A Tutorial on Time-Dependent Cohort State-Transition Models in R

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[**]

2021-04-06

Abstract

In an accompanying tutorial, we show how to implement time-independent cohort state-transition models (cSTMs) to conduct cost-effectiveness analyses (CEA) in R. In this tutorial, we introduce time-dependent cSTMs, where transition probabilities and rewards can depend on the time since the start of the simulation (simulation time-dependency) or on time spent in a health state (state residence). We illustrate how to conduct a CEA of multiple strategies using a time-dependent cSTM using a previously published cSTM, including probabilistic sensitivity analyses. We also demonstrate how to compute various epidemiological outcomes of interest, such as survival probability and prevalence. We provide a link to a public repository with all the R code described in this tutorial that readers can use to replicate the example or adapt for various decision modeling applications.

[*]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) are a type of decision model used to evaluate different health policy and clinical strategies, such as screening and surveillance programs,[1,2] diagnostic procedures,[3] disease management programs,[4] and interventions.[5,6]. cSTMs can be either time-independent or depend on either the time since the start of the simulation (simulation time-dependency) or on time spent in a health state (state residence). In an accompanying tutorial,[7] we describe the components of a time-independent cSTM, illustrate the implementation of these components and conduct a cost-effectiveness analysis (CEA) in R, a statistical software with an increasing use in health decision sciences.[8] This tutorial focuses on time-dependent cSTM. We cover both simulation time-dependency and time dependency on state residence. We first expand the Sick-Sicker model described in the accompanying tutorial[7] to account for age dependency and time dependency on state residence and allowing for transition rewards. We then provide a description of each of the components of a time-dependent cSTM, illustrate the implementation of these components by expanding the example described in the accompanying tutorial, conduct a CEA comparing four strategies, including a probabilistic sensitivity analysis. We also demonstrate how to compute various epidemiological outcomes of interest, such as survival probability and prevalence. 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). We assume that the reader is familiar with the basics of decision modeling and coding a time-independent cSTM in R. Thus, a prior introduction to R and linear algebra for decision modelers is recommended.

2 Time-dependent cSTMs

Time-dependency in cSTMs means that transition probabilities or rewards are not fixed over time. Time dependency in cSTMs is desirable to represent disease processes over relatively long periods accurately. In this tutorial, we describe two types of time dependency: (1) dependency on time since simulation starts ("simulation-time dependency"), with age-specific background mortality being an important example,[9] and (2) state-residence time dependency, which means that transition probabilities or rewards depend on the duration of time the cohort spends in a given state. Time-dependent cSTMs can capture the increasing age-specific background mortality as the cohort ages (age dependency) and dependency on the amount of time spent in a given state (state residence). We cover both types in this tutorial.

2.1 Simulation-time dependency

Transition probabilities that depend on the time since the start of the simulation (simulation-time dependency) can be accounted for by the transition probability matrix being a function of time, P_t. From here after we will refer to time since the start of the simulation as *simulation-time dependency.* The elements of P_t are the transition probabilities of moving from state i to state j as a function of time t, $p_{[i,j,t]}$, where $\{i, j\} = 1, \ldots, n_S$ and $t = 0, \ldots, n_T$

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

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

The state vector at 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, such that,

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

where $\mathbf{m}_1$ is computed from $\mathbf{m}_0$ and represents the initial state vector with the distribution of the cohort across all health states at the start of the simulation (cycle 0). Then, this equation 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 outcomes, such as prevalence and survival probability over time, and economic outcomes, such as cumulative resource use and costs.

2.2 Time dependency on state residence

A slightly more complex time dependency is *state residence*, for which transition probabilities or rewards depending on the time spent in a given state. One way to account for state-residence dependency is to expand the number of states with as many transient states as the number of cycles for which state residency is required. These transient states are referred to as *tunnel* states where the cohort can stay for only one cycle and either transition to the next tunnel state or completely exit the tunnel. The set of tunnel states then essentially equals the amount of time that the cohort has spent in the tunnel representing a specific health state.

If state-residence in a given state lasts $n_{tunnels}$ cycles, such a state needs to be expanded into $n_{tunnels}$ states, and the transition probability matrix also needs to be expanded to incorporate these additional states. This will result in a transition probability matrix (or array if the transition probabilities are also dependent on simulation time) of dimensions $n_{S_{tunnels}} \times n_{S_{tunnels}}$ (or $n_{S_{tunnels}} \times n_{S_{tunnels}} \times n_T$), where $n_{S_{tunnels}}$ is the total number of health states ($n_{S_{tunnels}} = n_S + n_{tunnels} - 1$).

Table 1 describes the elements related to the core components of time-dependent cSTMs and their suggested R code names. For a more detailed description of the variable types, data structure, R name for all cSTM elements, please see the Supplementary Material.

Table 1: Components of a time-dependent cSTM 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`
$n_{S_{tunnels}}$	Number of states including tunnel states	`n_states_tunnels`
$\mathbf{m}_{tunnels_0}$	Initial state vector for the model with tunnel states	`v_s_init_tunnels`

3 Case study: Time-dependent Sick-Sicker model

We expand the time-independent 4-state “Sick-Sicker” model[7] to account for time dependency. We first expand it to account for age dependency as an example of simulation time dependency. In a following section, we expand the age-dependent cSTM to account for time dependency in state residence With the time-dependent cSTM, we conduct a CEA of different strategies accounting for transition rewards. Figure 1 represents the state-transition diagram of the Sick-Sicker model.

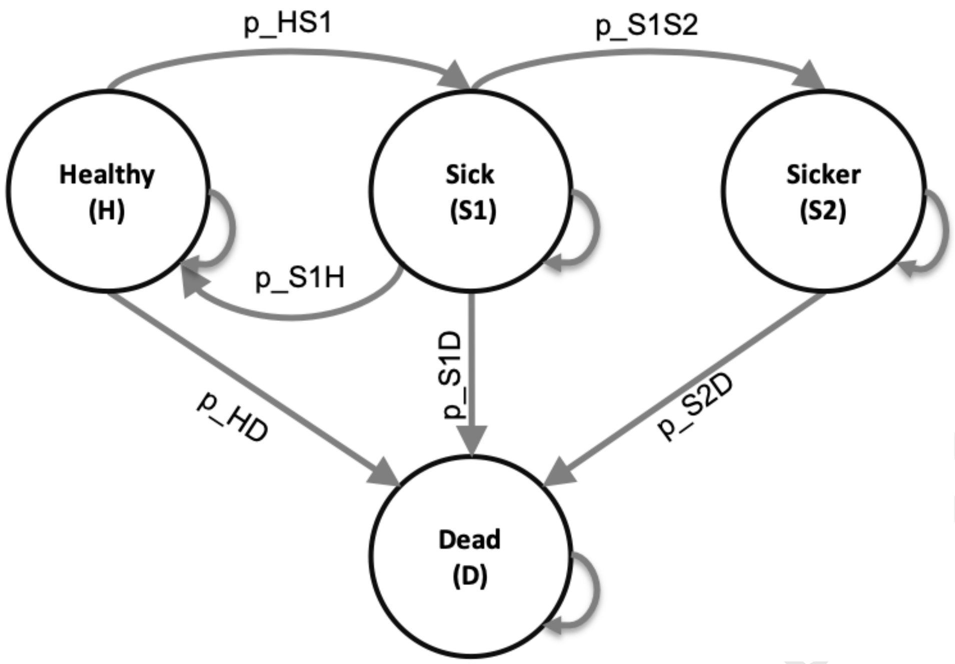

Figure 1: State-transition diagram of the Sick-Sicker cohort state-transition model with the name of the health states and possible transitions with their corresponding transition probabilities.

The model simulates a cohort to quantify the expected costs and quality-adjusted life years (QALYs) for individuals at risk of a hypothetical disease that has two stages: "Sick" and "Sicker".

All the parameters of the Sick-Sicker model and the corresponding R variable names are presented in Table 2 and follow the notation described in the DARTH coding framework.[10]

In the Sick-Sicker model, we simulate a hypothetical cohort of 25-year-old individuals over their lifetime (until a maximum age of 100 years) who all start in the "Healthy" state (denoted "H"). Meaning 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`. Healthy individuals are at risk of developing the disease when they transition to the "Sick" state (denoted by "S1"). 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 acute impacts of developing the illness. Sick individuals are at risk of further progressing to a more severe disease stage, the "Sicker" health state (denoted by "S2"). In the state-residence time-dependent model, the risk progressing from S1 to S2 is a function of the duration of time spent in S1. Individuals in S1 can recover and return to H. However, once individuals reach S2, they cannot recover; that is, the probability of transitioning to S1 or H from S2 is zero. Individuals in H face age-specific background mortality, meaning that we model all-cause mortality as a function of age. 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 mortality hazard rate. Individuals in S1 and S2 also experience increased health care costs (in addition to a one-time cost of $1,000) and reduced QoL compared to individuals in H. Once simulated individuals die, they transition to the absorbing "Dead" state (denoted by "D"), where they remain, and incur a one-time cost of $2,000 (`ic_D`) that reflects the expected acute care preceding death. 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 discount both costs and QALYs at an annual rate of 3%.

We are interested in evaluating 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 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`) and costs $12,000 per year (`c_trtA`).[11] 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 only the rate of Sick individuals progressing to the Sicker state with a hazard ratio (HR) of 0.6 (`hr_S1S2_trtB`) and costs $13,000 per year (`c_trtB`) and does not affect QoL. Strategy AB involves administering both treatments A and B.

