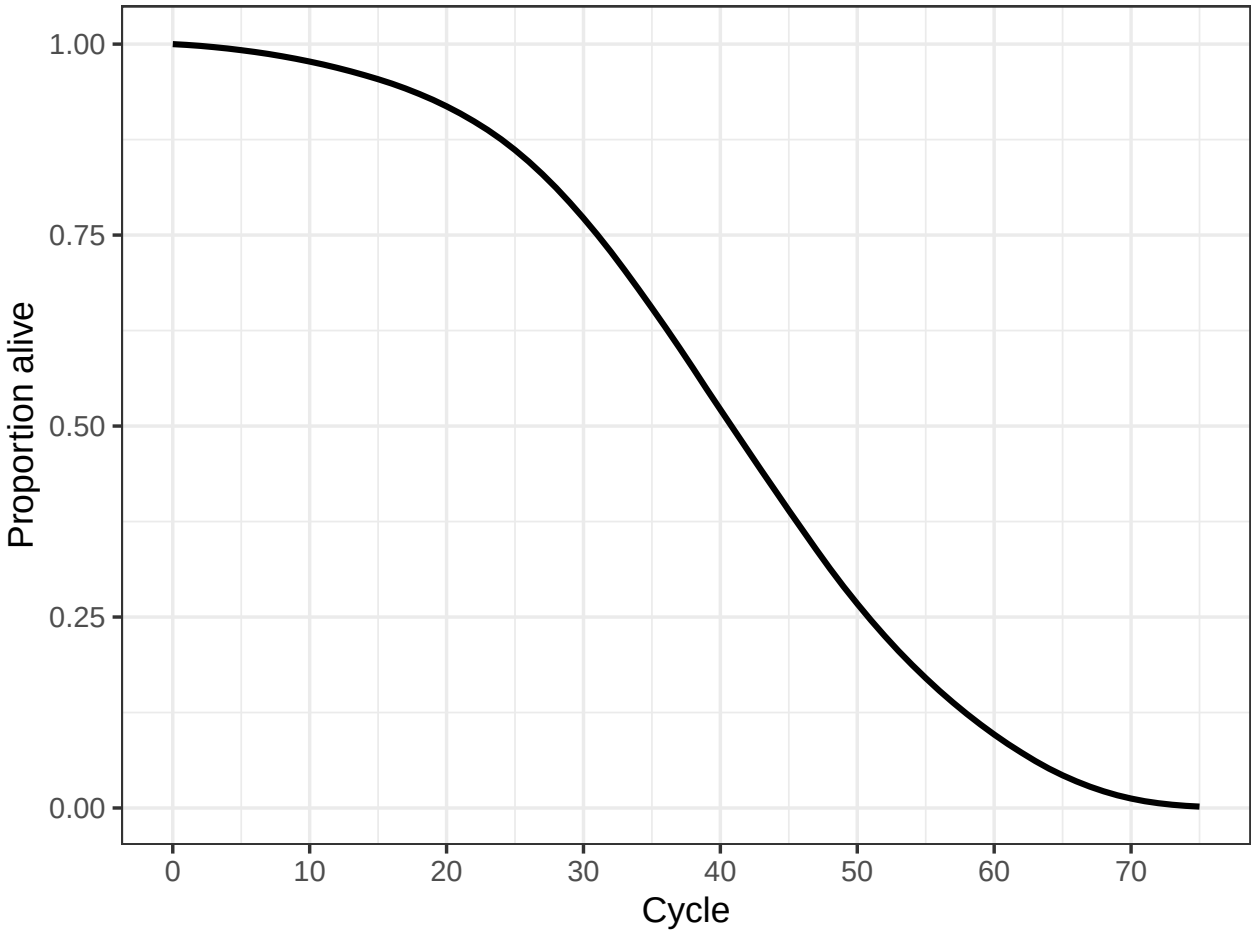

Figure 7: Survival curve of the age-dependent cSTM

5.1.2 Life expectancy

Life expectancy (LE) refers to the expected number of time units remaining to be alive.[16] In continuous-time, LE is the area under the entire survival curve.[17]

$$LE = \int_{t=0}^{\infty} S(t)dt.$$

In discrete-time using cSTMs, we often calculate restricted LE over a fixed time horizon (e.g., n_T) at which most of the cohort has transitioned to the Dead state and is defined as

$$LE = \sum_{t=0}^{n_T} S(t).$$

In the simulation-time-dependent Sick-Sicker model, where we simulated a cohort over n_T= 75 cycles, life

expectancy `le_SoC` is 41.2 cycles, which is calculated as

```
le_SoC <- sum(v_S_SoC) # life expectancy
```

Note that this equation expresses LE in the units of t. We use an annual cycle length; thus, the resulting LE will be in years. Analysts can also use other cycle lengths (e.g., monthly or daily), but the LE must be correctly converted to the desired unit if different than the cycle length units.

5.1.3 Prevalence

Prevalence is defined as the proportion of the population or cohort with a specific condition (or being in a particular health state) among those alive.[18] To calculate the prevalence of S1 at cycle t, $\text{prev}(t)_i$, we compute the ratio between the proportion of the cohort in S1 and the proportion alive at that cycle.[19] The proportion of the cohort alive is given by the survival probability $S(t)$ defined above. The individual prevalence of the S1 and S2 health states and the overall prevalence of sick individuals (i.e., S1 + S2) of the age-dependent Sick-Sicker cSTM at each cycle t is computed as follows and are shown in Figure 8.

```
v_prev_S1_SoC   <- m_M_SoC[, "S1"] / v_S_SoC          # vector with prevalence of Sick
v_prev_S2_SoC   <- m_M_SoC[, "S2"] / v_S_SoC          # vector with prevalence of Sicker
v_prev_S1S2_SoC <- rowSums(m_M_SoC[, c("S1", "S2")])/v_S_SoC # prevalence of Sick and Sicker
```

