hesim: Health Economic Simulation Modeling and Decision Analysis

Devin Incerti
Genentech

Jeroen Jansen
University of California, San Francisco

Abstract

Health economic models simulate the costs and effects of health technologies for use in health technology assessment (HTA) to inform efficient use of scarce resources. Models have historically been developed using spreadsheet software (e.g., Microsoft Excel) and while use of R is growing, general purpose modeling software is still limited. **hesim** helps fill this gap by facilitating parameterization, simulation, and analysis of economic models in an integrated manner. Supported model types include cohort discrete time state transition models (cDTSTMs), individual continuous time state transition models (iCTSTMs), and partitioned survival models (PSMs), encompassing Markov (time-homogeneous and time-inhomogeneous) and semi-Markov processes. A modular design based on **R6** and S3 classes allows users to combine separate submodels for disease progression, costs, and utility in a flexible way. Probabilistic sensitivity analysis (PSA) is used to propagate uncertainty in model parameters to model outputs. Simulation code is written in C++ so complex simulations such as those combining PSA and individual simulation can be run much more quickly than previously possible. Decision analysis within a cost-effectiveness framework is performed using simulated costs and quality-adjusted life years (QALYs) from a PSA.

Keywords: health economic evaluation, cost-effectiveness analysis, simulation, multi-state models, R.

1. Introduction

Health technology assessment (HTA) is a systematic approach for comparing competing health technologies to inform the efficient use of health care resources. Publicly funded health systems such as those in Australia, Canada, and the United Kingdom, among others, use HTA to help maximize health gains for the population given a fixed budget. HTA is also commonly used by private payers to guide coverage decisions and the adoption of new technologies (Trosman, Van Bebber, and Phillips 2011). Such assessments usually rely heavily on cost-effectiveness analysis (CEA) so that decisions can be made on the basis of a formal evaluation of costs and effects (Dakin, Devlin, Feng, Rice, O'Neill, and Parkin 2015). CEA is made feasible by development of health economic models that simulate costs and effects over relevant time horizons using the totality of available evidence.

The most commonly used economic models are Markov state transition models (STMs) that simulate transitions between mutually exclusive health states. When the Markov assumption holds so that transition probabilities are either constant over time or depend only on model time, a cohort-level model can be used (Briggs and Sculpher 1998). If the Markov assumption

is relaxed—e.g., with a semi-Markov model so that transition probabilities depend on time in an intermediate health state—then individual-level models are required in most cases (Brennan, Chick, and Davies 2006; Fiocco, Putter, and van Houwelingen 2008). Individual-level models afford considerably more flexibility and allow patient history to be tracked over time.

Decisions informed by economic models and CEA are subject to uncertainty. One source of decision uncertainty stems from uncertainty in the underlying model parameters. Parameter uncertainty is typically quantified using probabilistic sensitivity analysis (PSA), which involves randomly sampling the model parameters from suitable probability distribution and simulating the model for each sampled parameter set (Claxton, Sculpher, McCabe, Briggs, Akehurst, Buxton, Brazier, and O'Hagan 2005). When combined with individual-level simulation, PSA can take an appreciable amount of time to run (O'Hagan, Stevenson, and Madan 2007).

Estimation of model parameters should ideally be performed using statistical models that are aligned with the structure of the economic model. For instance, when patient-level data is available, a multi-state model can be used to parameterize all possible transitions in a STM while accounting for censoring and competing risks (Williams, Lewsey, Briggs, and Mackay 2017). Similarly, in oncology, partitioned survival models (PSMs) can be parameterized from estimates of progression-free survival (PFS) and overall survival (OS). In other cases, parameters might be combined from disparate sources, such as within a single Bayesian model (Baio 2012). When the clinical evidence base is not limited to a single study, a formal evidence synthesis, such as a network-meta analysis (NMA), might even be performed (Dias, Ades, Welton, Jansen, and Sutton 2018).

Despite their computational demands and foundations in statistics, health economic models have historically been developed with specialized commercial software (e.g., TreeAge) or more commonly with a spreadsheet (almost always Microsoft Excel). The limitations of such software relative to programming languages like R have been increasingly emphasized in the literature (Baio and Heath 2017; Incerti, Thom, Baio, and Jansen 2019; Jalal, Pechlivanoglou, Krijkamp, Alarid-Escudero, Enns, and Hunink 2017). It is therefore no surprise that a number of related R packages have recently been developed, such as **BCEA** (Baio, Berardi, and Heath 2017), **SAVI** (Strong, Oakley, and Brennan 2014), **survHE** (Baio 2020), and **heemod** (Filipović-Pierucci, Zarca, and Durand-Zaleski 2017). Still, of the available packages, only **heemod** provides a general purpose framework for developing simulation models and it is limited Markov cohort models.

hesim is an R package that advances the functionality and performance of the existing software. Multiple model types are supported including cohort discrete time state transition models (cDTSTMs), N-state PSMs, and individual-level continuous time state transition models (iCTSTMs), encompassing both Markov (time-homogeneous and time-inhomogeneous) and semi-Markov processes. To maximize flexibility and facilitate integration of the statistical methods and economic model, parameters can be set by either fitting models in R or by inputting values obtained from external sources. So that individual-level simulation and PSA can be run quickly, **Rcpp** and **data.table** are heavily utilized. After simulating costs and quality-adjusted life-years (QALYs) from a PSA, decision analysis can be performed within a cost-effectiveness framework.

The remainder of this paper is organized as follows. Section 2 provides a mathematical description the economic models supported by **hesim**. An overview of the coding framework is

provided in Section 3. An illustrative example using a three state model of disease progression in oncology is provided in Section 4. Analyses are conducted for each of the three supported model types, illustrating approaches suitable for both patient-level and aggregate-level data. The cost-effectiveness framework is described in Section 5 along with worked examples based on the output from the prior section. Section 6 discusses additional features not covered in this paper, makes comparisons to other software, and explores possible extensions. Finally, Section 7 concludes.

Analyses were performed using **hesim** version `0.5.0`. All code including a replication R script for this paper is available at https://github.com/hesim-dev/hesim-manuscript.

2. Model taxonomy

STMs simulate transitions between mutually exclusive health states. A common assumption is that a STM is a Markov model, meaning that transitions to the next health state can only depend on the present health state. In a time homogeneous Markov model transition probabilities are constant over time, whereas in a time inhomogeneous model they can depend on time since the start of the model. A semi-Markov model relaxes the Markov assumption and allows transitions to depend on time since entering an intermediate state.

Markov and semi-Markov models can be formulated in either continuous or discrete time, at either the cohort- or individual-level. In health economics, Markov cohort models tend to be formulated in discrete time (i.e., as cDTSTMs), although state probabilities can be computed with continuous time models using the Aalen-Johansen estimator (Aalen and Johansen 1978) or the Kolmogorov forward equation (Cox and Miller 1977). While tunnel states can be used to approximate a semi-Markov model using a cohort approach, they can only be simulated in a general fashion using individual-level models. Discrete time individual simulation is possible, but we use continuous time models (i.e., iCTSTMs) because they do not require specification of model cycles and can be run considerably faster.

PSMs are specialized models that can be parameterized using survival curves and are especially useful in oncology where PFS and OS are commonly reported. They are "area under the curve" models, although they can also be formulated as STMs by using the survival curves to construct transition probabilities.

2.1. Cohort discrete time state transition models

cDTSTMs simulate the probability that a cohort of patients is in each of H health states over time. Time is partitioned into intervals $\{[0, t_1), [t_1, t_2), \dots, [t_C, t_{C+1})\}$ with each interval $0, \dots, C$ reffered to as a model cycle. We denote t_c as the time at the start of model cycle c. An $H \times 1$ state vector that stores the probability of being in each health state at cycle c is written as $x_c = (x_{1c}, x_{2c}, \dots, x_{Hc})$ where $\sum_h x_{hc} = 1$. A transition probability matrix is an $H \times H$ matrix denoted by P_c where the (r, s)th element represents a transition from state r to state s between model cycles c and $c + 1$. The state vector at cycle $c + 1$ for each of C model cycles is given by,

$$x_{c+1}^T = x_c^T P_c, \quad c = 0, \dots, C. \tag{1}$$

The transition probability matrices can also be characterized in terms of transition intensity

matrices, Q_c, of the same dimensions as P_c and representing an underlying continuous process with a patient in state $X(t)$ at exact time t. The (r, s)th element is the hazard representing the instantaneous risk of moving from state r to state s,

$$h_{rs}(t) = \lim_{\Delta t \to 0} \frac{\mathsf{P}(X(t + \Delta t) = s | X(t) = r)}{\Delta t}. \tag{2}$$

Each row of Q_c sums to 0 whereas each row of P_c sums to 1. If transition intensities are constant over the interval (t_c, t_{c+1}) for a model cycle with length $u = t_{c+1} - t_c$, then the transition probability matrix can be solved with the matrix exponential as,

$$P_c = \text{Exp}(uQ_c). \tag{3}$$

Costs and QALYs are computed by assigning (potentially time-varying) values to each health state. Utility, a measure of preference for a health state that normally ranges from 0 (dead) to 1 (perfect health), is used when computing QALYs (Torrance 1986). Assuming model time is in years, state values for costs are estimated by annualizing costs. In a Markov model, state values, like transition probabilities, can depend on time since the start of the model but not on time since entering an intermediate health state.

Expected values are computed by integrating the "weighted" probability of being in each state, where weights are a function of the discount factor and state values. That is, for a time horizon T, discounted costs and QALYs in health state h are computed as,

$$\int_0^T z_h(t) e^{-rt} p_h(t) dt, \tag{4}$$

where r is the discount rate and at time t, $z_h(t)$ is the predicted cost or utility value and $p_h(t) = \mathsf{P}(X(t) = h)$ is the probability of being in health state h.