We assume that it is not possible to distinguish between Sick and Sicker patients; therefore, individuals in both disease states receive the treatments. Note that for strategy A, the model has the same structure and identical transition probabilities to SoC. The only difference is the added cost of the treatment for S1 or 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.

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

Parameter	**R name**	**Base-case**	**Distribution**
Number of cycles ($n_c ycles$)	`n_cycles`	75 years	-
Names of health states (n)	`v_names_states`	H, S1, S2, D	-
Annual discount rate for costs	`d_c`	3%	-
Annual discount rate for QALYs	`d_e`	3%	-
Number of PSA samples (K)	`n_sim`	1,000	-
Annual transition 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			
- Age-dependent background mortality rate (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)

Parameter	R name	Base-case	Distribution
- 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	-
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
```

```r
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
```

3.1 Simulation-time dependency

To illustrate the implementation of this time dependency in the Sick-Sicker cSTM, we model all-cause mortality as a function of age. It is common to 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` with age-specific background mortality hazard rates for 0 to 100 year-olds obtained from the 2015 US life-tables.[12] To compute the transition probability from state H to state D, corresponding to cohort's age at each cycle, we transform $\mu(a)$ 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. We will run the Sick-Sicker model for 75 cycles for ages 25 through 100. 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, adding age dependency on background mortality results in age-dependent mortality in S1 and S2 as well as in H. 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 stage-specific mortality rates to probabilities to ensure 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 age-dependency into the transition probability matrix, we expand the dimensions of the matrix and create a 3-dimensional transition probability array, $\mathbf{P}$ and `a_P` in R, of dimensions $n_S \times n_S \times n_T$, where the first two dimensions 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.

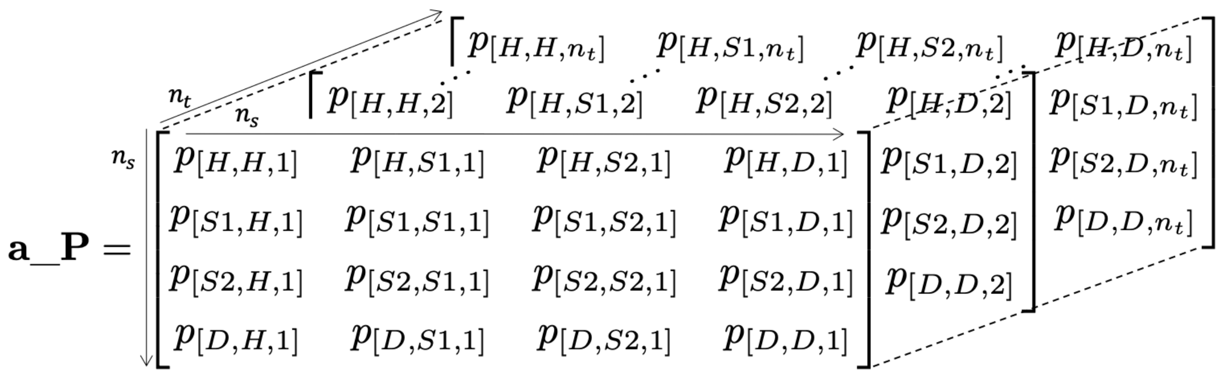
a_P =
n_t
n_s
n_s
$p_{[H,H,1]}$ $p_{[H,S1,1]}$ $p_{[H,S2,1]}$ $p_{[H,D,1]}$
$p_{[S1,H,1]}$ $p_{[S1,S1,1]}$ $p_{[S1,S2,1]}$ $p_{[S1,D,1]}$
$p_{[S2,H,1]}$ $p_{[S2,S1,1]}$ $p_{[S2,S2,1]}$ $p_{[S2,D,1]}$
$p_{[D,H,1]}$ $p_{[D,S1,1]}$ $p_{[D,S2,1]}$ $p_{[D,D,1]}$
$p_{[H,H,2]}$ $p_{[H,S1,2]}$ $p_{[H,S2,2]}$ $p_{[H,D,2]}$
$p_{[S1,D,2]}$ $p_{[S2,D,2]}$ $p_{[D,D,2]}$
$p_{[H,H,n_t]}$ $p_{[H,S1,n_t]}$ $p_{[H,S2,n_t]}$ $p_{[H,D,n_t]}$
$p_{[S1,D,n_t]}$ $p_{[S2,D,n_t]}$ $p_{[D,D,n_t]}$

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

First, we initialize a_P with a default value of zero for all transition probabilities.

```r
# Initialize the transition probability array
a_P <- 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 with the corresponding transition probabilities of the cohort under the SoC strategy is comparable with filling m_P for a time-independent cSTM, accounting for the time dimension, which is represented by the third dimension of the array. However, this requires a slight modification of the code from the time-independent cSTM. The code below illustrates how to assign age-dependent transition probabilities in the third dimension of the array. For those transitions that are constant over time, we only need to provide one transition probability. R replicates the value of such transitions as many times as the length of the time dimension, for our example $n_T + 1$ times.

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

As mentioned above, each of the elements on the third dimension of `a_P` correspond 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 using:

```
a_P[, , 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
```

Similar to the time-independent cSTM, treatment A does not change the transition probabilities. For treatment B, we first initialize the three-dimensional array of transition probabilities, `a_P_trtB` as a copy of `a_P` and update only the transition probabilities from S1 to S2 (i.e., `p_S1S2` is replaced with `p_S1S2_trtB`).

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

Once we create both three-dimensional transition probability arrays, we check they are valid using the functions `check_sum_of_transition_array` and `check_transition_probability`,which are provided in the `darthtools` package (https://github.com/DARTH-git/darthtools).

```
### Check if transition probability matrices are valid
## Check that transition probabilities are [0, 1]
check_transition_probability(a_P)
check_transition_probability(a_P_trtB)
## Check that all rows sum to 1
check_sum_of_transition_array(a_P,      n_states = n_states, n_cycles = n_cycles)
check_sum_of_transition_array(a_P_trtB, n_states = n_states, n_cycles = n_cycles)
```

For the time-dependent 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 cohorts under SoC strategy and strategy A because the transition probabilities are the same under both strategies, and by `m_M_trtB` for the cohorts under strategies B and AB because both transition probabilities under both strategies are affected by treatment B. 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 age-dependent cSTM, we initialize two new cohort trace matrices `m_M_ad` and m_M_ad_trtB.

```
## Initialize cohort trace for age-dependent (ad) cSTM under SoC
m_M_ad <- 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_ad[1, ] <- v_s_init
## Initialize cohort trace under treatment B
m_M_ad_trtB <- m_M_ad # structure and initial states remain the same.
```

We then use the matrix product to get the state vector at 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 a_P and a_P_trtB by t to obtain the cycle-specific transition probability matrices.

```
# Iterative solution of age-dependent cSTM
for(t in 1:n_cycles){
  # For SoC
  m_M_ad[t + 1, ] <- m_M_ad[t, ] %*% a_P[, , t]
  # For treatment B
  m_M_ad_trtB[t + 1, ] <- m_M_ad_trtB[t, ] %*% a_P_trtB[, , t]
}
```