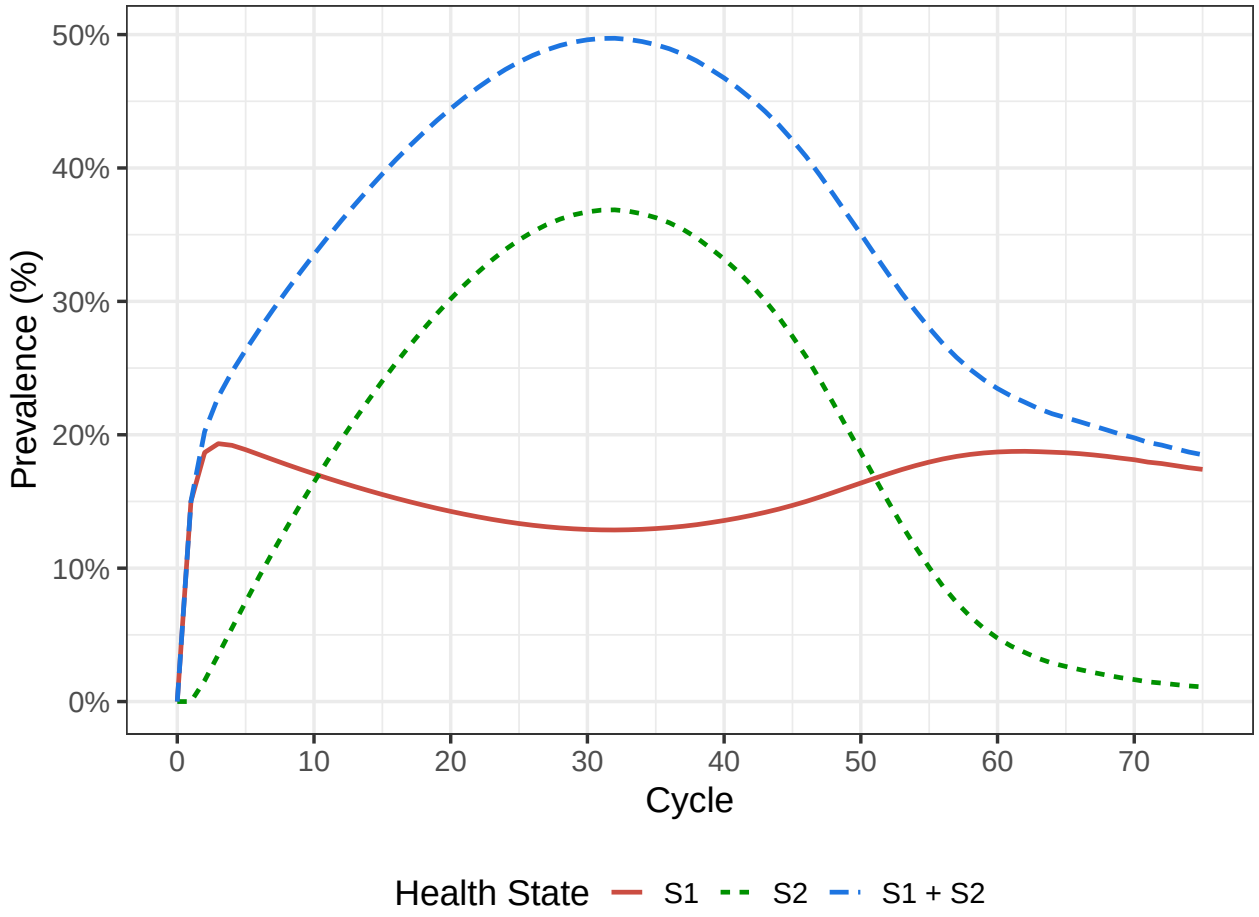

Figure 8: Prevalence of sick states in age-dependent cSTM

5.2 Economic measures

In CEA, we can calculate economic outcomes from state and transition rewards. A "state reward" refers to a value (e.g., cost, utility) assigned to individuals for remaining in a given health state for one cycle. A "transition reward" refers to the increase or decrease in either costs or utilities of transitioning from one health state to another, which may be associated with a one-time cost or utility impact. In the accompanying tutorial, we describe how to incorporate state rewards in CEA in detail.[7] Here, we describe and illustrate how to implement both state and transition rewards together using a transition array.

5.2.1 State rewards

As shown in the introductory tutorial, to add state rewards to the Sick-Sicker model, we first create a vector of utilities and costs for each of the four strategies considered. The vectors of utilities and costs, `v_u_SoC` and `v_c_SoC`, respectively, contain the utilities and costs corresponding to being in each of the four health states under SoC, shown in Table 2.

```
# Vector of state utilities under SoC
v_u_SoC <- c(H = u_H, S1 = u_S1, S2 = u_S2, D = u_D)
# Vector of state costs under SoC
v_c_SoC <- c(H = c_H, S1 = c_S1, S2 = c_S2, D = c_D)
```

We account for the benefits and costs of both treatments individually and their combination to create the state-reward vectors under treatments A and B (strategies A and B, respectively) and when applied jointly (strategy AB). Only treatment A affects QoL, so we create a vector of utilities for strategy A, `v_u_strA`, where we substitute the utility of being in S1 under SoC, `u_S1`, with the utility associated with the benefit of treatment A in being in that state, `u_trtA`. Treatment B does not affect QoL, so the vector of utilities for strategy B, `v_u_strB`, is the same as SoC's vector. However, when both treatments A and B are applied jointly (strategy AB), the resulting vector of utilities `v_u_strAB` equals that of strategy A.

```
# Vector of state utilities for strategy A
v_u_strA <- c(H = u_H, S1 = u_trtA, S2 = u_S2, D = u_D)
# Vector of state utilities for strategy B
v_u_strB <- v_u_SoC
# Vector of state utilities for strategy AB
v_u_strAB <- v_u_strA
```

Both treatments A and B incur a cost. To create the vector of state costs for strategy A, `v_c_strA`, we add the cost of treatment A, `c_trtA`, to S1 and S2 state costs. Similarly, when constructing the vector of state costs for strategy B, `v_c_strB`, we add the cost of treatment B, `c_trtB`, to S1 and S2 state costs. Finally, for the vector of state costs for strategy AB, `v_c_strAB`, we add both treatment costs to the state costs of S1 and S2.

```
# Vector of state costs for strategy A
v_c_strA <- c(H  = c_H,
              S1 = c_S1 + c_trtA,
              S2 = c_S2 + c_trtA,
              D  = c_D)
```

```r
# Vector of state costs for strategy B
v_c_strB <- c(H  = c_H,
              S1 = c_S1 + c_trtB,
              S2 = c_S2 + c_trtB,
              D  = c_D)
# Vector of state costs for strategy AB
v_c_strAB <- c(H  = c_H,
               S1 = c_S1 + (c_trtA + c_trtB),
               S2 = c_S2 + (c_trtA + c_trtB),
               D  = c_D)
```

5.2.2 Transition rewards

As previously mentioned, dying (i.e., transitioning to the Dead state) incurs a one-time cost of $2,000 that reflects the acute care that might be received immediately preceding death. We also have a disutility and a cost increment on the transition from H to S1. Incorporating transition rewards requires keeping track of the proportion of the cohort that transitions between health states in each cycle while capturing the origin and destination states for each transition. The cohort trace, M, does not capture this information. However, obtaining this information is relatively straightforward in a cSTM and described in detail by Krijkamp et al. (2020).[9] Briefly, this approach involves changing the core computation in a traditional cSTM, from $m_t P_t$ to $\text{diag}(m_t)P_t$. This simple change allows us to compute the proportion of the cohort that transitions between any two states in a cycle t. The result is no longer a cohort trace matrix, but rather a three-dimensional array that we refer to as a transition-dynamics array ($\mathbf{A}$) with dimensions $n_S \times n_S \times [n_T + 1]$. The $t-$th slice of $\mathbf{A}$, A_t, is a matrix that stores the proportion of the population that transitioned between any two states from cycles $t-1$ to t. Similarly, we define the transition rewards by the states of origin and destination.

To account for both state and transition rewards, we create a *matrix* of rewards R_t of dimensions $n_S \times n_S$. The off-diagonal entries of R_t store the transition rewards, and the diagonal of R_t stores the state rewards for cycle t and assumes that rewards occur at the beginning of the cycle.[9] Finally, we multiply this matrix by A_t, the t-th slice of A, and apply discounting, within-cycle correction, and compute the overall reward for each strategy outcome. Below, we illustrate these computations in R.