In discrete time, the integral is approximated with

$$\sum_{c=0}^{C} f(t_c^*) \Delta t_c, \tag{5}$$

where $\Delta t_c = t_{c+1} - t_c$, $t_c^* \in [t_c, t_{c+1}]$, and $f(t) = z_h(t) e^{-rt} p_h(t)$. Three methods can be used to estimate $f(t_c^*)$. First, a left Riemann sum uses values at the start of each time interval $f(t_c^*) = f(t_c)$. Second, a right Riemann sum uses values at the end of each time interval $f(t_c^*) = f(t_{c+1})$. Finally, the trapezoid rule averages values at the start and end of each interval $f(t_c^*) = \frac{1}{2} \Delta t_c [f(t_c) + f(t_{c+1})]$.

2.2. Individual continuous time state transition models

iCTSTMs simulate individual trajectories between health states using random number generation. Trajectories are simulated for multiple patients and costs and QALYs are computed by averaging across the simulated patients. A reasonably large number of patients must be simulated to ensure that expected values are stable (O'Hagan *et al.* 2007).

In continuous time, a patient is in state $X(t)$ at time t. State transitions are modeled using a multi-state modeling framework (Putter, Fiocco, and Geskus 2007) where the probability of a transition from state r to state s is governed by the hazard function from Equation 2.

Simulated disease progression is characterized by J distinct jumps between health states $D = \{(t_0, X(t_0)), (t_1, X(t_1)), \dots (t_J, X(t_J))\}$ with a patient remaining in a health state from time t_j until transitioning to the next state at t_{j+1}. Jumps between health states are simulated using parametric and flexible parametric survival models as implemented in the **flexsurv** package (Jackson 2016). Specifically, if a patient enters state r at time t_j, then a probability density function for the time-to-event t^* for the $r \to s$ transition is,

$$f_{rs}(t^*|\theta(z), t_j), \quad t^* \geq 0, \tag{6}$$

where parameters $\theta = (\theta_1, \theta_2, \dots, \theta_p)$ may depend on covariates z_p through the link function $g(\theta_p) = z_p^T \gamma$ and γ is a vector of regression coefficients. In a time inhomogeneous Markov model, time $t^* = t_{j+1}$ is conditional on not experiencing event s until time t_j (i.e., it is left-truncated at time t_j). In a semi-Markov model, time-to-event is expressed in terms of $t^* = t_{j+1} - t_j$ and a patient enters state s at time $t_j + t^*$.

A survival distribution is specified for each permitted transition in the multi-state model. A trajectory through the model can then be simulated as described in Algorithm 1. While the algorithm repeats until a patient dies, it can also be stopped at a specified time t or when a patient reaches a maximum age. In the latter scenario, death is assume to occur at the maximum age.

Algorithm 1 Simulation of individual continuous time state transition model

1. Let r be the state entered at time t_j. The number of permitted transitions from state r is given by n_r. If $j = 0$, then $t_j = 0$.
2. Simulate times $\mathcal{T} = \{t_{1,j+1}, t_{2,j+1}, \dots t_{n_r,j+1}\}$ to each of the n_r permitted transitions.
3. Set the time of the transition t_{j+1} equal to the minimum simulated time in $\mathcal{T}$ and the next state s to the state with the minimum simulated time.
4. Set $r = s$ and $t_j = t_{j+1}$. If the patient is still alive, repeat the previous steps until death.

Costs and QALYs are computed using the continuous time present value given a flow of state values, which change as patients transition between health states or as costs vary as a function of time. The state values can be partitioned into M time intervals indexed by $m = 1, \dots, M$ where interval m contains times t such that $t_m \leq t \leq t_{m+1}$ and values for state h are equal to z_{hm} during interval m. z_{hm} will equal zero during time intervals in which a patient is not in state h. Discounted costs and QALYs for health state h are then given by,

$$\sum_{m=1}^{M} \int_{t_m}^{t_m+1} z_{hm} e^{-rt} dt = \sum_{m=1}^{M} z_{hm} \left(\frac{e^{-rt_m} - e^{-rt_{m+1}}}{r} \right), \tag{7}$$

where $r > 0$ is again the discount rate. If $r = 0$, then the present value simplifies to $\sum_{m=1}^{M} z_{hm}(t_{m+1} - t_m)$.

Note that while state values in cohort models can depend on time since the start of the model, state values in individual-level models can depend on either time since the start of the model

or time since entering the most recent health state. Individual-level models consequently not only afford more flexibility than cohort models when simulating disease progression, but when simulating costs and/or QALYs as well.

2.3. Partitioned survival models

PSMs are conceptually similar to STMs in that they are characterized by mutually exclusive health states. They differ, however, in that state probabilities are not computed via matrix multiplication or individual simulation, but from a set of non-mutually exclusive survival curves (Glasziou, Simes, and Gelber 1990; Woods, Sideris, Palmer, Latimer, and Soares 2018). Each survival curve represents time to transitioning to that state or to a more severe health state.

In an N-state model, $N-1$ non-mutually exclusive survival curves are required. The cumulative survival function, $S_n(t)$, represents the probability that a patient survives to health state n or to a lower indexed state beyond time t. The probability that a patient is in health state 1 is $S_1(t)$. State membership in health states $2, \dots, N-1$ is computed as $S_n(t) - S_{n-1}(t)$. Finally, the probability of being in the final health state n (i.e., the death state) is $1-S_{N-1}(t)$, or one minus overall survival function.

Survival functions are estimated by fitting parametric or flexible parametric survival models as described in Section 2.2. The nth fitted survival model with density $f_n(t)$ has cumulative density function $F_n(t)$, survivor function $1 - F_n(T)$, cumulative hazard $H_n(t) = -\log S_n(t)$, and hazard $h_n(t) = f_n(t)/S_n(t)$. State probabilities for each health state, x_{hc}, can be computed for an arbitrarily fine grid of time periods c. Costs and QALYs are then computed in the same manner as in the cohort model described in Section 2.1.

3. Framework

Economic models consist of a disease model, a utility model, and a set of cost models for each cost category. Model development proceeds in 4 steps. The first step is to setup the model by defining the model structure, target population, and treatment strategies of interest. The relevant information is then stored in a `hesim_data` object.

Development and analysis of the model is performed in the next three steps as shown in Figure 1. First, the disease progression, costs, and utility submodels are parameterized using "estimation" datasets. Next, the submodels are combined to construct an economic model. The economic model is then used to simulate, disease progression, QALYs, and costs as a function of "input data". The input data always contains variables describing the target population and treatment strategies of interest, but can also contain variables related to time to facilitate use of time-varying covariates or parameters. Finally, the simulated QALYs and costs are used to perform decision analysis within a cost-effectiveness framework. While other approaches such as multi-criteria decision analysis (MCDA) could, in principle, be used as well, only CEA is currently supported. All models are simulated using PSA so that decision uncertainty can be represented.

The three economic models described in Section 2 are implemented as the **R6** classes `CohortDtstm`, `IndivCtstm`, and `Psm`. Each class contains methods for simulating disease progression, QALYs, and costs. These methods are made possible by separate **R6** classes for the disease, utility, and cost submodels. The remainder of this section describes the framework for parameterization

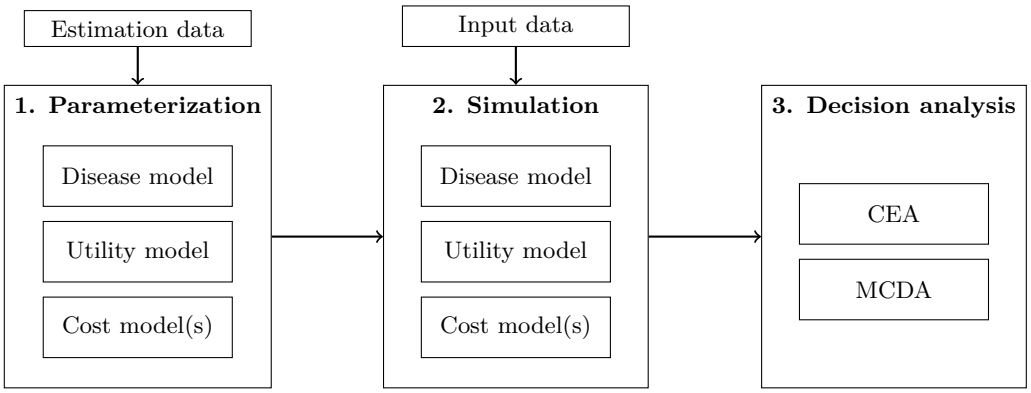


Figure 1: Economic modeling process

and simulation. The cost-effectiveness framework is described in Section 5.

3.1. Parameterization

Each submodel contains fields for the model parameters and the input data. Parameters can either be estimated from a statistical model fit using R or input directly by the user. There are two types of parameter objects, standard parameter objects prefixed by `"params"` and "transformed" parameter objects prefixed by `"tparams"`. The former contain the underlying parameters of a statistical model while the latter contain parameters more immediate to prediction that have been transformed as function of the input data. The regression coefficients of a logistic regression are an example of a parameter objects while the predicted probabilities are examples of a transformed parameter object.

Transformed parameter objects are often useful when parameterizing a model using external sources. Two common examples are the predicted probabilities comprising transition probability matrices and predicted means for assigning values to health states. The advantage of a transformed parameter object is that an explicit statistical model does not need to be specified which gives the user more flexibility.