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

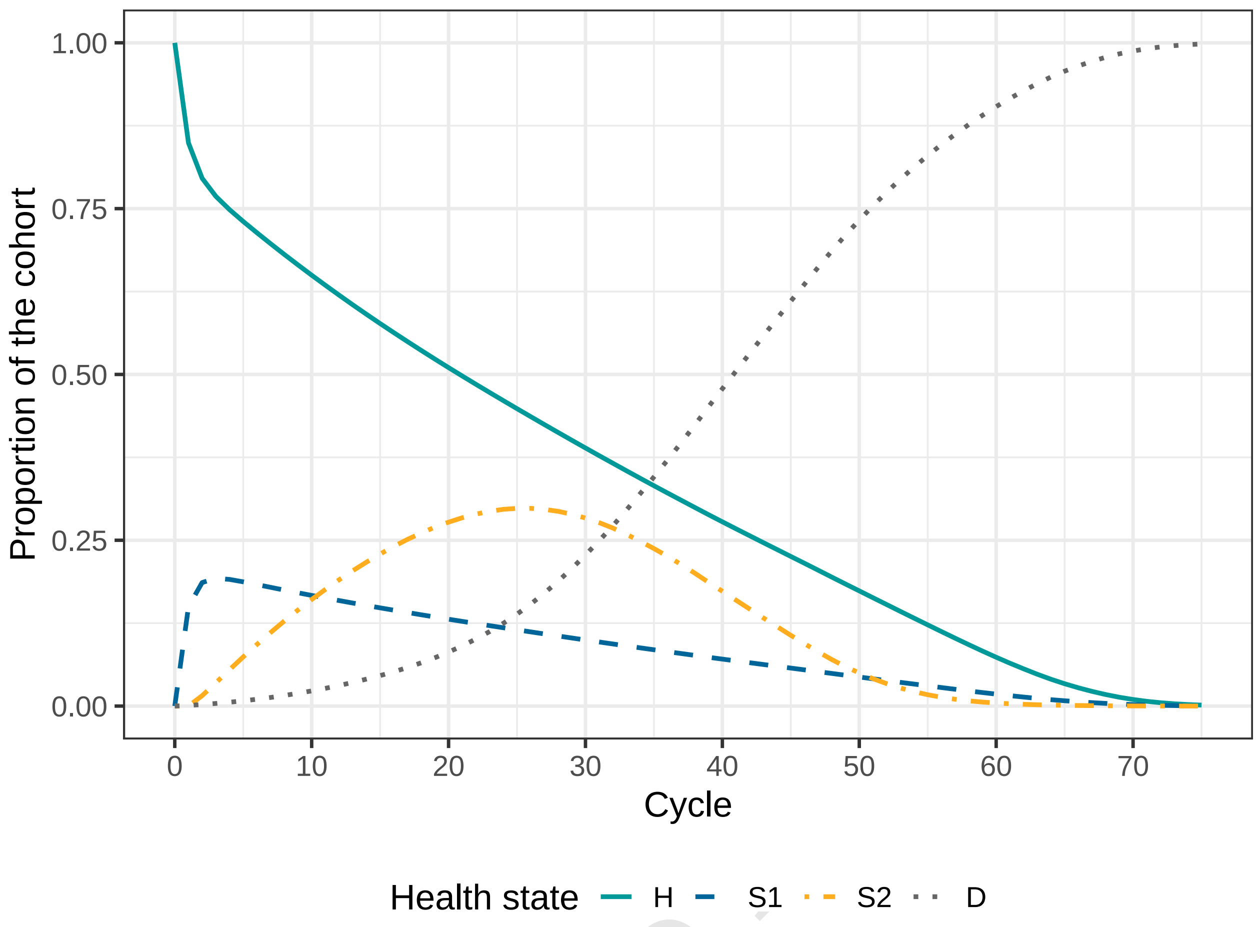

Figure 3: Cohort trace of the age-dependent cSTM under strategies SoC and A.

3.2 Time dependency on state residence

To illustrate state-residence dependency in the age-dependent Sick-Sicker model defined above, we assume the risk of progression from S1 to S2 increases the longer the cohort has been Sick. This increase follows a Weibull hazard function, defined as

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

where $\tau = 1, \ldots, n_{tunnels}$ represents the duration that the cohort has been in S1, and λ and γ are the scale and shape parameters of the Weibull function, respectively. We assume that state-residence dependency lasts the whole simulation (i.e., $n_{tunnels} = n_T$) 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 $n_{S_{tunnels}} = 78$.

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

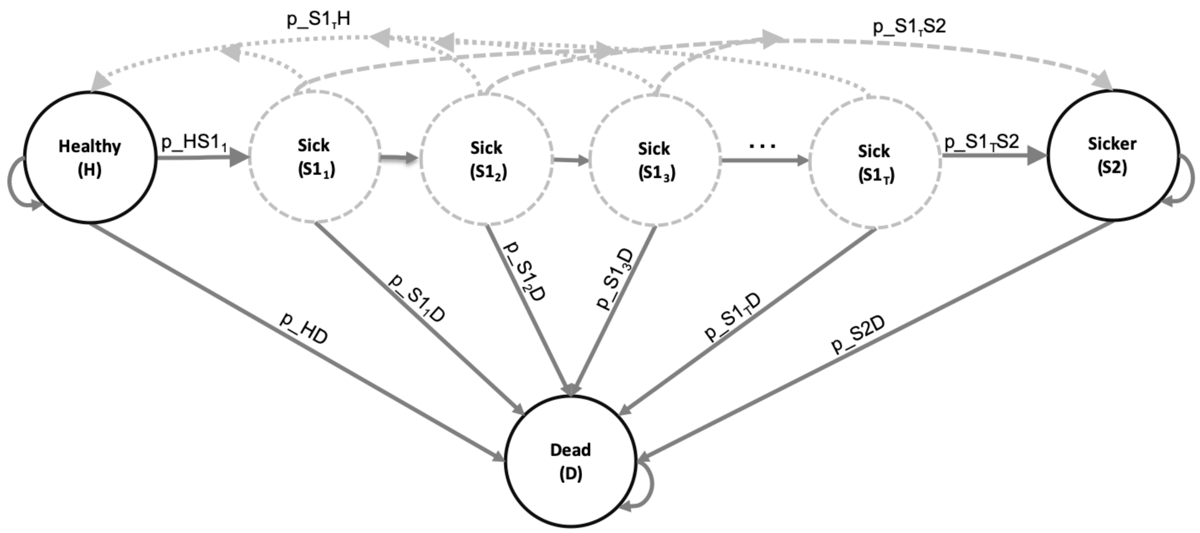

Figure 4: State-transition diagram of the Sick-Sicker model with tunnel states expanding the Sick state $(S1_1, S1_2, ..., S1_{n_{tunnels}})$.

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

```
## Number of tunnels
n_tunnel_size <- n_cycles
## Name for tunnel states of Sick 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)
```

Based on the updated parameters, the state-residence-dependent transition probability from Sick to Sicker based on a Weibull function, `v_p_S1S2_tunnels`, is:

```
# Weibull parameters
p_S1S2_scale <- 0.08 # scale
p_S1S2_shape  <- 1.1  # shape
# Weibull function
v_p_S1S2_tunnels <- p_S1S2_scale * p_S1S2_shape *
  (1:n_tunnel_size)^{p_S1S2_shape-1}
```

To adapt the 3-dimensional transition probability array to incorporate both age and state-residence dependence in the Sick-Sicker model under SoC, we first create an expanded 3-dimensional array accounting for tunnels, `a_P_tunnels`, of dimensions $n_{S_{tunnels}} \times n_{S_{tunnels}} \times n_T$. A visual representation of `a_P_tunnels` of the

Sick-Sicker model with tunnel states expanding the Sick state is shown in Figure 5.

```r
# Initialize array
a_P_tunnels <- 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)))
```

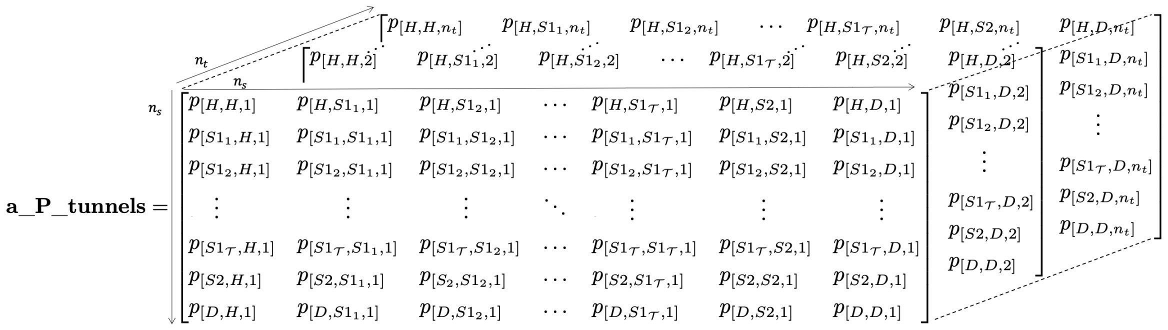

Figure 5: The 3-dimensional transition probability array of the Sick-Sicker model expanded to account for age-dependence and S1 state-residence using tunnel states.

Filling `a_P_tunnels` with the corresponding transition probabilities is similar to the `a_P` above, with the difference being that we now fill the transition probabilities from all the S1 tunnel states by iterating through all the tunnel states and assigning the corresponding disease progression transition probabilities.

```r
### Fill in array
## From H
a_P_tunnels["H", "H", ]                 <- (1 - v_p_HDage) * (1 - p_HS1)
a_P_tunnels["H", v_Sick_tunnel[1], ] <- (1 - v_p_HDage) * p_HS1
a_P_tunnels["H", "D", ]                 <- v_p_HDage
## From S1
for(i in 1:(n_tunnel_size - 1)){
  a_P_tunnels[v_Sick_tunnel[i], "H", ]  <- (1 - v_p_S1Dage) * p_S1H
  a_P_tunnels[v_Sick_tunnel[i],
              v_Sick_tunnel[i + 1], ]   <- (1 - v_p_S1Dage) *
                                           (1 - (p_S1H + v_p_S1S2_tunnels[i]))
  a_P_tunnels[v_Sick_tunnel[i], "S2", ] <- (1 - v_p_S1Dage) * v_p_S1S2_tunnels[i]
  a_P_tunnels[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[v_Sick_tunnel[n_tunnel_size], "H", ]  <- (1 - v_p_S1Dage) * p_S1H
a_P_tunnels[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[v_Sick_tunnel[n_tunnel_size], "S2", ] <- (1 - v_p_S1Dage) *
```

```
                                                     v_p_S1S2_tunnels[n_tunnel_size]
a_P_tunnels[v_Sick_tunnel[n_tunnel_size], "D", ]  <- v_p_S1Dage
### From S2
a_P_tunnels["S2", "S2", ] <- 1 - v_p_S2Dage
a_P_tunnels["S2", "D", ]  <- v_p_S2Dage
# From D
a_P_tunnels["D", "D", ] <- 1
```

Accounting for the effectiveness of treatment B is similar to the age-dependent approaches. We first transform `v_p_S1S2_tunnels` to a vector of rates, `v_r_S1S2_tunnels`, assuming exponentially distributed transitions across the tunnel states, and multiply times the hazard ratio of treatment B. Then, we transform back to probabilities to produce `v_p_S1S2_tunnels_trtB`, a vector of transition probabilities that account for the duration of S1 state-residence under treatment B.

```
## Transform risk of progression from Sick to Sicker to a rate
# vector of rates of becoming Sicker when Sick
v_r_S1S2_tunnels <- -log(1-v_p_S1S2_tunnels)/cycle_length
# apply hazard ratio to rate to obtain transition rate of becoming Sicker when
# Sick for treatment B
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(-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`. The only difference is that we update only the transition probabilities from S1 involving `v_p_S1S2_tunnels` to using `v_p_S1S2_tunnels_trtB` instead.