To compute $\mathbf{A}$ for the simulation-time-dependent Sick-Sicker model under SoC, we initialize a three-dimensional array `a_A_SoC` of dimensions $n_S \times n_S \times [n_T + 1]$ and set the diagonal of the first slice to the initial state vector `v_s_init`. Next, we create a three-dimensional array for each of the strategies as a copy of the array under SoC.

```r
# Initialize transition-dynamics array under SoC
a_A_SoC <- array(0,
                 dim = c(n_states, n_states, (n_cycles + 1)),
                 dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Set first slice to the initial state vector in its diagonal
diag(a_A_SoC[, , 1]) <- v_s_init
# Initialize transition-dynamics array for strategies A, B, and AB
# Structure and initial states are the same as for SoC
```

```
a_A_strA  <- a_A_SoC
a_A_strB  <- a_A_SoC
a_A_strAB <- a_A_SoC
```

We then compute a matrix multiplication between a diagonal matrix of each of the t-th rows of the cohort trace matrix under SoC and treatment B, denoted as `diag(m_M_SoC[t, ])` and `diag(m_M_strB[t, ])`, by the t-th matrix of the array of transition matrices, `a_P_SoC[, , t]` and `a_P_strB[, , t]`, respectively, over all n_T cycles.

```
# Iterative solution to produce the transition-dynamics array
for (t in 1:n_cycles){
  # For SoC
  a_A_SoC[, , t + 1] <- diag(m_M_SoC[t, ]) %*% a_P_SoC[, , t]
  # For strategy A
  a_A_strA[, , t + 1] <- diag(m_M_strA[t, ]) %*% a_P_strA[, , t]
  # For strategy B
  a_A_strB[, , t + 1] <- diag(m_M_strB[t, ]) %*% a_P_strB[, , t]
  # For strategy AB
  a_A_strAB[, , t + 1] <- diag(m_M_strAB[t, ]) %*% a_P_strAB[, , t]
}
```

To create the arrays of rewards for costs and utilities for the simulation-time-dependent Sick-Sicker cSTM, we create strategy-specific three-dimensional arrays of rewards and fill each of their rows across the third dimension with the vector of state rewards.

```
# Arrays of state and transition rewards
# Utilities under SoC
a_R_u_SoC <- array(matrix(v_u_SoC, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Costs under SoC
a_R_c_SoC <- array(matrix(v_c_SoC, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Utilities under Strategy A
a_R_u_strA <-  array(matrix(v_u_strA, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Costs under Strategy A
a_R_c_strA <- array(matrix(v_c_strA, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Utilities under Strategy B
a_R_u_strB <-  array(matrix(v_u_strB, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
```

```
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Costs under Strategy B
a_R_c_strB <- array(matrix(v_c_strB, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Utilities under Strategy AB
a_R_u_strAB <-  array(matrix(v_u_strAB, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
# Costs under Strategy AB
a_R_c_strAB <- array(matrix(v_c_strAB, nrow = n_states, ncol = n_states, byrow = T),
                  dim = c(n_states, n_states, n_cycles + 1),
                  dimnames = list(v_names_states, v_names_states, 0:n_cycles))
```

To account for the transition rewards, we either add or subtract them in the corresponding location of the reward matrix representing the transitions of interest. Thus, for example, to account for the disutility of transitioning from H to S1 under strategy A, we subtract the disutility to the entry of the array of rewards corresponding to the transition from H to S1 across all cycles.

```
# Add disutility due to transition from Healthy to Sick
a_R_u_strA["H", "S1", ] <- a_R_u_strA["H", "S1", ] - du_HS1
```

In a similar approach, we add the costs of transitioning from H to S1 and the cost of dying under strategy A.

```
# Add transition cost due to transition from Healthy to Sick
a_R_c_strA["H", "S1", ] <- a_R_c_strA["H", "S1", ] + ic_HS1
# Add transition cost of dying from all non-dead states
a_R_c_strA[-n_states, "D", ] <- a_R_c_strA[-n_states, "D", ] + ic_D
a_R_c_strA[, , 1]
```

```
##       H    S1    S2    D
## H  2000 17000 27000 2000
## S1 2000 16000 27000 2000
## S2 2000 16000 27000 2000
## D  2000 16000 27000    0
```

Below, we show how to add the transition rewards to the reward matrices under SoC and strategies B and AB.

```
## SoC
# Add disutility due to transition from H to S1
a_R_u_SoC["H", "S1", ] <- a_R_u_SoC["H", "S1", ] - du_HS1
# Add transition cost due to transition from H to S1
a_R_c_SoC["H", "S1", ] <- a_R_c_SoC["H", "S1", ] + ic_HS1
# Add transition cost of dying from all non-dead states
a_R_c_SoC[-n_states, "D", ] <- a_R_c_SoC[-n_states, "D", ] + ic_D
```

```r
## Strategy B
# Add disutility due to transition from Healthy to Sick
a_R_u_strB["H", "S1", ] <- a_R_u_strB["H", "S1", ] - du_HS1
# Add transition cost due to transition from Healthy to Sick
a_R_c_strB["H", "S1", ] <- a_R_c_strB["H", "S1", ] + ic_HS1
# Add transition cost of dying from all non-dead states
a_R_c_strB[-n_states, "D", ] <- a_R_c_strB[-n_states, "D", ] + ic_D

## Strategy AB
# Add disutility due to transition from Healthy to Sick
a_R_u_strAB["H", "S1", ] <- a_R_u_strAB["H", "S1", ] - du_HS1
# Add transition cost due to transition from Healthy to Sick
a_R_c_strAB["H", "S1", ] <- a_R_c_strAB["H", "S1", ] + ic_HS1
# Add transition cost of dying from all non-dead states
a_R_c_strAB[-n_states, "D", ] <- a_R_c_strAB[-n_states, "D", ] + ic_D
```

The state and transition rewards are applied to the model dynamics by element-wise multiplication between $\mathbf{A}$ and $\mathbf{R}$, indicated by the $\odot$ sign, which produces the array of outputs for all n_T cycles, $\mathbf{Y}$. Formally,

$$\mathbf{Y} = \mathbf{A} \odot \mathbf{R} \tag{7}$$

To obtain $\mathbf{Y}$ for QALYs and costs for all four strategies, we apply Equation (7) by the element-wise multiplication of the transition array `a_A_SoC` by the corresponding array of rewards.

```r
# For SoC
a_Y_c_SoC <- a_A_SoC * a_R_c_SoC
a_Y_u_SoC <- a_A_SoC * a_R_u_SoC
# For Strategy A
a_Y_c_strA <- a_A_strA * a_R_c_strA
a_Y_u_strA <- a_A_strA * a_R_u_strA
# For Strategy B
a_Y_c_strB <- a_A_strB * a_R_c_strB
a_Y_u_strB <- a_A_strB * a_R_u_strB
# For Strategy AB
a_Y_c_strAB <- a_A_strAB * a_R_c_strAB
a_Y_u_strAB <- a_A_strAB * a_R_u_strAB
```

The total rewards for each health state at cycle t, $\mathbf{y}_t$, is obtained by summing the rewards across all $j = 1, \ldots, n_S$ health states for all n_T cycles.

$$\mathbf{y}_t = \mathbf{1}^T Y_t = \left[\sum_{i=1}^{n_S} Y_{[i,1,t]}, \sum_{i=1}^{n_S} Y_{[i,2,t]}, \ldots, \sum_{i=1}^{n_S} Y_{[i,n_S,t]} \right]. \tag{8}$$