Disease progression

The statistical model used to parameterize the disease model depends on the type of economic model. For example, multinomial logistic regressions can be used to parameterize a cDTSTM, a set of N-1 independent survival models are used to parameterize an N-state partitioned survival model, and multi-state models can be used to parameterize an iCTSTM (Table 1).

Table 1: Parameterization of disease models

Economic model	Statistical model	Parameter object	Model object
`CohortDtstm`	Custom	`tparams_transprobs`	`msm::msm`
	Multinomial logistic regressions	`params_mlogit`	`multinom_list`
`IndivCtstm`	Multi-state model (joint likelihood)	`params_surv`	`flexsurv::flexsurvreg`
	Multi-state model (transition-specific)	`params_surv_list`	`flexsurvreg_list`
`Psm`	Independent survival models	`param_surv_list`	`flexsurvreg_list`

The parameters of a survival model are stored in a `params_surv` object and a `params_surv_list` can be used to store the parameters of multiple survival models. The latter is useful for stor-

ing the parameters of a multi-state model or the independent survival models required for a PSM. The parameters of a multinomial logistic regression are stored in a `params_mlogit` object and can be created by fitting a model for each row in a transition probability matrix with `nnet::multinom()`.

`tparams_transprobs` objects store transition probability matrices that have been predicted for each PSA sample, treatment strategy, patient from the target population, and optionally time interval (to allow for time inhomogeneous Markov models). Transition probabilities can be predicted from a fitted multi-state model using the **msm** package or constructed "by hand" in a custom manner.

Costs and utility

State values (i.e., costs and utilities) do not depend on the choice of disease model. They can currently either be modeled using a linear model or using predicted means (Table 2).

Table 2: Parameterization of state value models

Statistical model	Parameter object	Model object
Predicted means	`tparams_mean`	`stateval_tbl`
Linear model	`params_lm`	`stats::lm`

The most straightforward way to construct state values is with `stateval_tbl`, which is a special object used to assign values (i.e. predicted means) to health states that can vary across PSA samples, treatment strategies, patients, and/or time intervals. State values can be specified either as moments (e.g., mean and standard error) or parameters (e.g., shape and scale of gamma distribution) of a probability distribution, or by pre-simulating values from a suitable probability distribution (e.g., from a Bayesian model).

3.2. Simulation

The utility and cost models are always **R6** objects of class `StateVals`, whereas the disease models vary by economic model (Table 3). The disease model is used to simulate survival curves in a PSM and health state transitions in a cDTSTM and iCTSTM.

Table 3: Submodels comprising an economic model

Economic model	Disease model	Utility model	Cost model(s)
`CohortDtstm`	`CohortDtstmTrans`	`StateVals`	`StateVals`
`Psm`	`PsmCurves`	`StateVals`	`StateVals`
`IndivCtstm`	`IndivCtstmTrans`	`StateVals`	`StateVals`

The disease and state value models can either be instantiated from parameter or model objects using `S3` generic functions prefixed by "`create`" or using the **R6** constructor method `$new()`. `create_IndivCtstmTrans()` is an example of the former that can be used to create an `IndivCtstmTrans` object from a `flexsurvreg`, `flexsurvreg_list`, `params_surv`, or `params_surv_list` object. Similarly, a `StateVals` object can, for example, be created from a `stateval_tbl` object with `create_StateVals()`. The complete economic model is instantiated by combining the submodels using `$new()`.

Each economic model contains methods for simulating disease progression, QALYs, and costs (Table 4). The utility and cost models always simulate QALYs and costs from the simu-

lated progression of disease with the methods `$sim_qalys()` and `$sim_costs()`, respectively. Methods for simulating disease progression differ slightly since the simulation approaches differ as described in Section 2. In an N-state PSM $n-1$ survival curves are generated and in a iCTSTM a trajectory through the STM is simulated for multiple patients via individual simulation. All models can simulate state probabilities over time.

Table 4: Methods for simulating outcomes from economic models

Economic model	Disease progression	QALYs	Costs
`CohortDtstm`	`sim_stateprobs()`	`sim_qalys()`	`sim_costs()`
`Psm`	`sim_survival()`, `sim_stateprobs()`	`sim_qalys()`	`sim_costs()`
`IndivCtstm`	`sim_disease()`, `sim_stateprobs()`	`sim_qalys()`	`sim_costs()`

4. Illustrative example

We consider a three-state model commonly used in oncology. As shown in Figure 2, the three health states are stable disease (i.e. not progressed), progression, and death. Patients can transition to a more severe health state (stable → progression, stable → death, and progression → death) but cannot recover to a less severe health state. We assume that patients remain on first line (1L) treatment until progression, at which time they switch to a second line (2L) treatment.

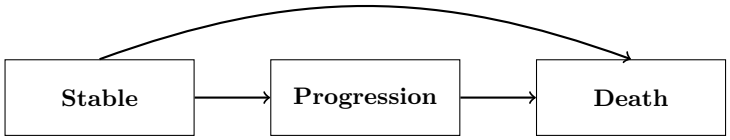


Figure 2: Three state model of disease progression in oncology

This model can be simulated using any of the model types described in Section 2. We demonstrate use of all three below, starting with the iCTSTM.

4.1. Individual continuous time state transition model

We setup the model by defining the model structure, target population, and treatment strategies of interest. The transitions are characterized by a matrix where each element denotes a transition from a row to a column. If a transition is not possible it is marked with `NA`; otherwise, an integer denotes the transition number.

```
R> tmat <- rbind(
+    c(NA, 1, 2),
+    c(NA, NA, 3),
+    c(NA, NA, NA)
+  )
R> colnames(tmat) <- rownames(tmat) <- c("Stable", "Progression", "Dead")
R> print(tmat)

            Stable Progression Dead
Stable          NA           1    2
```

```
Progression     NA          NA    3
Dead            NA          NA   NA
```

The target population, treatment strategies, and (non-death) health states are stored in a `hesim_data` object and identified with integer valued identification variables (`patient_id`, `strategy_id`, and `state_id`). There are 3 treatment strategies: standard of care (SOC) and two new interventions (New 1 and New 2). The target population consists of a heterogeneous population of 1,000 patients, which is large enough to ensure that averages across patients in the individual simulation are reasonably stable. As we will see in the next section, patients can also be placed into subgroups (with a `grp_id` column) or given weights (with a `patient_wt` column) in the `patients` table; if left unspecified as done here, then all patients are assumed to belong to a single subgroup and are weighted equally.

```
R> library("hesim")
R> library("data.table")
R> n_patients <- 1000
R> patients <- data.table(
+    patient_id = 1:n_patients,
+    age = rnorm(n_patients, mean = 45, sd = 7),
+    female = rbinom(n_patients, size = 1, prob = .51)
+  )
R>
R> states <- data.table(
+    state_id = c(1, 2),
+    state_name = c("Stable", "Progression") # Non-death health states
+  )
R>
R> strategies <- data.frame(
+    strategy_id = 1:3,
+    strategy_name = c("SOC", "New 1", "New 2"),
+    soc = c(1, 0, 0),
+    new1 = c(0, 1, 0),
+    new2 = c(0, 0, 1)
+  )
R>
R> hesim_dat <- hesim_data(
+    strategies = strategies,
+    patients = patients,
+    states = states
+  )
R> print(hesim_dat)

$strategies
  strategy_id strategy_name soc new1 new2
1           1           SOC   1    0    0
2           2         New 1   0    1    0
3           3         New 2   0    0    1
```

```
$patients
      patient_id      age female
   1:          1 46.26366      1
   2:          2 50.49314      1
   3:          3 35.52785      1
   4:          4 58.88309      0
   5:          5 53.66930      0
  ---
 996:        996 54.60555      0
 997:        997 50.10654      0
 998:        998 45.02035      0
 999:        999 45.60573      1
1000:       1000 39.45426      0

$states
   state_id  state_name
1:        1      Stable
2:        2 Progression

attr(,"class")
[1] "hesim_data"
```

Labels for the identification variables in a `hesim_data` object can be generated with `get_labels()`. These can (as will be shown below), in turn, be passed to plotting and summary functions to create tables and figures with more informative labels.

```
R> labs_indiv <- get_labels(hesim_dat)
R> print(labs_indiv)

$strategy_id
  SOC New 1 New 2
    1     2     3

$state_id
     Stable Progression       Death
          1           2           3
```

Parameterization

In a STM, the disease model governs transitions between health states. For now, we assume that individual patient data (IPD) with exact transition times is available so that a multi-state statistical model can be used to parameterize the disease model.