```
## Initialize transition probability array for treatment B
a_P_tunnels_trtB <- a_P_tunnels
## 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.

```
### Check if transition probability matrices are valid
## Check that transition probabilities are [0, 1]
check_transition_probability(a_P_tunnels)
check_transition_probability(a_P_tunnels_trtB)
## Check that all rows sum to 1
check_sum_of_transition_array(a_P_tunnels,      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 over the n_T cycles for the state-residence cSTM, we initialize two new cohort trace matrices for the SoC and treatment B, `m_M_tunnels` and `m_M_tunnels_trtB`, respectively. The dimensions of both matrices are $(n_T + 1) \times n_{S_{tunnels}}$.

```
# Initialize cohort for state-residence cSTM under SoC
m_M_tunnels <- 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[1, ] <- v_s_init_tunnels
## Initialize cohort trace under treatment B
m_M_tunnels_trtB <- m_M_tunnels
```

and then we use the matrix product, similar to the age-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[t + 1, ] <- m_M_tunnels[t, ] %*% a_P_tunnels[, , 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 under SoC comparable with the time-independent and age-dependent cSTM, we aggregate over the tunnel states of S1 in each cycle (Figure 6).

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

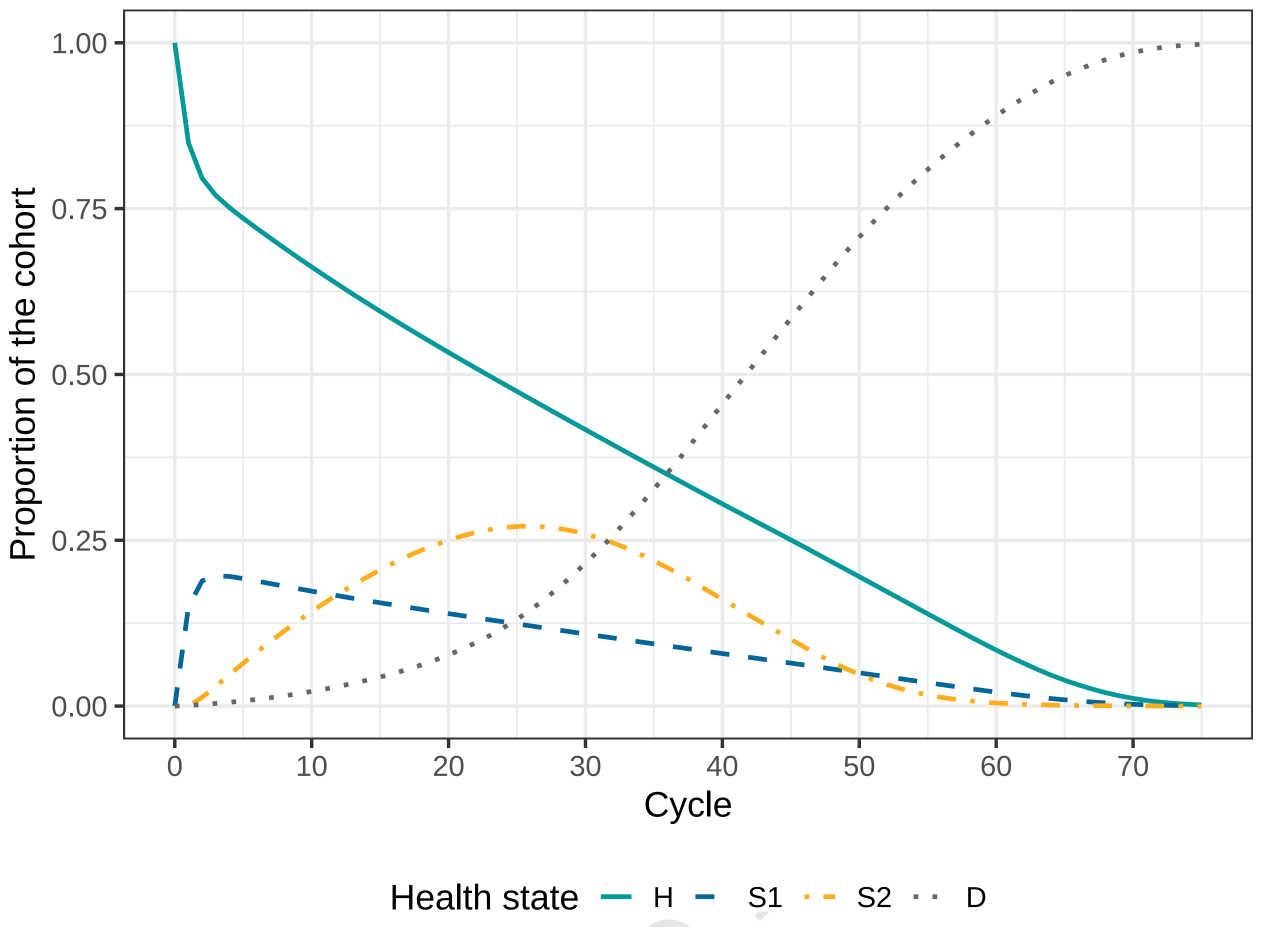

Figure 6: Cohort trace of the age-dependent cSTM accounting for state-residence dependency under strategies SoC and A.

4 Epidemiological and economic outputs

cSTMs can be used to generate different epidemiological and economic outputs. In a CEA, the main outcomes are typically the total expected QALYs and total costs accrued by the cohort over a predefined time horizon. However, epidemiological outcomes are often used to produce other measures of interest or model calibration and validation. Some common epidemiological outcomes include survival, prevalence, incidence, the average number of events, and lifetime risk of events.[13]

4.1 Epidemiological outcomes

In this tutorial, we provide the epidemiological definition of some of these outcomes and how they can be generated from a cSTM using the age-dependent Sick-Sicker cSTM under SoC. In the supplementary material, we provide the code to generate these outcomes from the state-residence-dependent cSTM.

4.1.1 Survival probability

The survival function, $S(t)$, captures the probability of the cohort remaining alive by cycle t. To estimate $S(t)$ from the simulated cohort of the age-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_ad`.

```r
v_S_ad <- rowSums(m_M_ad[, -which(v_names_states=="D")]) # vector with survival curve
```

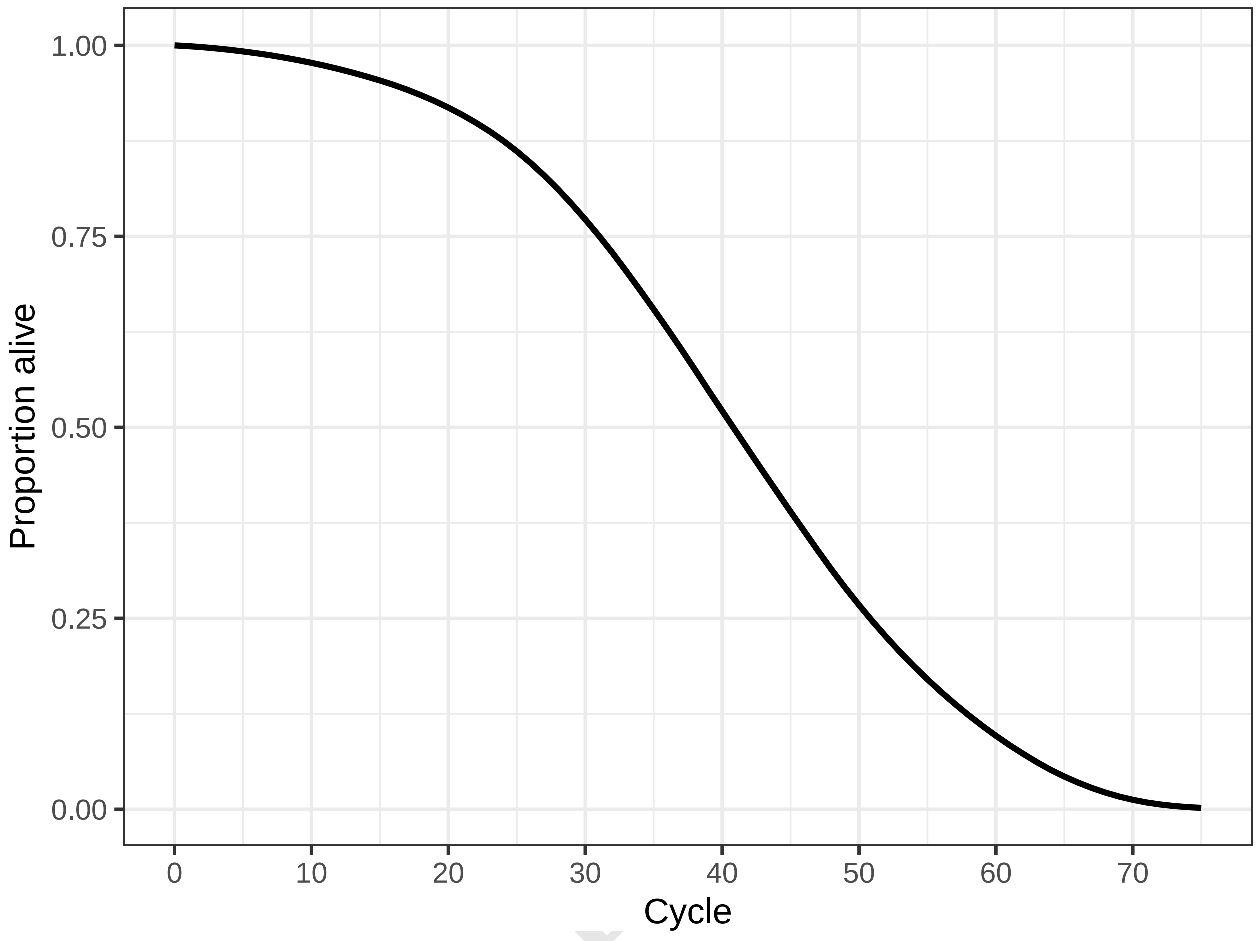

Figure 7: Survival curve of the age-dependent cSTM

4.1.2 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.[14] 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.[15] 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.