To obtain the expected costs and QALYs per cycle for each strategy, $\mathbf{y}$, we apply Equation (8) again across

all the matrices of the third dimension of $\mathbf{Y}$ for all the outcomes.

```
# Vectors of rewards
# QALYs under SoC
v_qaly_SoC <- rowSums(t(colSums(a_Y_u_SoC)))
# Costs under SoC
v_cost_SoC <- rowSums(t(colSums(a_Y_c_SoC)))
# QALYs under Strategy A
v_qaly_strA <- rowSums(t(colSums(a_Y_u_strA)))
# Costs under Strategy A
v_cost_strA <- rowSums(t(colSums(a_Y_c_strA)))
# QALYs under Strategy B
v_qaly_strB <- rowSums(t(colSums(a_Y_u_strB)))
# Costs under Strategy B
v_cost_strB <- rowSums(t(colSums(a_Y_c_strB)))
# QALYs under Strategy AB
v_qaly_strAB <- rowSums(t(colSums(a_Y_u_strAB)))
# Costs under Strategy AB
v_cost_strAB <- rowSums(t(colSums(a_Y_c_strAB)))
```

5.2.3 Within-cycle correction and discounting future rewards

Following the steps in the introductory cSTM tutorial,[7] here we use Simpson's 1/3rd rule for within-cycle correction (WCC),[21] and use exponential discounting for costs and QALYs. In our example, the WCC vector, **wcc**, is the same for both costs and QALYs; thus, only one vector, `v_wcc`, is required.

```
## Vector with cycles
v_cycles <- seq(1, n_cycles + 1)
## Generate 2/3 and 4/3 multipliers for even and odd entries, respectively
v_wcc <- ((v_cycles %% 2)==0)*(2/3) + ((v_cycles %% 2)!=0)*(4/3)
## Substitute 1/3 in first and last entries
v_wcc[1] <- v_wcc[n_cycles + 1] <- 1/3
```

The discount vectors, $\mathbf{d}$, for costs and QALYs for the Sick-Sicker model, `v_dwc` and `v_dwe`, respectively, are

```
# Discount weight for effects
v_dwe <- 1 / ((1 + d_e) ^ (0:(n_cycles)))
# Discount weight for costs
v_dwc <- 1 / ((1 + d_c) ^ (0:(n_cycles)))
```

To account for both discounting and WCC, we incorporate **wcc** in equation (9) using an element-wise multiplication with $\mathbf{d}$, indicated by the $\odot$ symbol, such that

$$y = \mathbf{y}^{'} (\mathbf{d} \odot \mathbf{wcc}) . \tag{9}$$

The total expected discounted costs and QALYs under all four strategies accounting for WCC, y, is obtained

by applying Equation (9) to the expected outcomes accounting for transition rewards.

```
### For SoC
## QALYs
n_tot_qaly_SoC <- t(v_qaly_SoC) %*% (v_dwe * v_wcc)
## Costs
n_tot_cost_SoC <- t(v_cost_SoC) %*% (v_dwc * v_wcc)
### For Strategy A
## QALYs
n_tot_qaly_strA <- t(v_qaly_strA) %*% (v_dwe * v_wcc)
## Costs
n_tot_cost_strA <- t(v_cost_strA) %*% (v_dwc * v_wcc)
### For Strategy B
## QALYs
n_tot_qaly_strB <- t(v_qaly_strB) %*% (v_dwe * v_wcc)
## Costs
n_tot_cost_strB <- t(v_cost_strB) %*% (v_dwc * v_wcc)
### For Strategy AB
## QALYs
n_tot_qaly_strAB <- t(v_qaly_strAB) %*% (v_dwe * v_wcc)
## Costs
n_tot_cost_strAB <- t(v_cost_strAB) %*% (v_dwc * v_wcc)
```

The total expected discounted QALYs and costs for the simulation-time-dependent Sick-Sicker model under the four strategies accounting for WCC are shown in Table 3.

Table 3: Total expected discounted QALYs and costs per average individual in the cohort of the simulation-time-dependent Sick-Sicker model by strategy accounting for within-cycle correction .

	Costs	QALYs
Standard of care	$114,560	19.142
Strategy A	$211,911	19.840
Strategy B	$194,481	20.480
Strategy AB	$282,370	21.302

6 Incremental cost-effectiveness ratios (ICERs)

To conduct the cost-effectiveness analysis, we follow the coding approach described in the introductory cSTM tutorial.[7] We combine the total expected discounted costs and QALYs for all four strategies into outcome-specific vectors, `v_cost_str` for costs and `v_qaly_str` for QALYs. We use the R package `dampack` (https://cran.r-project.org/web/packages/dampack/)[22] to calculate the incremental costs and effectiveness and the incremental cost-effectiveness ratio (ICER) between non-dominated strategies. `dampack` organizes and formats the results as a data frame,`df_cea`, that can be printed as a formatted table.

```r
### Vector of costs
v_cost_str <- c(n_tot_cost_SoC, n_tot_cost_strA, n_tot_cost_strB, n_tot_cost_strAB)
### Vector of effectiveness
v_qaly_str <- c(n_tot_qaly_SoC, n_tot_qaly_strA, n_tot_qaly_strB, n_tot_qaly_strAB)

### Calculate incremental cost-effectiveness ratios (ICERs)
df_cea <- dampack::calculate_icers(cost = v_cost_str,
                                   effect = v_qaly_str,
                                   strategies = v_names_str)
```

In terms of their costs and effectiveness, SoC is the least costly and effective strategy, followed by Strategy B producing an expected benefit of 1.338 QALYs per individual for an additional expected cost of $79,920 with an ICER of $59,726/QALY followed by Strategy AB with an ICER $106,927/QALY. Strategy A is a dominated strategy. The results of the CEA of the simulation-time-dependent Sick-Sicker model are presented in Table 4. The non-dominated strategies, SoC, B, and AB, form the cost-effectiveness efficient frontier of the CEA based on the simulation-time-dependent Sick-Sicker model (Figure 9).

Table 4: Cost-effectiveness analysis results for the simulation-time-dependent Sick-Sicker model. ND: Non-dominated strategy; D: Dominated strategy.

Strategy	Costs ($)	QALYs	Incremental Costs ($)	Incremental QALYs	ICER ($/QALY)	Status
Standard of care	114,560	19.142	NA	NA	NA	ND
Strategy B	194,481	20.480	79,920	1.338	59,726	ND
Strategy AB	282,370	21.302	87,890	0.822	106,927	ND
Strategy A	211,911	19.840	NA	NA	NA	D