Multi-state data consists of one row for each possible transition from a given health state where at most one transition is observed and all others are censored. We use the simulated dataset `onc3` from the **hesim** package.

```
R> onc3[patient_id %in% c(1, 2)]
```

```
          from          to strategy_name female      age patient_id
1:      Stable Progression         New 2      0 59.85813          1
2:      Stable       Death         New 2      0 59.85813          1
3: Progression       Death         New 2      0 59.85813          1
4:      Stable Progression         New 2      0 62.57282          2
5:      Stable       Death         New 2      0 62.57282          2
   time_start time_stop status transition_id strategy_id      time
1:   0.000000  2.420226      1             1           3  2.420226
2:   0.000000  2.420226      0             2           3  2.420226
3:   2.420226 14.620258      1             3           3 12.200032
4:   0.000000  7.497464      0             1           3  7.497464
5:   0.000000  7.497464      0             2           3  7.497464
```

Multi-state models can be fit in one of two ways. First, a joint survival model with interaction terms for different transition can be estimated, which is useful if there are constraints in the parameters, such as coefficients that are assumed equal across transitions. Here, we will take a second approach that is computationally more efficient and fit a separate model for each transition. We illustrate with parametric Weibull models but multiple distributions should be compared and evaluated in a real application (Williams *et al.* 2017). The treatment effect is assumed to only influence transitions from stable disease since patients are assumed to switch to a 2L treatment after progression.

```
R> library("flexsurv")
R> n_trans <- max(tmat, na.rm = TRUE)
R> wei_fits <- vector(length = n_trans, mode = "list")
R> f <- as.formula(Surv(time, status) ~ factor(strategy_name) + female + age)
R> for (i in 1:length(wei_fits)){
+    if (i == 3) f <- update.formula(f, .~.-factor(strategy_name))
+    wei_fits[[i]] <- flexsurvreg(f, data = onc3,
+                                 subset = (transition_id == i),
+                                 dist = "weibull")
+  }
R> wei_fits <- flexsurvreg_list(wei_fits)
```

Utility and cost values are stored in `stateval_tbl` objects. Utility is assumed to equal 0.8 with stable disease (standard error (SE) = 0.02) and 0.6 (SE = 0.05) with progressed disease. Utility values are assumed to follow a beta distribution, which can be parameterized either using its shape parameters or via the mean and standard error (in which case the method of moments is used to derive the shape parameters).

```
R> utility_tbl <- stateval_tbl(
+    data.table(state_id = states$state_id,
+               mean = c(.8, .6),
+               se = c(0.02, .05)
```

```
+    ),
+    dist = "beta")
R> print(utility_tbl)

   state_id mean   se
1:        1  0.8 0.02
2:        2  0.6 0.05
```

Both medical and drug costs are considered. Medical costs are assumed to follow a gamma distribution which is often appropriate for costs since they tend to be right skewed. Like utility, the distribution is characterized by the mean ($2,000 with stable disease, $9,500 with progressed disease) and standard error (assumed equal to the mean) and the method of moments is used to derive the underlying shape and scale parameters.

```
R> medcost_tbl <- stateval_tbl(
+    data.table(state_id = states$state_id,
+               mean = c(2000, 9500),
+               se = c(2000, 9500)
+    ),
+    dist = "gamma")
R> print(medcost_tbl)

   state_id mean   se
1:        1 2000 2000
2:        2 9500 9500
```

Drug costs are assumed to be fixed (i.e., measured without uncertainty). When on 1L treatment, costs are $2,000 with SOC, $12,000 with New 1, and $15,000 with New 2. All patients are assumed to switch to the same treatment at 2L after progression, which costs $1,500 for the first 3 months and $1,200 thereafter.

```
R> drugcost_tbl <- stateval_tbl(
+    drugcost_dt,
+    dist = "fixed")
R> print(drugcost_tbl)

    strategy_id state_id time_id time_start time_stop   est
 1:           1        1       1       0.00      0.25  2000
 2:           1        1       2       0.25       Inf  2000
 3:           1        2       1       0.00      0.25  1500
 4:           1        2       2       0.25       Inf  1200
 5:           2        1       1       0.00      0.25 12000
 6:           2        1       2       0.25       Inf 12000
 7:           2        2       1       0.00      0.25  1500
 8:           2        2       2       0.25       Inf  1200
 9:           3        1       1       0.00      0.25 15000
```

```
10:           3        1       2       0.25       Inf 15000
11:           3        2       1       0.00      0.25  1500
12:           3        2       2       0.25       Inf  1200
```

Simulation

Before constructing each submodel, we specify the number of parameter samples that will be drawn for the PSA.

```
R> n_samples <- 100
```

The input data for the transition model consists of one row for each treatment strategy and patient and can be easily generated from a `hesim_data` object with the `expand` function.

```
R> transmod_data <- expand(hesim_dat,
+                          by = c("strategies", "patients"))
R> head(transmod_data)
```

```
   strategy_id patient_id strategy_name soc new1 new2      age female
1:           1          1           SOC   1    0    0 46.26366      1
2:           1          2           SOC   1    0    0 50.49314      1
3:           1          3           SOC   1    0    0 35.52785      1
4:           1          4           SOC   1    0    0 58.88309      0
5:           1          5           SOC   1    0    0 53.66930      0
6:           1          6           SOC   1    0    0 53.40432      1
```

The transition model is then constructed as a function of the parameters (from the fitted Weibull models) and the input data. We must also specify the allowed transitions, the desired number of PSA samples, the "clock" (recall that we used a clock reset approach when fitting the Weibull models), and the starting age of the patients. Parameters for the PSA are, by default, drawn from the multivariate normal distribution of the maximum likelihood estimate of the regression coefficients, although we make this explicit with the `uncertainty` argument. The starting age of the patients is useful because it allows us to specify a maximum age for patients during the simulation, ensuring that the simulated patients do not live unrealistically long lives.

```
R> transmod <- create_IndivCtstmTrans(wei_fits, transmod_data,
+                                     trans_mat = tmat, n = n_samples,
+                                     uncertainty = "normal",
+                                     clock = "reset",
+                                     start_age = patients$age)
```

The utility and cost models can be easily constructed from the `stateval_tbl` objects with the `create_StateVals()` function, in which the (transformed) parameters are `tparams_mean` objects. The `hesim_data` object provides the information required to ensure that state values are constant within groups not specified within a particular `stateval_tbl`; for instance, since utility only varies by health state, it is assumed constant across patients, treatment strategies, and time. The models for each cost category are combined in a list.

```
R> # Utility
R> utilitymod <- create_StateVals(utility_tbl, n = n_samples,
+                                 hesim_data = hesim_dat)
R>
R> # Costs
R> drugcostmod <- create_StateVals(drugcost_tbl, n = n_samples,
+                                  time_reset = TRUE, hesim_data = hesim_dat)
R> medcostmod <- create_StateVals(medcost_tbl, n = n_samples,
+                                 hesim_data = hesim_dat)
R> costmods <- list(Drug = drugcostmod,
+                   Medical = medcostmod)
```

The complete economic model is an constructed by combining the transition, utility, and cost submodels.

```
R> ictstm <- IndivCtstm$new(trans_model = transmod,
+                           utility_model = utilitymod,
+                           cost_models = costmods)
```

Once the economic model has been constructed, it is straightforward to simulate outcomes. Trajectories through the multi-state model are simulated with the `$sim_disease()` method with patients assumed to live to a maximum age of 100.

```
R> ictstm$sim_disease(max_age = 100)
R> head(ictstm$disprog_)
```

```
   sample strategy_id patient_id grp_id from to final time_start
1:      1           1          1      1    1  3     1   0.000000
2:      1           1          2      1    1  2     0   0.000000
3:      1           1          2      1    2  3     1   7.122132
4:      1           1          3      1    1  2     0   0.000000
5:      1           1          3      1    2  3     1   3.815690
6:      1           1          4      1    1  2     0   0.000000
   time_stop
1:  2.964441
2:  7.122132
3: 12.852926
4:  3.815690
5:  9.809671
6:  3.374204
```

State probabilities (averaged across patients in a given subgroup) can be constructed from the disease progression simulation output with the `$sim_stateprobs()` method. As shown in Figure 3, patients using the newer treatments remain in stable disease longer and have longer expected survival.

```
R> ictstm$sim_stateprobs(t = seq(0, 30 , 1/12))
R> head(ictstm$stateprobs_)
```

```
   sample strategy_id grp_id state_id          t  prob
1:      1           1      1        1 0.00000000 1.000
2:      1           1      1        1 0.08333333 1.000
3:      1           1      1        1 0.16666667 1.000
4:      1           1      1        1 0.25000000 1.000
5:      1           1      1        1 0.33333333 0.999
6:      1           1      1        1 0.41666667 0.998
```

`autoplot()` methods are available to quickly visualize simulation output using **ggplot2** graphics (Wickham 2016). State probabilities from such a plot are shown in Figure 3, with labels for the treatment strategies and health states based on the output of `get_labels()` produced above.

```
R> library("ggplot2")
R> autoplot(ictstm$stateprobs_, labels = labs_indiv,
+           ci = FALSE)
```

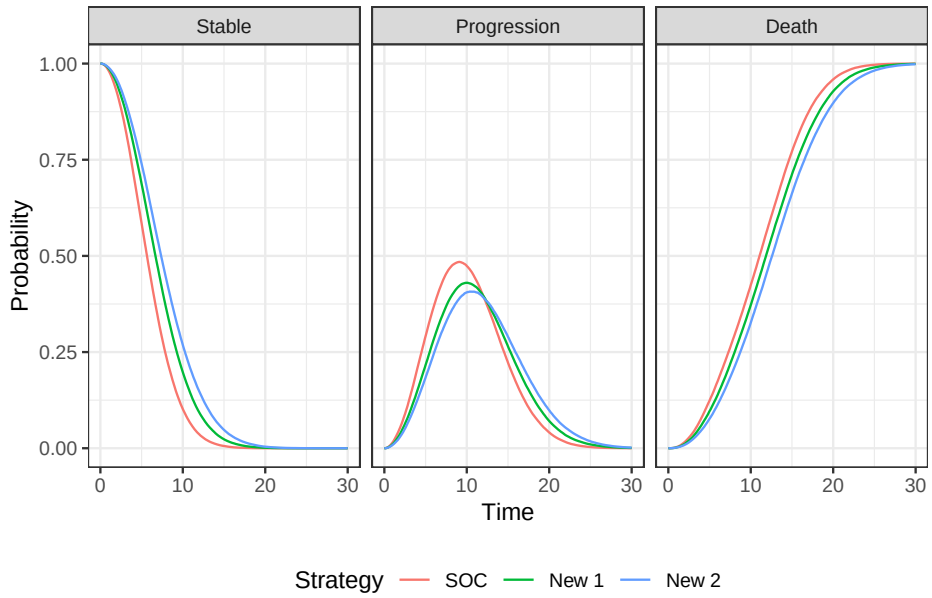


Figure 3: Mean simulated state probabilities from the probabilistic sensitivity analysis