```r
v_prev_S1   <- m_M_ad[, "S1"] / v_S_ad          # vector with prevalence of Sick
v_prev_S2   <- m_M_ad[, "S2"] / v_S_ad          # vector with prevalence of Sicker
v_prev_S1S2 <- rowSums(m_M_ad[, c("S1", "S2")])/v_S_ad # prevalence of Sick and Sicker
```

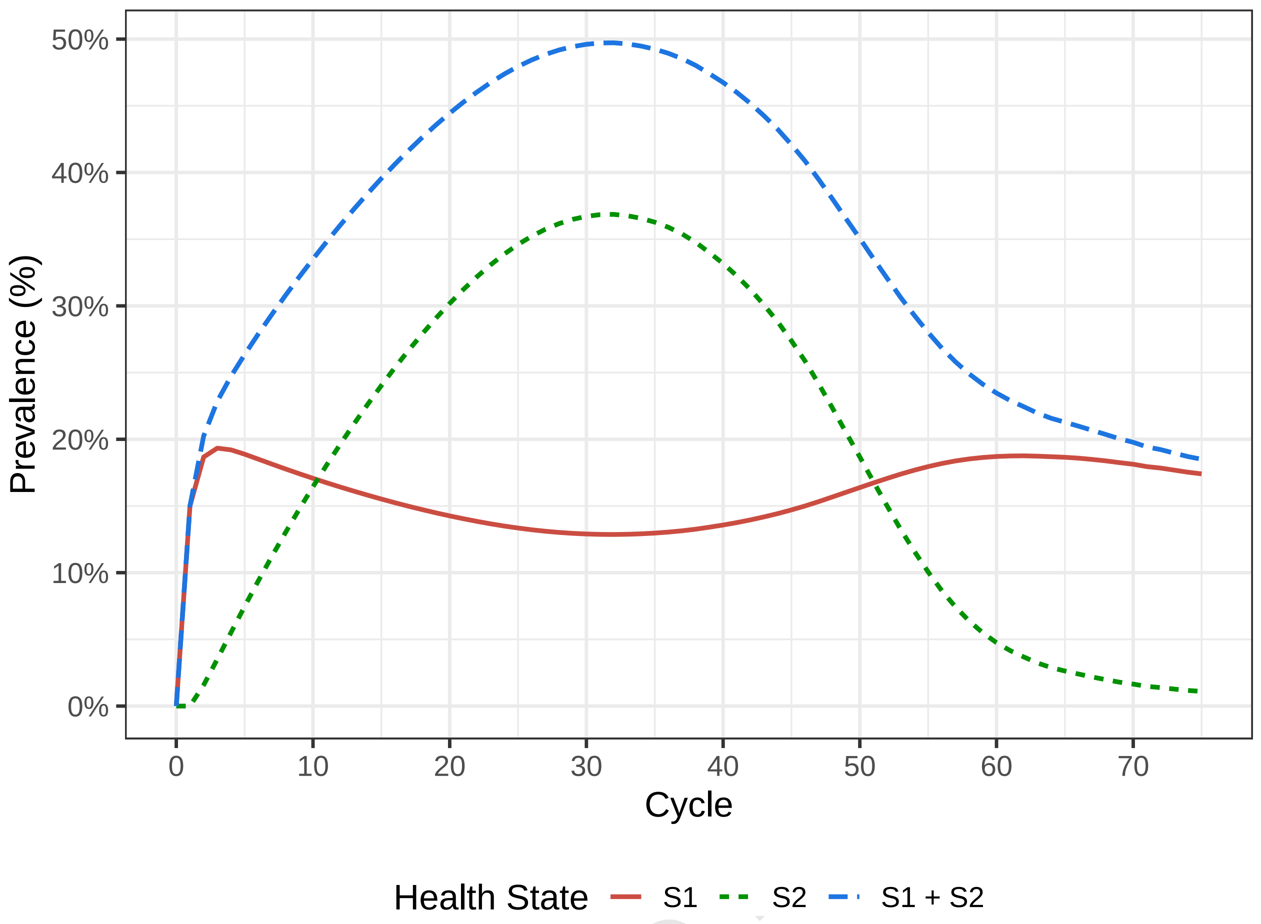

Figure 8: Prevalence of sick states in age-dependent cSTM

Another epidemiological outcome that could be of interest, is the proportion S1 among all individuals with the disease. To compute this outcome at each cycle t, we divide $m_{[S2,t]}$ by $\text{prev}(t)_{\{S1,S2\}}$ where $t > 0$, as presented in Figure 9. Note that t does not start at 0 because it takes one cycle for the cohort to get sick.

```
# Vector with proportion of Sicker among sick individuals
v_prop_S2 <- m_M_ad[-1, "S2"] / v_prev_S1S2[-1]
```

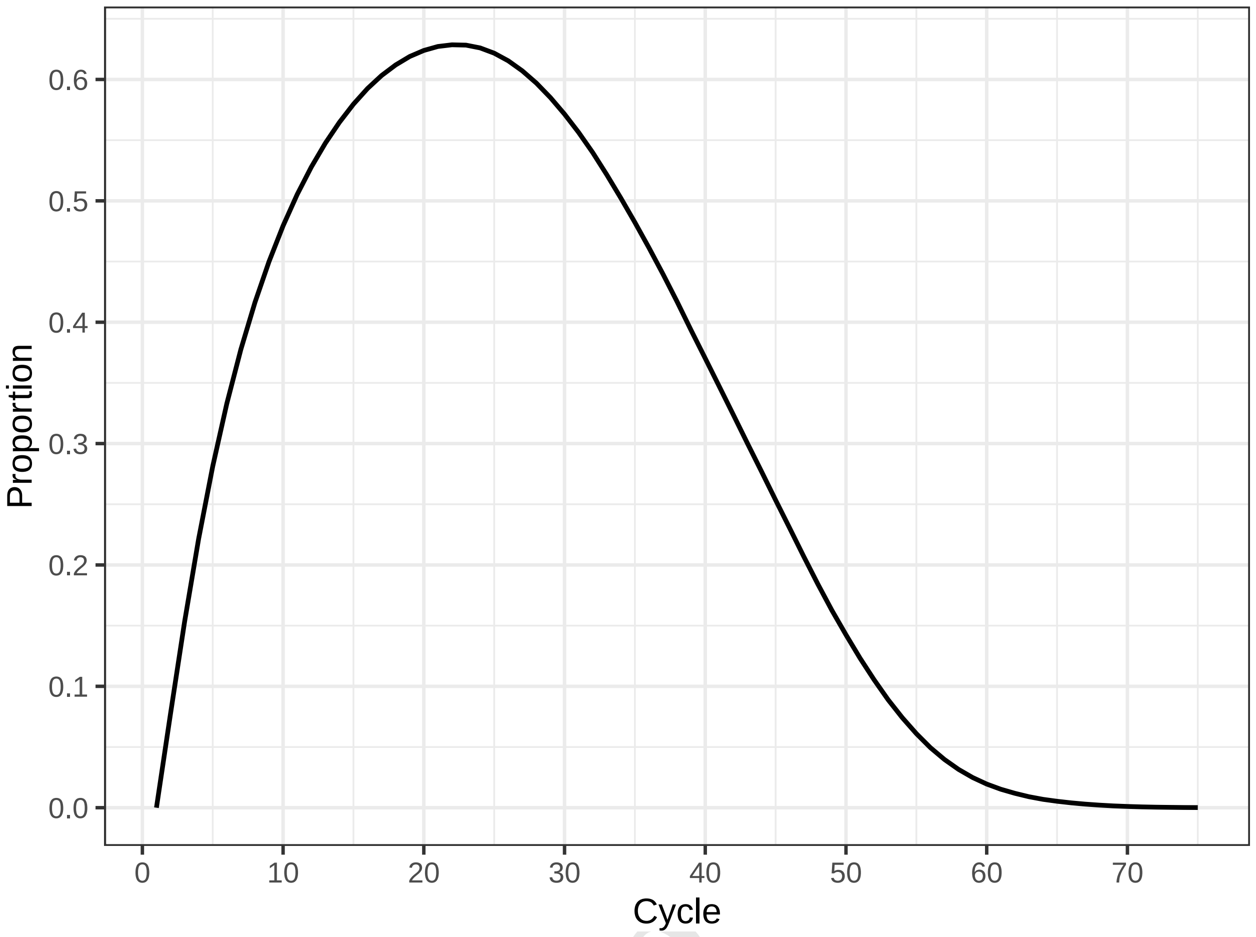

Figure 9: Proportion of Sicker (S2) individuals among all sick patients of age-dependent cSTM

4.1.3 Life expectancy

Life expectancy (LE) refers to the expected number of years 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 LE over a fixed time horizon at which most of the cohort has transitioned to the Dead state, n_T, and is defined as

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

In the age-dependent Sick-Sicker model, where we simulated a cohort over n_T= 75 cycles, life expectancy `le_ad` is 41.2 cycles, which is calculated as

```r
le_ad <- sum(v_S_ad) # 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.