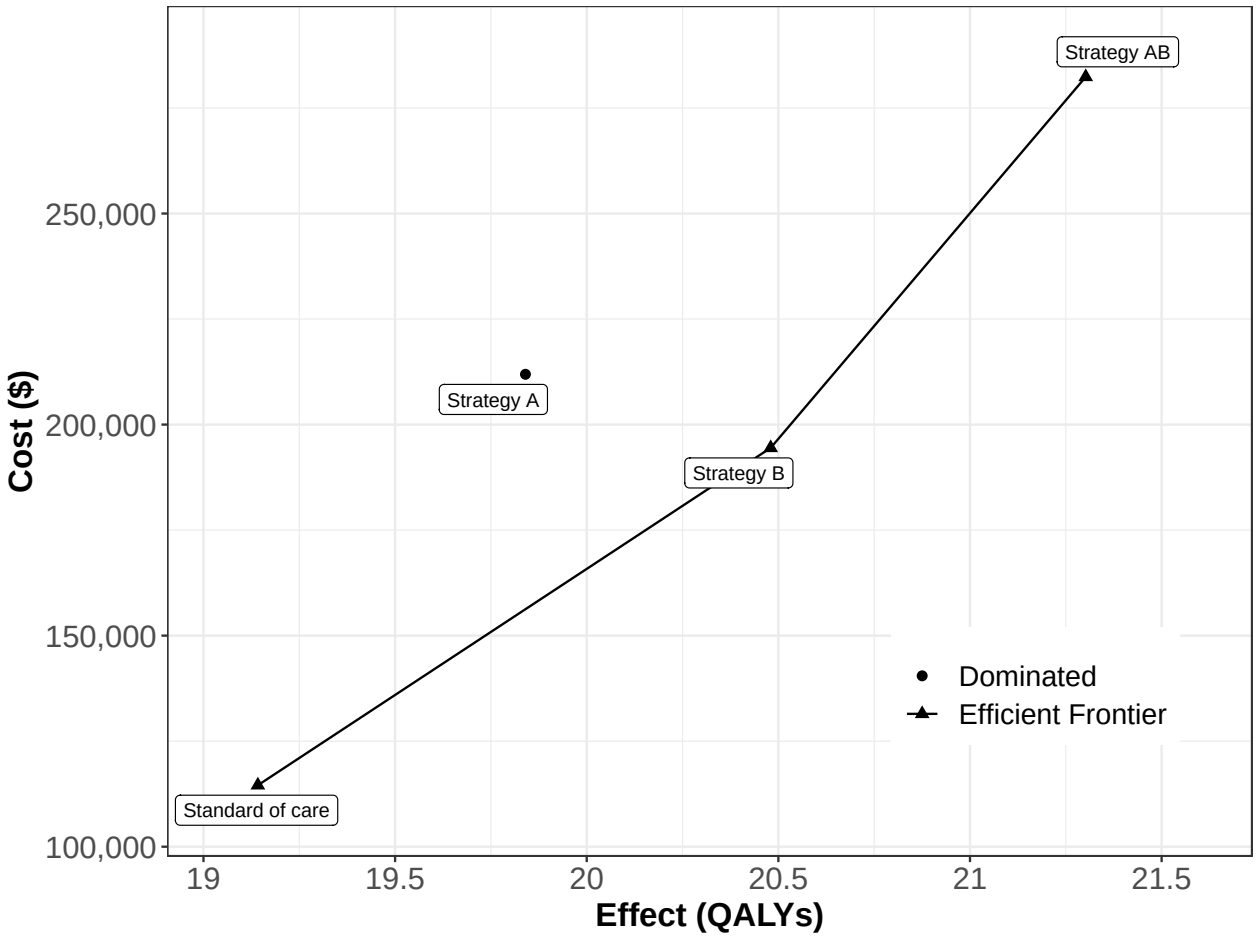

Figure 9: Cost-effectiveness efficient frontier of all four strategies for the simulation-time-dependent Sick-Sicker model.

7 Probabilistic sensitivity analysis

We conducted a probabilistic sensitivity analysis (PSA) to quantify the effect of model parameter uncertainty on cost-effectiveness outcomes.[23] In a PSA, we randomly draw parameter sets from distributions that reflect the current uncertainty in model parameter estimates. The parameters' distributions and their values are described in Table 2 and more detail in the Supplementary Material. We compute model outcomes for each sampled set of parameter values (e.g., total discounted cost and QALYs) for each strategy. We follow the steps to conduct a PSA from a previously published article describing the Decision Analysis in R for technologies in Health (DARTH) coding framework.[12]

To conduct the PSA of the CEA using the simulation time-dependent Sick-Sicker cSTM, we sampled 1,000 parameter sets. For each set, we computed the total discounted costs and QALYs of each simulated strategy. Results from a PSA can be represented in various ways. For example, the joint distribution, 95% confidence ellipse, and the expected values of the total discounted costs and QALYs for each strategy can be plotted in a cost-effectiveness (CE) scatter plot (Figure 10),[24] where each of the 4,000 simulations (i.e., 1,000 combinations of total discounted expected costs and QALYs for each of the four strategies) are plotted as a point in the graph. The CE scatter plot for the CEA using the simulation-time-dependent model shows that strategy

AB has the highest expected costs and QALYs. Standard of care has the lowest expected cost and QALYs. Strategy B is more effective and least costly than strategy A. And therefore, strategy A is strongly dominated by strategy B.

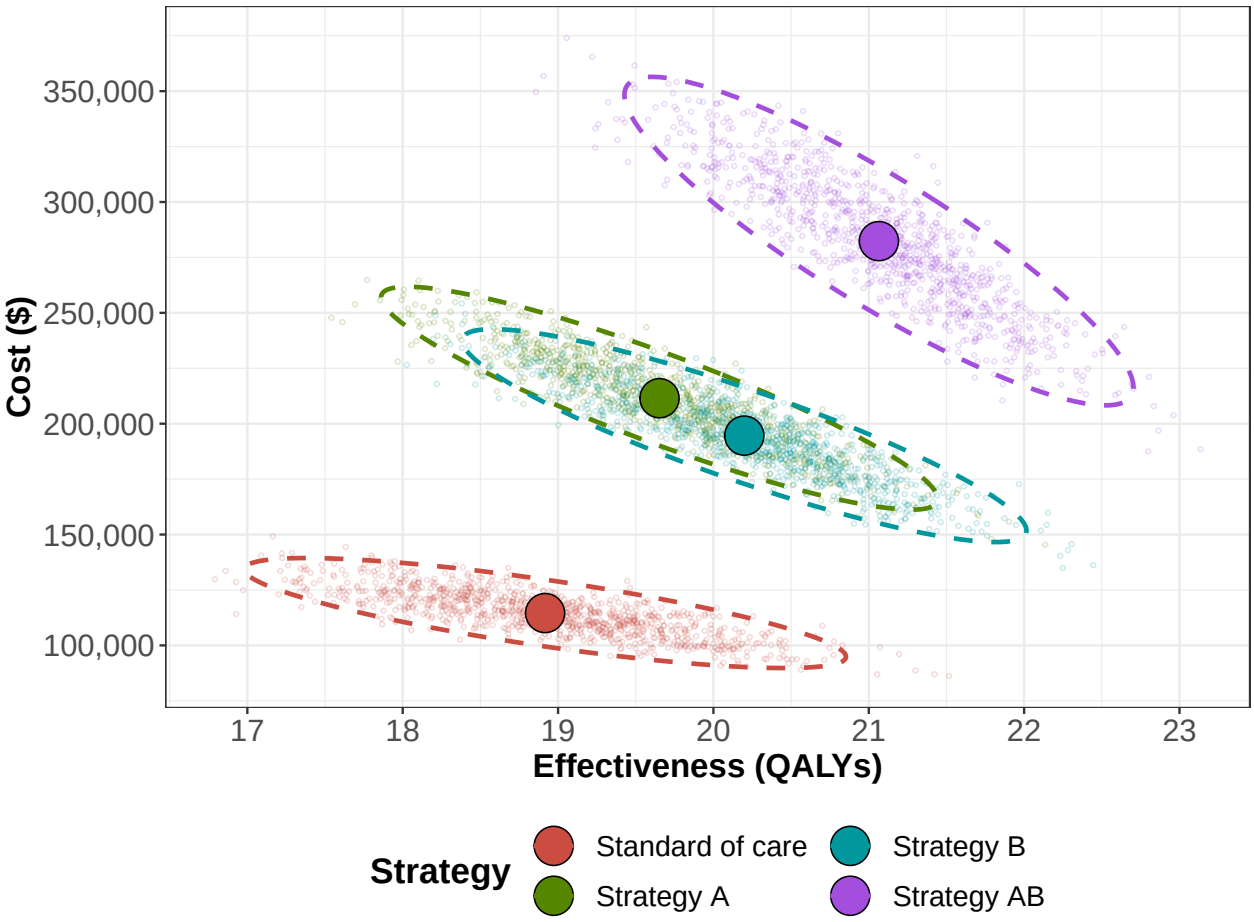

Figure 10: Cost-effectiveness scatter plot.