QALYs and costs are simulated from the disease progression simulation output using the utility and cost submodels with the `$sim_qalys()` and `$sim_costs()` methods. Outcomes are simulated for each discount rate specified with the `dr` argument. By default, `$sim_qalys()` will return life-years (i.e., a utility value of 1 for each year of life) in addition to QALYs.

```
R> ictstm$sim_qalys(dr = c(0,.03))
R> head(ictstm$qalys_)
```

```
   sample strategy_id grp_id state_id dr    qalys      lys
1:      1           1      1        1  0 4.806160 6.165409
```

```
2:      1           1      1        2  0 2.865952 4.924144
3:      1           2      1        1  0 5.703347 7.316331
4:      1           2      1        2  0 2.930233 5.034588
5:      1           3      1        1  0 6.545572 8.396750
6:      1           3      1        2  0 2.882091 4.951873
```

```
R> ictstm$sim_costs(dr = .03)
R> head(ictstm$costs_)
```

```
   sample strategy_id grp_id state_id   dr category      costs
1:      1           1      1        1 0.03     Drug  11012.687
2:      1           1      1        2 0.03     Drug   4432.177
3:      1           2      1        1 0.03     Drug  76896.149
4:      1           2      1        2 0.03     Drug   4373.490
5:      1           3      1        1 0.03     Drug 108224.004
6:      1           3      1        2 0.03     Drug   4165.827
```

For a given subgroup, QALYs and costs are simulated for each PSA sample, treatment strategy, and health state. They are summarized with the `$summarize()` method, which first sums across health states and then computes a (weighted) average across patients, by PSA sample, treatment strategy, and subgroup. The output is a generic cost-effectiveness object (of class `ce`) that can be used to perform CEA as demonstrated in Section 5. Here, we use the `summary()` method to summarize the results and make use the "pipes" from the **magrittr** to nicely format the output.

```
R> library("magrittr")
R> ce_sim_ictstm <- ictstm$summarize()
R> summary(ce_sim_ictstm, labels = labs_indiv) %>%
+    format()
```

```
   Discount rate        Outcome                       SOC
1:          0.00          QALYs         7.83 (7.12, 8.63)
2:          0.03          QALYs         6.55 (6.02, 7.15)
3:          0.03    Costs: Drug  15,366 (14,688, 16,257)
4:          0.03 Costs: Medical  49,596 (5,780, 144,325)
5:          0.03   Costs: total 64,962 (20,783, 160,124)
                        New 1                       New 2
1:         8.57 (7.81, 9.45)          9.19 (8.52, 10.10)
2:         7.08 (6.51, 7.70)           7.50 (7.03, 8.16)
3:   78,839 (74,630, 82,241) 105,997 (100,276, 112,298)
4:   48,841 (6,156, 133,816)    49,743 (6,171, 135,256)
5: 127,680 (85,226, 212,775) 155,740 (113,708, 243,946)
```

4.2. Partitioned survival model

In the previous section a multi-state model was fit to IPD. Partitioned survival analysis is an alternative approach that is frequently used for economic evaluations of therapies in oncology. Although there is reason to believe that PSMs are more biased than multi-state models (Woods, Sideris, Palmer, Latimer, and Soares 2020), PSMs are easier to parameterize with aggregate-level data because techniques for fitting multi-state models with aggregate data require more constraints and are less established (Price, Welton, and Ades 2011; Pahuta, Werier, Wai, Patchell, and Coyle 2019; Jansen, Incerti, and Trikalinos 2020). PSMs, in contrast, consist of separate independent survival models that can be estimated with aggregate-level data by reconstructing IPD from digitized survival curves (Guyot, Ades, Ouwens, and Welton 2012).

Since a PSM is a cohort model, the `hesim_data` object must be modified accordingly. We now assume that the cohort can be characterized by four representative patients, each given an equal weight. We consider separate subgroups (identified with `grp_id`) for males and females.

```
R> hesim_dat$patients <- data.table(
+    patient_id = 1:4,
+    grp_id = c(1, 1, 2, 2),
+    patient_wt = rep(1/4, 4),
+    age = c(55, 65, 55, 65),
+    female = c(1, 1, 0, 0),
+    grp_name = rep(c("Female", "Male"),
+                   each = 2)
+  )
```

Since there are now subgroups, we generate new labels that include the subgroup variables.

```
R> labs_cohort <- get_labels(hesim_dat)
```

Parameterization

A multi-state dataset can be converted into dataset with one row per patient and outcomes for PFS and OS using the function `as_pfs_os()`.

```
R> onc_pfs_os <- as_pfs_os(onc3, patient_vars = c("patient_id", "female", "age",
+                                                 "strategy_name"))
R> onc_pfs_os[patient_id %in% c(1, 2)]
```

```
   patient_id female      age strategy_name pfs_time pfs_status
1:          1      0 59.85813         New 2 2.420226          1
2:          2      0 62.57282         New 2 7.497464          0
   os_status   os_time
1:         1 14.620258
2:         0  7.497464
```

Survival models must be fit for both PFS and OS. To imitate a realistic scenario, we assume that IPD is only available for SOC and that relative treatment effects are estimated with aggregate data (e.g., using a NMA). As such, we start by fitting Weibull models for SOC.

One difficulty with partitioned survival analysis is that it does not account for the correlation in the survival endpoints and the curves may consequently cross during the PSA. One way to better account for correlations between the endpoints is to draw PSA samples via bootstrapping whereby the PFS and OS models are refit repeatedly to resamples of the estimation dataset. To allow for bootstrapping we create a `partsurvfit` object, which stores both the survival models and the estimation data.

```
R> onc_pfs_os_soc <- onc_pfs_os[strategy_name == "SOC"]
R> fit_pfs_soc_wei <- flexsurv::flexsurvreg(
+    Surv(pfs_time, pfs_status) ~ female,
+    data = onc_pfs_os_soc,
+    dist = "weibull")
R>
R> fit_os_soc_wei <- flexsurvreg(
+    Surv(os_time, os_status) ~  female,
+    data = onc_pfs_os_soc,
+    dist = "weibull")
R>
R> psmfit_soc_wei <- partsurvfit(
+    flexsurvreg_list(pfs = fit_pfs_soc_wei, os = fit_os_soc_wei),
+    data = onc_pfs_os_soc
+  )
```

The relative treatment effects are available (from a hypothetical external analysis) in terms of regression coefficients for the scale parameter on the log scale. The matrices `coef_pfs_wei` and `coef_os_wei` contain samples of the treatment effects for the PSA. We display the first 6 rows of the PFS coefficients.

```
R> head(coef_pfs_wei)
```

```
          new1      new2
[1,] 0.1772437 0.2900170
[2,] 0.1847671 0.2658466
[3,] 0.1510383 0.2900614
[4,] 0.1916200 0.2755158
[5,] 0.1823881 0.2807196
[6,] 0.1819337 0.2606821
```

The parameters for utility and medical costs remain the same as in the previous section. We do, however, modify drug costs. Since we are no longer using an individual-level model, costs cannot vary as a function of time in a health state. We now assume conservatively that drug costs remain at their initial level when in the progression state.

```
R> drugcost_tbl <- stateval_tbl(
+    drugcost_dt[time_start ==  0][, time_start := NULL],
+    dist = "fixed")
R> print(drugcost_tbl)
```

```
   strategy_id state_id   est
1:           1        1  2000
2:           1        2  1500
3:           2        1 12000
4:           2        2  1500
5:           3        1 15000
6:           3        2  1500
```

Simulation

As with with the multi-state model, input data for the survival models is constructed with the `expand()` function.

```
R> survmods_data <- expand(hesim_dat, by = c("strategies", "patients"))
R> head(survmods_data)
```

```
   strategy_id patient_id strategy_name soc new1 new2 grp_id
1:           1          1           SOC   1    0    0      1
2:           1          2           SOC   1    0    0      1
3:           1          3           SOC   1    0    0      2
4:           1          4           SOC   1    0    0      2
5:           2          1         New 1   0    1    0      1
6:           2          2         New 1   0    1    0      1
   patient_wt age female grp_name
1:       0.25  55      1   Female
2:       0.25  65      1   Female
3:       0.25  55      0     Male
4:       0.25  65      0     Male
5:       0.25  55      1   Female
6:       0.25  65      1   Female
```

Since we did not fit a survival regression model containing both the absolute and relative treatment effects, we must construct the survival parameter object manually. We obtain the parameters of the fitted Weibull regression models with the generic function `create_params()`, and then manually add the relative treatment effects to the coefficients for the scale parameters. (Note that if we had not fit a model for SOC, the survival parameter objects could have still been constructed with `params_surv()`.)

```
R> survmods_params <- create_params(psmfit_soc_wei, n = n_samples,
+                                   uncertainty = "normal")
R> survmods_params$pfs$coefs$scale <- cbind(survmods_params$pfs$coefs$scale,
+                                           coef_pfs_wei)
R> survmods_params$os$coefs$scale <- cbind(survmods_params$os$coefs$scale,
+                                          coef_os_wei)
R> survmods_params$pfs$coefs$scale[1:2, ]
```

```
     (Intercept)      female      new1      new2
[1,]    1.761089 -0.04472303 0.1772437 0.2900170
[2,]    1.802749 -0.13793188 0.1847671 0.2658466
```