4.2 Economic outcomes

In CEA, we can calculate economic outcomes by using either state or transition rewards. A “state reward” refers to a value assigned to individuals for being in a given state. In a cost-utility context, these could be either utilities or costs associated with remaining in a specific health state for one cycle. In the accompanying tutorial, we describe how to incorporate state rewards in CEA.[7] A “transition reward” refers to the increase or decrease in either costs or utilities of transitioning from one state to another, which may be associated with a one-time cost or utility impact.

4.2.1 State rewards

For the Sick-Sicker model, we 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 with being in each of the four health states under SoC which are shown in Table 2.

```r
# 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 for SoC 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.

```r
# 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 the state costs of S1 and S2. Similarly, when constructing the vector of state costs for strategy B, `v_c_strB`, we add the cost of treatment B, `c_trtB`, to the state costs of S1 and S2. 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)
# 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)
```

4.2.2 Transition rewards

In the Sick-Sicker model, we previously mentioned that 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. This one-time cost could include emergency services, hospitalization, or other healthcare utilization to address the ultimately fatal health complication. We also have a utility decrement and a cost increment on the transition from H to S1. Whereas the cost and utility of S1 reflect the cost and utility of being chronically sick, respectively, transition rewards capture the short-term impact of the acute events of becoming sick, such as hospitalization, stabilization, and so on.

Incorporating transition rewards requires keeping track of the proportion of the cohort that transitions between health states in each cycle while capturing what the states of origin and destination are. The cohort trace, M, does not capture this information. However, recovering this information is relatively straightforward and has been described in detail by Krijkamp et al. (2020).[18] 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 the states of origin and destination in 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 transition between states of origin and destination between cycles $t-1$ and t. Similarly, we define the transition rewards by the states of origin and destination.

To account for both state and transition rewards, 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.[22] Finally, we multiply this matrix by A_t, the t-th slice of A, apply discounting, within-cycle correction, and compute the overall reward for each strategy outcome. Below, we illustrate these concepts in R.

To compute $\mathbf{A}$ for the age-dependent Sick-Sicker model under SOC, we initialize a three-dimensional array `a_A` 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`.

We create a three-dimensional array for the cohort under treatment B, `a_A_trtB`, as a copy of the array under SoC.

```
# Initialize transition-dynamics array under SoC
a_A <- 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[, , 1]) <- v_s_init
# Initialize transition-dynamics array under treatment B
a_A_trtB <- a_A
```

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_ad[t, ])` and `diag(m_M_ad_trtB[t, ])`, by the t-th matrix of the array of transition matrices, `a_P[, , t]` and `a_P_trtB[, , 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[, , t + 1] <- diag(m_M_ad[t, ]) %*% a_P[, , t]
  # For treatment B
  a_A_trtB[, , t + 1] <- m_M_ad_trtB[t, ]  * a_P_trtB[, , t]
}
```

To create the arrays of rewards for costs and utilities for the age-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))
```

```r
# 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. For example, to account for the disutility of transitioning from H to S1 for 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.

```r
# 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 for strategy A.

```r
# 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 corresponding to SoC and strategies B and AB.

```r
## 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

## 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,

$$\mathbf{Y} = \mathbf{A} \odot \mathbf{R} \tag{1}$$

To obtain $\mathbf{Y}$ for QALYs and costs for all four strategies, we apply Equation (1) by the element-wise multiplication of the transition array `a_A` by the corresponding array of rewards.

```
# For SoC
a_Y_c_SoC <- a_A * a_R_c_SoC
a_Y_u_SoC <- a_A * a_R_u_SoC
# For Strategy A
a_Y_c_strA <- a_A * a_R_c_strA
a_Y_u_strA <- a_A * a_R_u_strA
# For Strategy B
a_Y_c_strB <- a_A_trtB * a_R_c_strB
a_Y_u_strB <- a_A_trtB * a_R_u_strB
# For Strategy AB
a_Y_c_strAB <- a_A_trtB * a_R_c_strAB
a_Y_u_strAB <- a_A_trtB * 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.

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

To obtain the expected costs and QALYs per cycle for each strategy, $\mathbf{y}$, we apply Equation (2) again across all the matrices of the third dimension of $\mathbf{Y}$ for all the outcomes.

```r
# 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)))
```

4.2.3 Within-cycle correction and discounting future rewards

Following the accompanying tutorial,[7] here we use Simpson's 1/3rd rule as a WCC,[20] and use exponential discounting for costs and QALYs. The within-cycle correction vector, **wcc**, is the same for both costs and QALYs; thus, only one vector, `v_wcc`, is required.

```r
## 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, **d**, for costs and QALYs for the Sick-Sicker model, `v_dwc` and `v_dwe`, respectively, are

```r
# 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 within-cycle correction, we incorporate **wcc** in equation (3) using an element-wise multiplication with **d**, indicated by the $\odot$ sign.

$$y = \mathbf{y}^{'} \left(\mathbf{d} \odot \mathbf{wcc}\right). \tag{3}$$

The total expected discounted costs and QALYs under all four strategies accounting for within-cycle correction, y, is obtained by applying Equation (3) 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 age-dependent Sick-Sicker model under the four strategies accounting for within-cycle correction are shown in Table 3.

Table 3: Total expected discounted QALYs and costs per average individual in the cohort of the age-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

5 Cost-effectiveness analysis and incremental cost-effectiveness ratios (ICERs)

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/)[21] to calculate the incremental costs and effectiveness and the incremental cost-effectiveness ratio (ICER) the non-dominated strategies and create the data frame `df_cea` with this information.

```
### 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)
```

The results of the CEA of the age-dependent Sick-Sicker model are presented in Table 4. 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.

Table 4: Cost-effectiveness analysis results for the age-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 10 shows the cost-effectiveness efficient frontier of all four strategies for the age-dependent Sick-Sicker model.

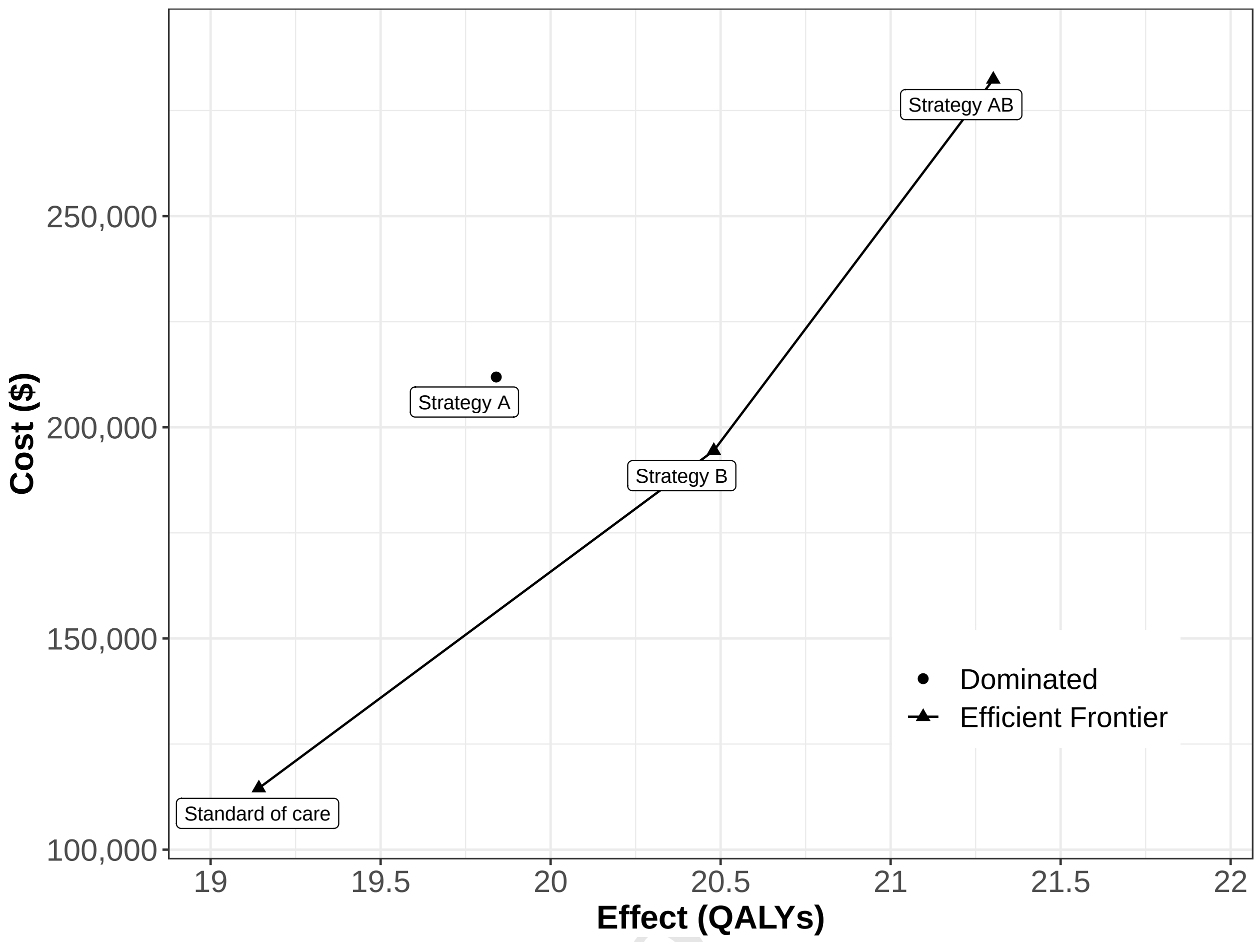

Figure 10: Cost-effectiveness efficient frontier of all four strategies for the age-dependent Sick-Sicker model.