In Figure 11, we present the cost-effectiveness acceptability curves (CEACs) showing the probability that each strategy is cost-effective, and the cost-effectiveness frontier (CEAF), which shows the strategy with the highest expected net monetary benefit (NMB), over a range of willingness-to-pay (WTP) thresholds. Each strategy's NMB is computed using NMB = QALY $\times$ WTP $-$ Cost[25] for each PSA sample. At WTP thresholds less than \$65,000 per QALY, SoC is the strategy with the highest probability of being cost-effective and the highest expected NMB. Strategy B has the highest probability of being cost-effective and the highest expected NMB for WTP thresholds greater than \$65,000 and smaller than \$105,000 per QALY. Strategy AB, has the highest expected NMB for WTP thresholds greater than or equal to \$105,000 and is the strategy with the highest probability of being cost-effective.

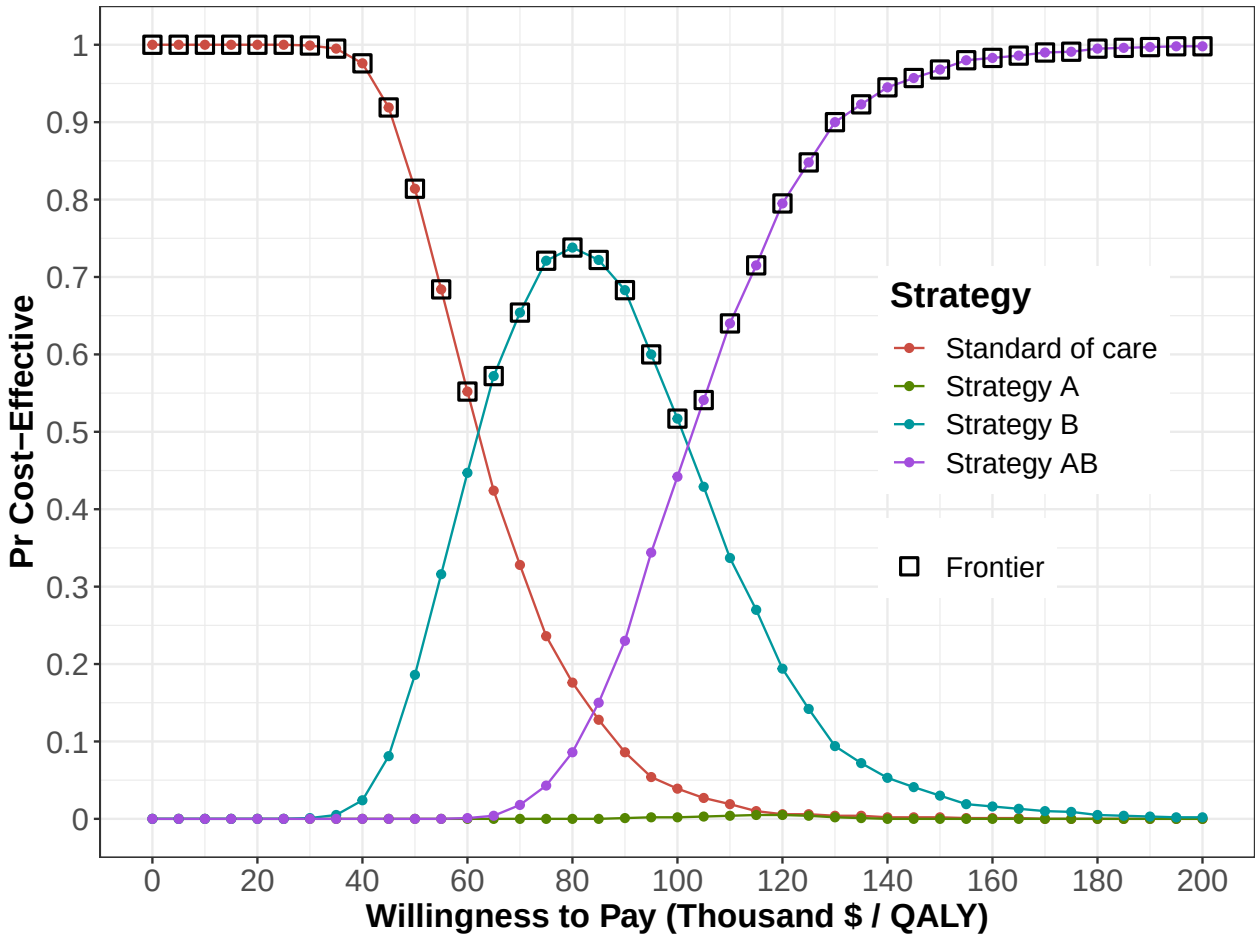

Figure 11: Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF).

The CEAC and CEAF do not show the magnitude of the expected net benefit lost (i.e., expected loss) when the chosen strategy is not the cost-effective strategy in all the samples of the PSA. We quantify expected loss from each strategy over a range of WTP thresholds with the expected loss curves (ELCs) to complement these results (Figure 12). The expected loss considers both the probability of making the wrong decision and the magnitude of the loss due to this decision, representing the foregone benefits of choosing a suboptimal strategy. The expected loss of the optimal strategy represents the lowest envelope of the ELCs because, given current information, the loss cannot be minimized further. The lower envelope also represents the expected value of perfect information (EVPI), which quantifies the value of eliminating parameter uncertainty. The strategy SoC has the lowest expected loss for WTP thresholds less than $60,000 per QALY, strategy B has the lowest expected loss for WTP threshold greater than or equal to $65,000 and less than $105,000. Strategy AB has the lowest expected loss for WTP threshold greater than or equal to $105,000 per QALY. At a WTP threshold of $65,000 per QALY, the EVPI is highest at $6,953. For a more detailed description of these outputs and the R code to generate them, we refer the reader to a previous publication by our group.[26]