Survival curves for both PFS and OS are predicted using the `PsmCurves` class, which is setup using the input data and the parameters of the Weibull model. However, since we manually created the parameter object—meaning that the model terms were not generated using a formula interface—the input data must contain each term contained in the parameter object. We must consequently create a column of ones for the intercept.

```
R> survmods_data[, ("(Intercept)") := 1]
R> survmods <- create_PsmCurves(survmods_params,
+                               input_data = survmods_data)
```

We re-instantiate all of the utility and cost models. This is necessary because our patient population changed and we are now only making predictions for 4 representative patients (rather than 1,000 in the individual simulation). The `hesim_data` argument to `create_StateVals()` does this for us automatically.

```
R> utilitymod <- create_StateVals(utility_tbl, n = n_samples,
+                                 hesim_data = hesim_dat)
R> drugcostmod <- create_StateVals(drugcost_tbl, n = n_samples,
+                                  hesim_data = hesim_dat)
R> medcostmod <- create_StateVals(medcost_tbl, n = n_samples,
+                                 hesim_data = hesim_dat)
R> costmods <- list(Drug = drugcostmod, Medical = medcostmod)
```

Now that the survival, utility, and cost submodels have been constructed, we can instantiate the complete PSM.

```
R> psm <- Psm$new(survival_models = survmods,
+                 utility_model = utilitymod,
+                 cost_models = costmods)
```

We first simulate survival curves by month for 30 years. Survival curves produced using an `autoplot()` method are displayed in Figure 4.

```
R> times <- seq(0, 30, by = .1)
R> psm$sim_survival(t = times)
R> head(psm$survival_)
```

```
   sample strategy_id patient_id grp_id patient_wt curve   t
1:      1           1          1      1       0.25     1 0.0
2:      1           1          1      1       0.25     1 0.1
3:      1           1          1      1       0.25     1 0.2
4:      1           1          1      1       0.25     1 0.3
```

```
5:      1           1           1       1       0.25     1 0.4
6:      1           1           1       1       0.25     1 0.5
    survival
1: 1.0000000
2: 0.9998352
3: 0.9992601
4: 0.9982191
5: 0.9966803
6: 0.9946210
```

```
R> autoplot(psm$survival_,
+           labels = c(labs_cohort,
+                      list(curve = c("PFS" = 1, "OS" = 2))
+                      ),
+           ci = TRUE, ci_style = "ribbon")
```

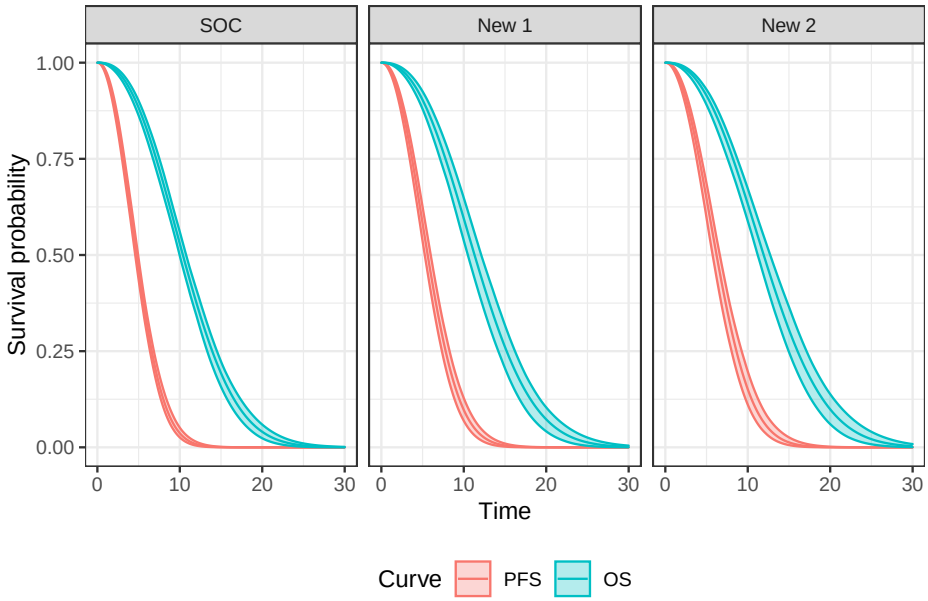


Figure 4: Mean progression free survival and overall survival, weighted by the weight given to each patient in the target population. The solid line and shaded regions represent means and 95% confidence intervals from the probabilistic sensitivity analysis, respectively.

State probabilities are easily computed from these survival curves with the `$sim_stateprobs()` method. The area under the PFS curve is the proportion of patients in stable disease while the area between the PFS and OS curves is the proportion of patient whose disease has progressed.

```
R> psm$sim_stateprobs()
R> psm$stateprobs_[sample == 1 & patient_id == 1 & state_id == 2 & t == 12]
```

```
   sample strategy_id patient_id grp_id patient_wt state_id  t
1:      1           1          1      1       0.25        2 12
2:      1           2          1      1       0.25        2 12
3:      1           3          1      1       0.25        2 12
       prob
1: 0.3627529
2: 0.4054802
3: 0.4247281
```

QALYs and costs are simulated using the same `sim_qalys()` and `sim_costs()` methods as in the previous section, and the results are summarized in the same fashion as well. A difference, however, is that we now use the `by_grp = TRUE` option to summarize costs and QALYs by subgroup (rather than aggregating across all groups).

```
R> psm$sim_qalys(dr = .03)
R> psm$sim_costs(dr = .03)

R> ce_sim_psm <- psm$summarize(by_grp = TRUE)
R> summary(ce_sim_psm, labels = labs_cohort) %>%
+    format()

    Group Discount rate        Outcome                       SOC
1: Female          0.03          QALYs        6.07 (5.51, 6.60)
2: Female          0.03    Costs: Drug  15,205 (14,604, 15,719)
3: Female          0.03 Costs: Medical  49,091 (5,695, 149,686)
4: Female          0.03   Costs: total 64,296 (20,855, 164,972)
5:   Male          0.03          QALYs        6.47 (5.98, 7.02)
6:   Male          0.03    Costs: Drug  16,194 (15,529, 17,017)
7:   Male          0.03 Costs: Medical  49,768 (6,150, 146,128)
8:   Male          0.03   Costs: total 65,962 (22,247, 162,216)
                      New 1                       New 2
1:         6.60 (5.97, 7.21)          6.95 (6.36, 7.60)
2:   66,679 (63,128, 70,154)    88,753 (82,610, 93,432)
3:   50,032 (6,175, 148,545)    50,191 (6,462, 145,719)
4: 116,711 (72,599, 215,138)  138,944 (96,006, 232,798)
5:         7.03 (6.40, 7.62)          7.39 (6.69, 8.04)
6:   74,010 (70,000, 77,613)   98,675 (92,317, 104,870)
7:   50,390 (6,271, 141,608)    50,301 (6,205, 137,616)
8: 124,401 (80,789, 214,873) 148,976 (103,510, 233,099)
```

4.3. Cohort disctete time state transition model

The third model type supported by **hesim** is the cDTSTM. Although it is commonly used in health economics, the approaches taken in Section 4.1 and 4.2 make better use of multi-state data if the time-to-event outcomes are continuously observed. Unfortunately, observations are sometimes only collected at arbitrary times, which results in what is referred to as panel

data. This may, for instance, occur in routine practice oncology data when progression of disease is not continuously evaluated. To facilitate analysis of this type of data, we have converted the `onc3` dataset from above into panel form.

```
R> head(onc3p)
```

```
         state strategy_name female      age patient_id      time
1:      Stable         New 2      0 59.85813          1  0.000000
2: Progression         New 2      0 59.85813          1  2.420226
3:       Death         New 2      0 59.85813          1 14.620258
4:      Stable         New 2      0 62.57282          2  0.000000
5:      Stable         New 2      0 62.57282          2  7.497464
6:      Stable           SOC      1 61.44379          3  0.000000
   strategy_id state_id
1:           3        1
2:           3        2
3:           3        3
4:           3        1
5:           3        1
6:           1        1
```

Parameterization

As in Section 4.2, we consider a case where IPD is available for SOC but aggregate-level data is used to estimate the relative treatment effects. When exact transition times are not observed, the survival models estimated in Section 4.1 are inappropriate. The **msm** package, can, however, be used to estimate models from such data. The loss of information does come at a price as only exponential and piecewise exponential models are possible. We illustrate by fitting an exponential multi-state model for SOC using **msm**. (Note that if exact times were observed and we set the argument `exacttimes = TRUE`, then the parameter estimates would be the same as an exponential model fit using `flexsurvreg()`.)

```
R> library("msm")
R> qinit <- matrix(0, nrow = 3, ncol = 3)
R> qinit[1, 2] <-  0.28; qinit[1, 3] <-  0.013; qinit[2, 3] <-  0.10
R> msm_fit <- msm(state_id ~ time, subject = patient_id,
+                 data = onc3p[strategy_name == "SOC"],
+                 exacttimes = FALSE,
+                 covariates = ~ age + female,
+                 qmatrix = qinit, gen.inits = FALSE)
```

We assume that the relative treatment effects were estimated as relative risks. Either a posterior distribution from a Bayesian analysis or parameters of a parametric distribution can be used. We use the latter approach here and assume that the relative risk is approximately normally distributed on the log scale. **hesim** provides the `define_rng()` and `eval_rng()` functions for defining and evaluating random number generators for a PSA. While standard

functions in R could be used to draw random deviates from probability distributions instead, these functions can be more convenient for health economic modeling. Notable advantages include only needing to specify the number of random samples once, being able to pass moments of distributions as arguments (as in `stateval_tbl()`), and guaranteeing a common output (either a vector or a two-dimensional tabular object, with dimensions determined by the number or random samples).

```
R> params_rr <- list(
+    lrr_12_est = c(soc = log(1), new1 = log(.80), new2 = log(.71)),
+    lrr_12_se = c(soc = 0, new1 = .03, new2 = .04),
+    lrr_13_est = c(soc = log(1), new1 = log(.90), new2 = log(.85)),
+    lrr_13_se = c(soc = 0, new1 = .02, new2 = .03)
+  )
R>
R> params_rr_def <- define_rng({
+    list(
+      rr_12 = lognormal_rng(lrr_12_est, lrr_12_se),
+      rr_13 = lognormal_rng(lrr_13_est, lrr_13_se)
+    )}, n = n_samples)
R> params_rr_rng <- eval_rng(params_rr_def, params_rr)
```

In Sections 4.1 and 4.2, we constructed the disease model using parameter (i.e., `params_surv`) objects. Here, we will use a transformed parameter (i.e., `tparams_transprobs`) object, which is a very flexible way to construct the transition probability matrices for a cDTSTM. A `tparams_transprobs` object requires a 3-dimensional array of transition matrices, where the transition matrices are ordered by PSA sample, treatment strategy, patient, and optionally (in a time-inhomogeneous model) time interval.