6 Probabilistic sensitivity analysis

To quantify the effect of model parameter uncertainty on cost-effectiveness outcomes, we conducted a probabilistic sensitivity analysis (PSA).[22] In a PSA, we randomly draw parameter sets from distributions that reflect the current uncertainty in model parameter estimates. The distribution for all the parameters and their values are described in Table and in more detailed in the Supplementary Material. For each sampled set of parameter values, we compute model outcomes (e.g., total discounted cost and QALYs) for each strategy. In a previously published manuscript, we describe the implementation of these steps in R.[10] Briefly, to conduct the PSA, we create three R functions, which are provided in the supplementary material and in the accompanying GitHub repository:

1. A function called `generate_psa_params(n_sim, seed)` that generates a sample of size `n_sim` for the model parameters from their distributions defined in Table 2, the function also takes a seed number as input, which ensures reproducibility of the PSA results. By calling this function, we generate the sample of parameter sets for the PSA: `df_psa_input <- generate_psa_params(n_sim = n_sim)`
2. A function called `decision_model` that wraps the R code of the age-dependent cSTM described in section Simulation-time dependency and requires as input a list of model parameter values.

3. A function called `calculate_ce_out` that calculates outcomes of interest (e.g., total discounted costs and QALYs) based on the `decision_model` function output.

In Figure 11, we present the cost-effectiveness acceptability curves (CEACs), which shows the probability that each strategy is cost-effective, and the cost-effectiveness frontier (CEAF) which shows the strategy with the highest expected monetary benefit, over a range of willingness-to-pay (WTP) thresholds. The net monetary benefit for each strategy is defined as the product of total discounted QALYs and the WTP threshold minus the total discounted costs, calculated for each PSA parameter set 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 net monetary benefit. Strategy B has the highest probability of being cost-effective and the highest expected net monetary benefit for WTP thresholds between \$65,000 and \$100,000 per QALY. Strategy AB, has the highest expected net monetary benefit for WTP thresholds greater than or equal to \$100,000 per QALY and is the strategy with the highest probability of being cost-effective.

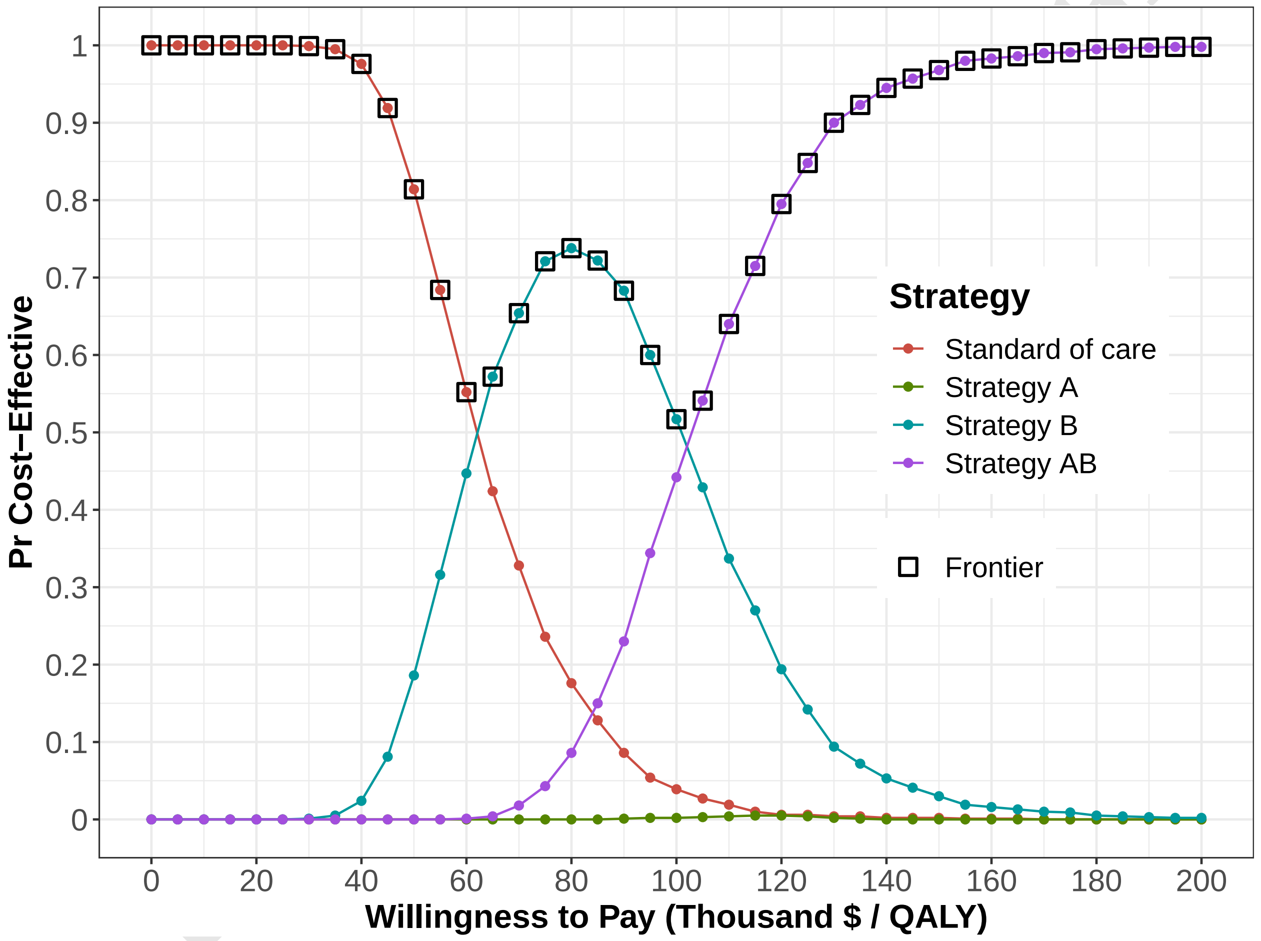

Figure 11: Cost-effectiveness acceptability curves (CEACs) and frontier (CEAF).

The code provided in the GitHub repository also produces expected loss curves (ELCs). These curves quantify the expected loss from each strategy over a range of WTP thresholds (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.[23]

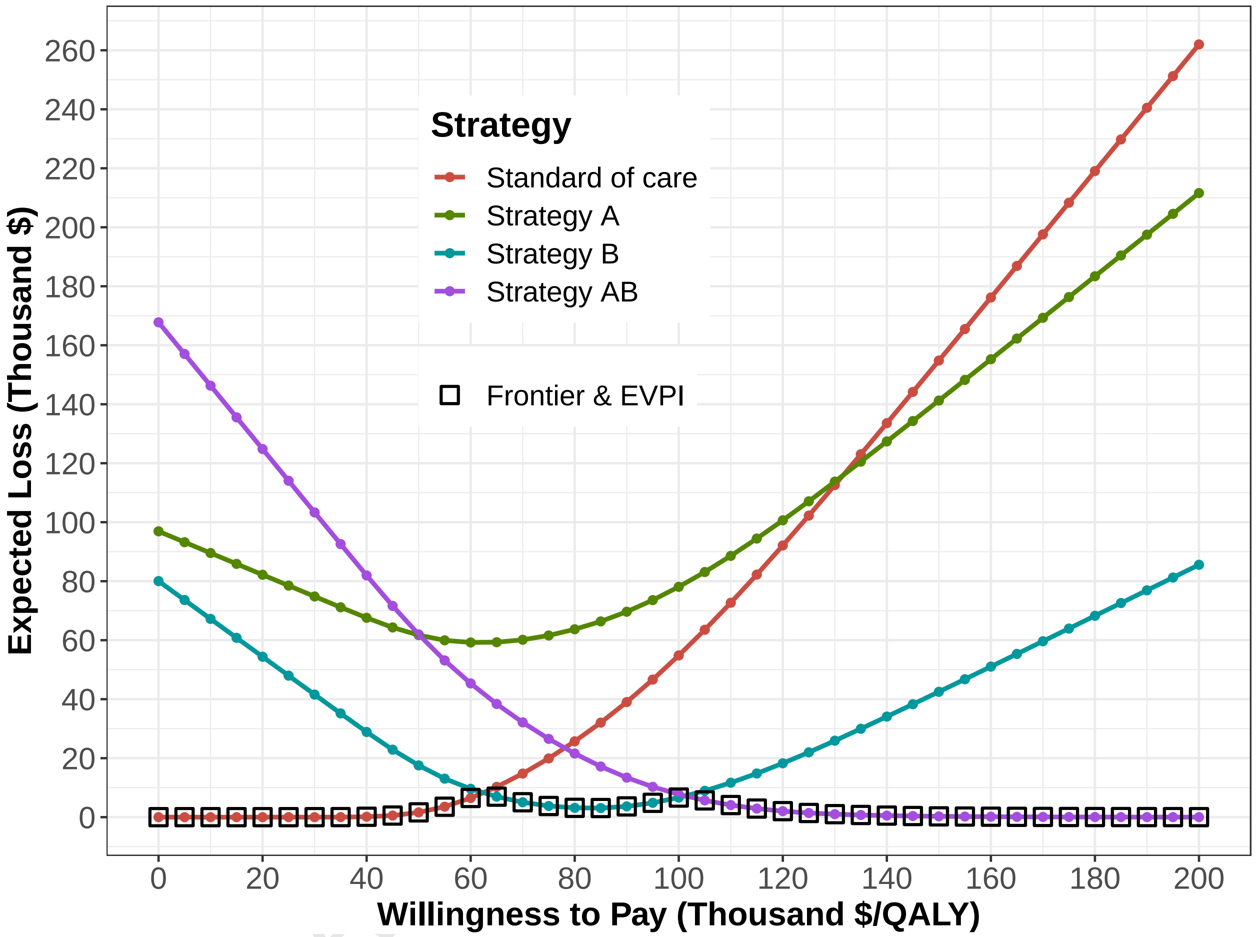

Figure 12: Expected loss curves (ELCs) and expected value of perfect information (EVPI).