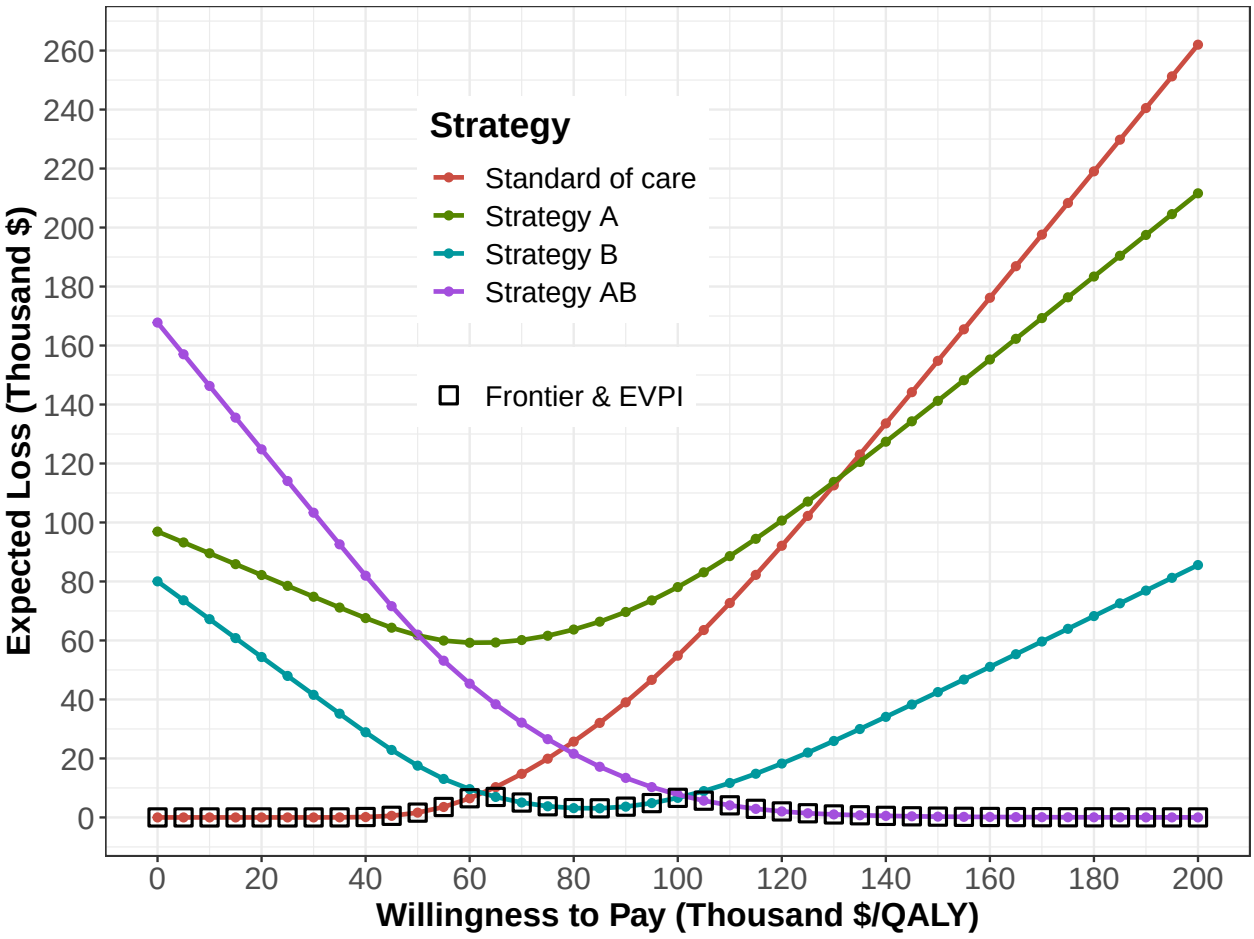

Figure 12: Expected loss curves (ELCs) and expected value of perfect information (EVPI) generated from the probabilistic sensitivity analysis (PSA) output.

8 Discussion

In this tutorial, we conceptualize time-dependent cSTMs with their mathematical description and a walk-through of their implementation for a CEA in R using the Sick-Sicker example. We described two types of time-dependency: dependence on the time since the start of the simulation (simulation-time dependency) or on time spent in a health state (state-residence time dependency). We also illustrate how to generate various epidemiological measures from the model, incorporate transition rewards in CEAs, and conduct a PSA.

We implemented simulation-time dependency by expanding the transition probability matrix into a transition probability array, where the third dimension captures time. However, there are alternative implementations of simulation-time dependency in cSTMs. For example, the model could be coded such that the time-varying elements of the transition probability matrix P_t are updated at each time point t as the simulation is run. This would eliminate the need for the transition probability array `a_P`, reducing computer memory requirements. But this comes at the expense of increasing the number of operations at every cycle, potentially slowing down the simulation.

We incorporated state-residence time dependency using tunnel states by expanding the corresponding health

states on the first and second dimensions of the 3-dimensional array to account for time spent in the current state in addition to simulation-time dependence. Another approach to account for state-residence time dependency is to use a 3-dimensional transition probability array with dimensions for the current state, future state, and time in the current state.[27] However, in examples combining simulation-time and state-residence time dependencies, this would necessitate a 4-dimensional array, which may be challenging to index.

It is the case that any time-varying feature in a discrete-time model can most generally be implemented as tunnel states with, at the extreme, every state having a different tunnel state for each time step. The cohort would then progressively move through these tunnel states to capture their progression through time, and the model features (e.g., transition probabilities, costs or utilities) that change over time. Using time-varying transition probabilities is a shortcut that is possible when the cohort experiences these time-varying processes simultaneously as a function of the time from the simulation start. Even if the time-varying process has a different periodicity than the cycle length, either tunnel states or time-varying transition probabilities can be used to capture these effects. However, this time-varying process needs to be represented or approximated over an integer number of cycle lengths.

As described in the introductory tutorial, cSTMs are recommended when the number of states is considered "not too large".[15] Incorporating time-dependency in cSTMs using the provided approaches requires expanding the number of states by creating a multidimensional array for simulation-time dependency and/or creating tunnel states for state-residence time dependency, increasing the amount of computer memory (RAM) required. It is possible to build reasonably complex time-dependent cSTMs in R as long as there is sufficient RAM to store the transition probability array and outputs of interest. For example, a typical PC with 8GB of RAM can handle a transition probability array of about 1,000 states and 600 time-cycle slices. However, if a high degree of granularity is desired, the dimensions of these data objects can grow quickly; if the required number of states gets too large and difficult to code, it may be preferable to use a stochastic (Monte Carlo) version of the state-transition model – often called individual-based state-transition models (iSTM) or microsimulation models – rather than a cohort simulation model.[15] In an iSTM, the risks and rewards of simulated individuals need not depend only on a current health state; they may also depend on an individual's characteristics and attributes. In addition, modelers can store health state history and other events over time for each individual to determine the risk of new events and corresponding costs and effects. Thus, we recommend carefully considering the required model structure before implementing it. An iSTM will also require additional functions to describe the dependency of transition probabilities and rewards on individuals' history. In a previous tutorial, we showed how to construct these functions for an iSTM using the Sick-Sicker example.[28]

We also described the concept and implementation of transition rewards. While these event-driven impacts could instead be captured by expanding the model state-space to include an initial disease state or a pre-death, end-of-life state, we can avoid state-space expansion by calculating incurred rewards based on the proportion of the cohort making the relevant transition. For implementation, this requires storing not just the cohort trace, which reflects how many individuals are in each state at a given cycle, but also a cohort state-transition array, which records how many individuals are making each possible transition at a given cycle.[9]

In summary, this tutorial extends our conceptualization of time-independent cSTMs to allow for time dependency. It provides a step-by-step guide to implementing time-dependent cSTMs in R to generate epidemiological and economic measures, account for transition rewards, and conduct a CEA and a corresponding PSA. We hope that health decision scientists and health economists find this tutorial helpful in developing their cSTMs in a more flexible, efficient, and open-source manner. Ultimately, our goal is to facilitate R in

health economic evaluations research with the overall aim to increase model transparency and reproducibility.