We start by predicting transition intensity matrices for SOC from the exponential multi-state model using the same input data used for the survival models in the PSM example. A prediction is made for each representative patient and samples of the transition intensity matrices are drawn for the PSA using an asymptotic normal approximation. The array contains $B \times N_p$ matrices where B is the number of PSA samples and N_p is the number of patients.

```
R> transmod_data <- survmods_data
R> qmat_soc <- qmatrix(msm_fit,
+                      newdata = transmod_data[strategy_name == "SOC"],
+                      uncertainty = "normal", n = n_samples)
R> dim(qmat_soc)

[1]   3   3 400

R> qmat_soc[,, 1]

            [,1]       [,2]        [,3]
[1,] -0.3543684  0.3473318 0.007036617
[2,]  0.0000000 -0.1144912 0.114491243
[3,]  0.0000000  0.0000000 0.000000000
```

Transition probability matrices of arbitrary duration are derived from the transition intensity matrices with the matrix exponential. We use a cycle length of 3 months.

```
R> cycle_len <- 1/4
R> pmat_soc <- expmat(qmat_soc, t = cycle_len)
R> pmat_soc[,, 1]

          [,1]       [,2]        [,3]
[1,] 0.9152188 0.08190242 0.002878769
[2,] 0.0000000 0.97178294 0.028217059
[3,] 0.0000000 0.00000000 1.000000000
```

Identifiers for the complete array of transitions matrices (inclusive of SOC and the new interventions) are generated with `tpmatrix_id()`.

```
R> tpmat_id <- tpmatrix_id(transmod_data, n_samples)
R> head(tpmat_id)

   sample strategy_id patient_id grp_id patient_wt
1:      1           1          1      1       0.25
2:      1           1          2      1       0.25
3:      1           1          3      2       0.25
4:      1           1          4      2       0.25
5:      1           2          1      1       0.25
6:      1           2          2      1       0.25
```

Relative risks can be predicted for each row of the input data and each sample of the coefficients for the PSA using matrix multiplication. In particular, let x be a $N \times k$ matrix where k is the number of predictors and β be a $B \times k$ matrix where each row contains a given sample for the PSA. The function `xbeta` performs the matrix multiplication operation $x\beta^T$ to produce a $N \times B$ matrix of predicted relative risks and then "flattens" the matrix to create a single vector of length $N \times B$.

```
R> xbeta <- function(x, beta) c(x %*% t(beta))
```

We use this function to predict distributions of relative risks (for the transitions stable $\rightarrow$ progression and stable $\rightarrow$ death) for each row of the input data as a function of the three treatment strategies.

```
R> x_rr <- as.matrix(transmod_data[, .(soc, new1, new2)])
R> rr <- cbind(xbeta(x_rr, params_rr_rng$rr_12),
+              xbeta(x_rr, params_rr_rng$rr_13))
R> head(rr) # Corresponds to 1st 6 rows in tpmat_id

          [,1]      [,2]
[1,] 1.0000000 1.0000000
```

```
[2,] 1.0000000 1.0000000
[3,] 1.0000000 1.0000000
[4,] 1.0000000 1.0000000
[5,] 0.8168533 0.8896146
[6,] 0.8168533 0.8896146
```

Relative risks are applied to the elements (stable → progression (1, 2) and stable → death (1, 3)) of each transition probability matrix for SOC with `apply_rr()`. The result is an array of matrices for each PSA sample, treatment strategy, and representative patient, as desired. The `tparams_transprobs` object can then be created by simply combining the transition probability matrices and the identification dataset.

```
R> pmat <- apply_rr(pmat_soc, rr = rr,
+                   index = list(c(1, 2), c(1, 3)))
R> tprobs <- tparams_transprobs(pmat, tpmat_id)
```

Simulation

Once the model has been parameterized, the simulation is straightforward. The transition model depends only on the transition probability matrices and the cycle length,

```
R> transmod <- CohortDtstmTrans$new(params = tprobs,
+                                   cycle_length = cycle_len)
```

and the complete economic model is again instantiated from the transition, utility, and cost submodels.

```
R> cdtstm <- CohortDtstm$new(trans_model = transmod,
+                            utility_model = psm$utility_model,
+                            cost_models = psm$cost_models)
```

We simulate state probabilities for 120 cycles (i.e., 30 years).

```
R> cdtstm$sim_stateprobs(n_cycles = 30/cycle_len)
```

QALYs and costs are again simulated using a 3 percent discount rate

```
R> cdtstm$sim_qalys(dr = .03)
R> cdtstm$sim_costs(dr = .03)
```

and the results are summarized. The `by_grp` argument of `$summarize()` is by default `FALSE` so costs and QALYs are aggregated across subgroups here.

```
R> ce_sim_cdtstm <- cdtstm$summarize()
R> summary(ce_sim_cdtstm, labels = labs_cohort) %>%
+    format()
```

```
   Discount rate        Outcome                       SOC
1:          0.03          QALYs       5.51 (4.86, 6.14)
2:          0.03    Costs: Drug  13,835 (12,394, 14,875)
3:          0.03 Costs: Medical  58,271 (4,702, 196,164)
4:          0.03   Costs: total 72,105 (19,216, 210,501)
                         New 1                      New 2
1:         5.87 (5.05, 6.56)         6.09 (5.26, 6.76)
2:   45,888 (40,015, 49,193)   60,478 (51,131, 66,549)
3:   58,321 (5,266, 190,880)   57,937 (5,681, 182,892)
4: 104,209 (49,893, 238,788) 118,415 (66,374, 244,740)
```

5. Cost-effectiveness analysis

CEA is based on the net monetary benefit (NMB). For a given parameter set θ, the NMB with treatment j is computed as the difference between the monetized health gains from an intervention less costs, or,

$$NMB(j, \theta) = e_j(\theta) \cdot k - c_j(\theta), \tag{8}$$

where e_j and c_j are measures of health outcomes (e.g. QALYs) and costs using treatment j respectively, and k is a decision makers willingness to pay (WTP) per unit of a health outcome. The optimal treatment is the one that maximizes the expected NMB,

$$j^* = \arg\max_j E_\theta \left[NMB(j, \theta) \right]. \tag{9}$$

For a pairwise comparison, treatment 1 is preferred to treatment 0 if the expected incremental net monetary benefit (INMB) is positive; that is, if $E_\theta \left[INMB \right] > 0$ where the INMB is given by

$$INMB(\theta) = NMB(j = 1, \theta) - NMB(j = 0, \theta). \tag{10}$$

Treatments can be compared in an equivalent manner using the incremental cost-effectiveness ratio (ICER). The most common case occurs when a new treatment is more effective and more costly so that treatment 1 is preferred to treatment 0 if the ICER is less than than the willingness to pay threshold k,

$$k > \frac{E_\theta[c_1 - c_0]}{E_\theta[e_1 - e_0]} = ICER. \tag{11}$$

There are three additional cases. Treatment 1 is considered to *dominate* treatment 0 if it is more effective and less costly. Treatment 1 is *dominated* by treatment 0 if it is less effective and more costly. Finally, treatment 1 is preferred to treatment 0 if it is less effective and less costly when $k < ICER$.