7 Discussion

In this tutorial, we provided a conceptualization of various types of cSTMs with their mathematical description and a walk-through of their implementation for CEA in R using a previously published example. We used R as the programming language of choice to show the implementation of these models with accompanying code throughout the tutorial. We describe both time-independent and time-dependent models. We showed two different implementations of the time-dependent model, accounting for transition probabilities that are dependent on simulation time (e.g., age-dependence) and those that are dependent on state-residence.

There are several alternative approaches to incorporate time-dependency in cSTMs in R and various programming languages. For example, another approach to incorporate age-dependency involves updating the time-varying elements of the transition probability matrix P_t at each time point t. That would alleviate the need for the construction of the array `a_P`. This can reduce computer memory requirements but at the expense of increasing the number of operations in the update of P_t at every cycle. Another approach to account for history dependence is to use a 3-dimensional transition probability matrix with dimensions for the current state, future state, and time in the current state.[24] However, to incorporate age-dependence, this multidimensional matrix will have to be expanded by another dimension. The states will have to be replicated for as many age groups are considered in the model. In this tutorial, we decided to use the third dimension as the cohort's age and incorporate state-residence by expanding the corresponding health states on the second dimension of the 3-dimensional array to account for time spent in the current state. This approach's benefit is that we can still the transition dynamics array to capture all transitions between all states for all cycles.

The parameterization of our example model assumes all parameters are known or at least, the characterization of their uncertainty is known (i.e., we know their distributions). However, to construct a real-world cSTM it is important that modelers conduct a thorough synthesis of current evidence to determine the appropriate structure of these models and inform all parameters. For example, determining whether transitions between non-death health states are estimated conditional on being alive or mortality risks are also considered competing risks.[22] Similarly, our PSA analysis is a simplification of reality where all model parameters are assumed to be independent from each other. However, parameters could be correlated with each other or have a rank ordering, and appropriate statistical methods that simulate these correlations or rank ordering might be needed.[25] We encourage modelers to use appropriate statistical methods to accurately synthesize and quantify model parameters uncertainty. In addition, all model parameters should be correctly specified for the cycle length of the model. For example, some probability revision is required to correctly adjust an annual mortality rate to a weekly probability.[26]

In general, cSTMs are recommended when the number of states is "not too large".[13] This recommendation arises because as the number of states increases, it becomes more difficult to keep track of their construction but not because of the added computational expense. It is possible to build fairly complex cSTMs in R as long as the size of the transition probability matrix and outputs of interest can be stored in the RAM memory of the computer running the analysis. For example, a typical PC with 8GB of RAM can handle a transition probability array of about 1000 states and 600 slices. However, these matrices can grow quickly, and if the required number of state descriptions gets too large and difficult to manage its coding, it becomes 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.[13] In iSTM, the risks and rewards of simulated individuals do not need to depend only on a specific health state and can depend on their individual characteristics and attributes. Besides, 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. It is recommended to think about the required model structure before implementing the model in R or any other tool because turning a cSTM into an iSTM requires a different code structure. However, most input parameters and some model structures might remain the same. Still, 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 write these additional functions for the Sick-Sicker example model.[11]

With increasing model complexity and the use of functions for PSA, it is important to ensure all code and functions work as expected and all elements of cSTM are valid. This can be achieved by functions that help with model debugging and validation and through unit testing. In the accompanying GitHub repository, we provided functions to check that transition probability matrices and their elements are valid. However, unit testing is beyond the scope of this tutorial but we refer the reader to a previously published manuscript where we describe unit testing in more detail and provide accompanying code.[10]

We focused on discrete-time matrix-form cSTMs but these can also be implemented via a set of difference equations and in continuous time using differential equatiopns in R.[27,28] We refer readers interested in learning more on continuous-time cSTMs to previously published manuscripts[29–32] and a tutorial using R.[33] There are other approaches to construct Markov models in R, most of them through purpose-specific R packages. For example, the `heemod`[34] package is designed to build Markov models that account for dependency using a pre-defined structure. Although `heemod` is a well-structured package, it requires users to set up the structure of the Markov model, specify the parameters and run analyses in a pre-specified approach. This, however, limits the understanding on how cSTMs and Markov models work and are constructed. In this tutorial, we show how cSTMs are constructed, parameterized and run by using only base R so readers get a deep understanding of this type of decision models. Finally, the variable names used in this paper reflect our own style. While we provide guidance on standardized variable names, the adoption of these conventions are ultimately a personal preference.

In summary, this tutorial provides a conceptualization of cSTMs and a step-by-step guide to implement them in R. We aim to add to the current body of literature and material on building this type of decision models so health decision scientists and health economists are able to develop cSTMs in a more flexible, efficient, open-source manner, and encouraging increased transparency and reproducibility.

8 Acknowledgements

Dr Alarid-Escudero was supported by a grant from the National Cancer Institute (U01-CA-199335) as part of the Cancer Intervention and Surveillance Modeling Network (CISNET), and 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 receives Royalties from Cambridge University Press for a textbook on Medical Decision Making, reimbursement of expenses from the European Society of Radiology (ESR) for work on the ESR guidelines for imaging referrals, reimbursement of expenses from the European Institute for Biomedical Imaging Research (EIBIR) for membership of the Scientific Advisory Board, and 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 that took our classes to try our our 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 to Cohort State-Transition Models in R. 2021.

8. 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).

9. 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).

10. 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).

11. 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).

12. 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_04.pdf (2017).

13. 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).

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

15. 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.

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+&+health/book/978-0-387-95399-1 (2003).

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

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

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

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

22. 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.

23. 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.

24. 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).

25. Goldhaber-Fiebert JD, Jalal HJ. Some Health States Are Better Than Others: Using Health State Rank Order to Improve Probabilistic Analyses. *Medical Decision Making*; 36: 927–940, http://mdm.sagepub.com/cgi/doi/10.1177/0272989X15605091 (2015).

26. Hunink MGGM, Weinstein MC, Wittenberg E, et al. *Decision Making in Health and Medicine.* 2nd ed. Cambridge: Cambridge University Press, http://ebooks.cambridge.org/ref/id/CBO9781139506779 (2014).

27. Grimmett G, Welsh D. Markov Chains. In: *Probability: An introduction.* Oxford University Press, pp. 203–, www.statslab.cam.ac.uk/ grg/teaching/chapter12.pdf (2014).

28. Axler S, Gehring FW, Ribet KA. Difference Equations. New York, NY: Springer, http://link.springer.com/10.1007/0-387-27645-9 (2005).

29. Cao Q, Buskens E, Feenstra T, et al. Continuous-Time Semi-Markov Models in Health Economic Decision Making: An Illustrative Example in Heart Failure Disease Management. *Medical Decision Making*; 36: 59–71, http://mdm.sagepub.com/cgi/doi/10.1177/0272989X15593080 (2016).

30. Rosmalen J van, Toy M, O'Mahony JF. A Mathematical Approach for Evaluating Markov Models in Continuous Time without Discrete-Event Simulation. *Medical Decision Making*; 33: 767–779, http://mdm.sagepub.com/cgi/doi/10.1177/0272989X13487947 (2013).

31. Begun A, Icks A, Waldeyer R, et al. Identification of a multistate continuous-time nonhomogeneous Markov chain model for patients with decreased renal function. *Medical Decision Making*; 33: 298–306, http://www.ncbi.nlm.nih.gov/pubmed/23275452 (2013).

32. Soares MO, Canto E Castro L. Continuous time simulation and discretized models for cost-effectiveness analysis. *PharmacoEconomics*; 30: 1101–1117, http://www.ncbi.nlm.nih.gov/pubmed/23116289 (2012).

33. Frederix GWJ, Hasselt JGC van, Severens JL, et al. Development of a framework for cohort simulation in cost-effectiveness analyses using a multistep ordinary differential equation solver algorithm in R. *Medical Decision Making*; 33: 780–92, http://www.ncbi.nlm.nih.gov/pubmed/23515213 (2013).

34. Filipović-Pierucci A, Zarca K, Durand-Zaleski I. Markov Models for Health Economic Evaluation: The R Package heemod. *arXiv:170203252v1*; April: 30, http://arxiv.org/abs/1702.03252 (2017).