9 Acknowledgements

Dr. Alarid-Escudero was supported by grants U01-CA199335 and U01-CA253913 from the National Cancer Institute (NCI) as part of the Cancer Intervention and Surveillance Modeling Network (CISNET), and a grant by the Gordon and Betty Moore Foundation. Miss Krijkamp was supported by the Society for Medical Decision Making (SMDM) fellowship through a grant by the Gordon and Betty Moore Foundation (GBMF7853). Dr. Enns was supported by a grant from the National Institute of Allergy and Infectious Diseases of the National Institutes of Health under award no. K25AI118476. Dr. Hunink received research funding from the American Diabetes Association, the Netherlands Organization for Health Research and Development, the German Innovation Fund, Netherlands Educational Grant ("Studie Voorschot Middelen"), and the Gordon and Betty Moore Foundation. Dr. Jalal was supported by a grant from the National Institute on Drug Abuse of the National Institute of Health under award no. K01DA048985. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. The funding agencies had no role in the design of the study, interpretation of results, or writing of the manuscript. The funding agreement ensured the authors' independence in designing the study, interpreting the data, writing, and publishing the report. We also want to thank the anonymous reviewers of *Medical Decision Making* for their valuable suggestions and the students who took our classes where we refined these materials.

References

1. Suijkerbuijk AWM, Van Hoek AJ, Koopsen J, et al. Cost-effectiveness of screening for chronic hepatitis B and C among migrant populations in a low endemic country. *PLoS ONE* 2018; 13: 1–16.

2. Sathianathen NJ, Konety BR, Alarid-Escudero F, et al. Cost-effectiveness Analysis of Active Surveillance Strategies for Men with Low-risk Prostate Cancer. *European Urology*; 75: 910–917, https://linkinghub.elsevier.com/retrieve/pii/S0302283818308534 (2019).

3. Lu S, Yu Y, Fu S, et al. Cost-effectiveness of ALK testing and first-line crizotinib therapy for non-small-cell lung cancer in China. *PLoS ONE* 2018; 13: 1–12.

4. Djatche LM, Varga S, Lieberthal RD. Cost-Effectiveness of Aspirin Adherence for Secondary Prevention of Cardiovascular Events. *PharmacoEconomics - Open*; 2: 371–380, https://doi.org/10.1007/s41669-018-0075-2 (2018).

5. Pershing S, Enns EA, Matesic B, et al. Cost-Effectiveness of Treatment of Diabetic Macular Edema. *Annals of Internal Medicine* 2014; 160: 18–29.

6. Smith-Spangler CM, Juusola JL, Enns EA, et al. Population Strategies to Decrease Sodium Intake and the Burden of Cardiovascular Disease: A Cost-Effectiveness Analysis. *Annals of Internal Medicine*; 152: 481–487, http://annals.org/article.aspx?articleid=745729 (2010).

7. Alarid-Escudero F, Krijkamp E, Enns EA, et al. An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. *arXiv:200107824v3*; 1–26, https://arxiv.org/abs/2001.07824 (2021).

8. Snowsill T. A New Method for Model-Based Health Economic Evaluation Utilizing and Extending Moment-Generating Functions. *Medical Decision Making*; 39: 523–539, http://journals.sagepub.com/doi/10.1177/0272989X19860119 (2019).

9. Krijkamp EM, Alarid-Escudero F, Enns E, et al. A Multidimensional Array Representation of State-Transition Model Dynamics. *Medical Decision Making* 2019; In Press.

10. Jalal H, Pechlivanoglou P, Krijkamp E, et al. An Overview of R in Health Decision Sciences. *Medical Decision Making*; 37: 735–746, http://journals.sagepub.com/doi/10.1177/0272989X16686559 (2017).

11. Enns EA, Cipriano LE, Simons CT, et al. Identifying Best-Fitting Inputs in Health-Economic Model Calibration: A Pareto Frontier Approach. *Medical Decision Making*; 35: 170–182, http://www.ncbi.nlm.nih.gov/pubmed/24799456 (2015).

12. Alarid-Escudero F, Krijkamp E, Pechlivanoglou P, et al. A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling. *PharmacoEconomics*; 37: 1329–1339, https://doi.org/10.1007/s40273-019-00837-x (2019).

13. Arias E, Heron M, Xu J. United States Life Tables, 2014. *National Vital Statistics Reports*; 66: 63, https://www.cdc.gov/nchs/data/nvsr/nvsr66/nvsr66%7B/_%7D04.pdf (2017).

14. Diaby V, Adunlin G, Montero AJ. Survival modeling for the estimation of transition probabilities in model-based economic evaluations in the absence of individual patient data: A tutorial. *PharmacoEconomics* 2014; 32: 101–108.

15. Siebert U, Alagoz O, Bayoumi AM, et al. State-Transition Modeling: A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force-3. *Medical Decision Making*; 32: 690–700, http://mdm.sagepub.com/cgi/doi/10.1177/0272989X12455463 (2012).

16. Lee ET, Wang JW. *Statistical methods for Survival Data Analysis*. 3rd ed. Hoboken, NJ: Wiley, 2003.

17. Klein JP, Moeschberger ML. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. Springer-Verlag, http://www.springer.com/statistics/life+sciences,+medicine+%7B/&%7D+health/book/978-0-387-95399-1 (2003).

18. Rothman KJ, Greenland S, Lash TL. *Modern Epidemiology*. 3rd ed. Lippincott Williams & Wilkins, 2008.

19. Keiding N. Age-Specific Incidence and Prevalence: A Statistical Perspective. *Journal of the Royal Statistical Society Series A (Statistics in Society)* 1991; 154: 371–412.

20. Elbasha EH, Chhatwal J. Theoretical foundations and practical applications of within-cycle correction methods. *Medical Decision Making* 2016; 36: 115–131.

21. Elbasha EH, Chhatwal J. Myths and misconceptions of within-cycle correction: a guide for modelers and decision makers. *PharmacoEconomics* 2016; 34: 13–22.

22. Alarid-Escudero F, Knowlton G, Easterly CA, et al. Decision analytic modeling package (dampack), https://cran.r-project.org/web/packages/dampack/%20https://github.com/DARTH-git/dampack (2021).

23. Briggs AH, Weinstein MC, Fenwick EAL, et al. Model Parameter Estimation and Uncertainty Analysis: A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force Working Group-6. *Medical Decision Making* 2012; 32: 722–732.

24. Briggs AH, Goeree R, Blackhouse G, et al. Probabilistic Analysis of Cost-Effectiveness Models: Choosing between Treatment Strategies for Gastroesophageal Reflux Disease. *Medical Decision Making*; 22: 290–308, http://mdm.sagepub.com/cgi/doi/10.1177/027298902400448867 (2002).

25. Stinnett AA, Mullahy J. Net Health Benefits: A New Framework for the Analysis of Uncertainty in Cost-Effectiveness Analysis. *Medical Decision Making*; 18: S68–S80, http://mdm.sagepub.com/cgi/doi/10.1177/0272989X9801800209 (1998).

26. Alarid-Escudero F, Enns EA, Kuntz KM, et al. "Time Traveling Is Just Too Dangerous" But Some Methods Are Worth Revisiting: The Advantages of Expected Loss Curves Over Cost-Effectiveness Acceptability Curves and Frontier. *Value in Health* 2019; 22: 611–618.

27. Hawkins N, Sculpher M, Epstein D. Cost-effectiveness analysis of treatments for chronic disease: Using R to incorporate time dependency of treatment response. *Medical Decision Making*; 25: 511–9, http://www.ncbi.nlm.nih.gov/pubmed/16160207 (2005).

28. Krijkamp EM, Alarid-Escudero F, Enns EA, et al. Microsimulation Modeling for Health Decision Sciences Using R: A Tutorial. *Medical Decision Making*; 38: 400–422, http://journals.sagepub.com/doi/10.1177/0272989X18754513 (2018).