5.1. Application

Uncertainty over θ is captured using the PSA. The simulated distribution of QALYs and costs for each treatment strategy can be produced with the `$summarize()` method as was demonstrated in Section 4. We will use the output from the iCTSTM in this example.

```
R> ce_sim_ictstm

$costs
     category   dr sample strategy_id     costs grp_id
  1:     Drug 0.03      1           1  15444.86      1
  2:     Drug 0.03      1           2  81269.64      1
  3:     Drug 0.03      1           3 112389.83      1
  4:     Drug 0.03      2           1  15546.71      1
  5:     Drug 0.03      2           2  79716.43      1
 ---
896:    total 0.03     99           2 119878.89      1
897:    total 0.03     99           3 148166.70      1
898:    total 0.03    100           1  33033.54      1
899:    total 0.03    100           2  97753.20      1
900:    total 0.03    100           3 122663.71      1

$qalys
       dr sample strategy_id    qalys grp_id
  1: 0.00      1           1 7.672112      1
  2: 0.00      1           2 8.633579      1
  3: 0.00      1           3 9.427662      1
  4: 0.00      2           1 8.223988      1
  5: 0.00      2           2 8.774143      1
 ---
596: 0.03     99           2 6.774198      1
597: 0.03     99           3 7.266402      1
598: 0.03    100           1 6.083653      1
599: 0.03    100           2 6.718671      1
600: 0.03    100           3 7.057538      1

attr(,"class")
[1] "ce"
```

A CEA is performed from this output with the `cea()` and `cea_pw()` functions. `cea()` summarizes results by simultaneously accounting for each treatment strategy in the analysis, while `cea_pw()` summarizes "pairwise" results in which each treatment is compared to a comparator (in this case SOC).

```
R> wtp <- seq(0, 250000, 500) # Willingness to pay per QALY
R> cea_ictstm <- cea(ce_sim_ictstm, dr_costs = .03, dr_qalys = .03, k = wtp)
R> cea_pw_ictstm <- cea_pw(ce_sim_ictstm, comparator = 1,
```

```
+                           dr_qalys = .03, dr_costs = .03,
+                           k = wtp)
```

Incremental cost-effectiveness ratio

The `cea_pw()` function computes the ICER. A summary table is produced with `icer()` and `format()` formats the table for pretty printing. The argument `k` is the WTP threshold and is used to compute the INMB. Estimates of incremental QALYs and incremental costs are computed by averaging over PSA samples. By default, 95% confidence intervals are presented and are computed using quantiles from the PSA.

```
R> icer(cea_pw_ictstm, k = 100000, labels = labs_indiv) %>%
+    format()
```

```
             Outcome                      New 1
1: Incremental QALYs        0.53 (0.24, 0.83)
2: Incremental costs  62,718 (52,601, 74,091)
3:   Incremental NMB -9,968 (-36,778, 16,020)
4:              ICER                  118,898
                      New 2
1:        0.95 (0.51, 1.35)
2: 90,778 (78,247, 105,749)
3:  4,246 (-28,891, 35,486)
4:                  95,532
```

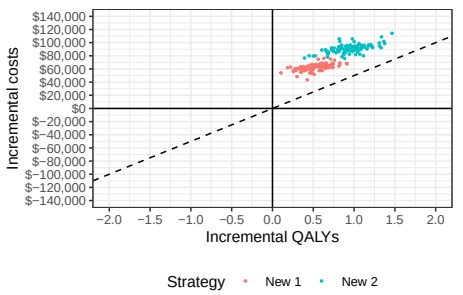
Representing decision uncertainty

A number of common measures can be used to represent decision uncertainty from a CEA, including cost-effectiveness planes, cost-effectiveness acceptability curves (CEAC), cost-effectiveness acceptability frontiers (CEAF), and the expected value of perfect information (EVPI). Plots using **ggplot2** can be quickly generated with the `plot_ceplane()`, `plot_ceac()`, `plot_ceaf()`, and `plot_evpi()` functions, respectively. Output from `cea()` and `cea_pw()` is readily available, so users may create custom plots as well. Example figures from the code that follows are shown in Figure 5.

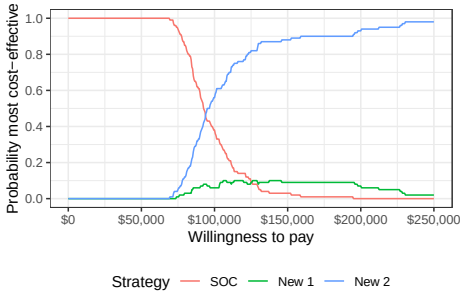
The cost-effectiveness plane plots the incremental effectiveness of a treatment strategy (relative to a comparator) against the incremental cost of the treatment strategy. The plot is useful because it demonstrates both the uncertainty and the magnitude of the estimates. Each point on the plot is from a particular draw from the PSA. Data for plotting a cost-effectiveness plane comes from the `delta` output generated from `cea_pw()`.

```
R> plot_ceplane(cea_pw_ictstm, labels = labs_indiv)
```

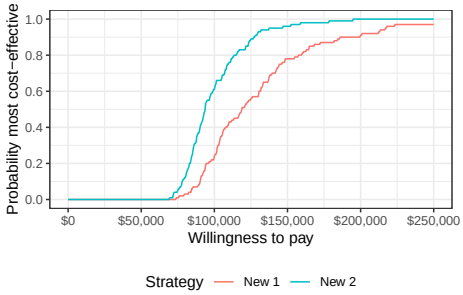
The CEAC can be generated based on simultaneous comparisons of all treatment strategies or by comparing each treatment strategy to a single comparator. The former is the probability that each strategy is the most cost-effective and is available from the `mce` element produced



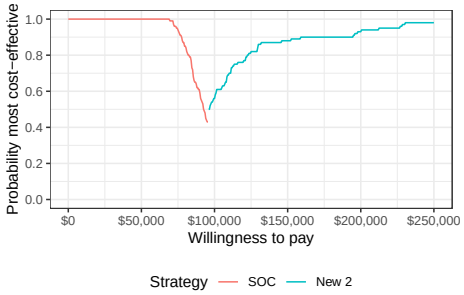
(a) Cost-effectiveness plane



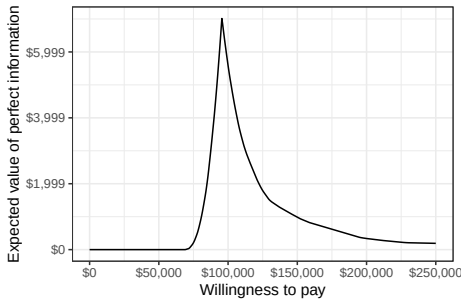
(b) CEAC (simultaneous)



(c) CEAC (pairwise)



(d) CEAF



(e) EVPI

Figure 5: Representations of decision uncertainty

by `cea()`. The relevant information for the latter is available from the `ceac` element produced by `cea_pw()`, and is the probability that each treatment is more cost-effective than the comparator.

```
R> plot_ceac(cea_ictstm, labels = labs_indiv)
```

```
R> plot_ceac(cea_pw_ictstm, labels = labs_indiv)
```

One drawback of the CEAC is that the probability of being cost-effective cannot be used to determine the optimal treatment option. Instead, if the objective is to maximize health gain, then decisions should be based on the expected NMB. The CEAF, which plots the probability that the optimal treatment strategy (i.e., the strategy with the highest expected NMB) is

cost-effective, is appropriate in this context. A CEAF curve can be created by using the `best` column from the `mce` element to subset to the treatment strategy with the highest expected NMB for each WTP value.

```
R> plot_ceaf(cea_ictstm, labels = labs_indiv)
```

A limitation of the prior measures are that they ignore the magnitude of QALY or cost gains. A measure which combines the probability of being most effective with the magnitude of the expected NMB is the EVPI. Intuitively, the EVPI provides an estimate of the amount that a decision maker would be willing to pay to collect additional data and completely eliminate uncertainty. Mathematically, the EVPI is defined as the difference between the maximum expected NMB given perfect information and the maximum expected NMB given current information. In other words, we calculate the NMB for the optimal treatment strategy for each random draw of the parameters and compare that to the NMB for the treatment strategy that is optimal when averaging across all parameters,

$$EVPI = E_\theta \left[\max_j NMB(j, \theta) \right] - \max_j E_\theta \left[NMB(j, \theta) \right]. \tag{12}$$

The `cea()` function performs the EVPI calculation across all simulation draws from the PSA and for a number of WTP values. The kink in the plot represents the WTP value where the optimal strategy changes.

```
R> plot_evpi(cea_ictstm, labels = labs_indiv)
```

6. Discussion

This paper formally describes the economic models and cost-effectiveness framework used by the **hesim** package. An illustrative example is provided for a three state model commonly used to evaluate treatments in oncology. However, since **hesim** aims to be flexible, the example only covers a subset of the relevant modeling approaches and functionality. Additional examples as well as documentation for all functions and classes are consequently provided on the package website (https://hesim-dev.github.io/hesim/).

One notable set of features not covered relates to the construction of transition probability matrices, the core of any cDTSTM. In Section 4.3, an approach was used that was tailored to a case in which transition intensity matrices were available for a reference treatment and relative treatment effects (i.e., relative risks) were available for the remaining competing interventions. To maximize efficiency, the matrix exponential was only applied where necessary (for SOC) and relative risks were applied to the transition probability matrices for SOC with the efficient `apply_rr()` function.

Alternative approaches might be appropriate in other contexts. For example, if evenly spaced panel data is available, a multinomial logistic regressions can be used to predict transition probabilities. On the other hand, some users may prefer a functional approach that is arguably more human readable. To facilitate this type of analysis, a `define_model()` block can be used to construct transition probability matrices using nonstandard evaluation. Each possible combination of the parameters and input data (treatment strategies and patients) is automatically

constructed and the user can write custom expressions that create the transition probability matrices as a function of the parameters and input data. The same `define_model()` block may also be used to construct utility and cost models, if desired. Examples that leverage multinomial logistic regression and `define_model()` are provided on the package website. The latter is used to reproduce two examples (a simple Markov cohort model of an HIV treatment and a time-inhomogeneous Markov cohort model of total hip replacement) from the reference textbook by Briggs, Sculpher, and Claxton (2006).

In Section 4.1, we fit a clock-reset multi-state model and simulated outcomes with an individual-level simulation. An iCTSTM can, however, also be developed without fitting a statistical model using R by manually inputting the arguments (e.g., coefficients, probability distributions) of `params_surv` objects for each possible transition. If IPD is available for the reference treatment, `create_params()` can be use to obtain the baseline `params_surv` object from a fitted multi-state model, and relative treatment effects (e.g., hazard ratios) could be added as additional columns to the coefficients stored in the `params_surv` object. A similar approach was used in the partitioned survival analysis example above (Section 4.2). A re-analysis of the total hip replacement model from the Briggs *et al.* (2006) textbook using an individual-level simulation is provided on the package website to illustrate.

A third feature worth mentioning is that a PSM can be extended to N health states. The 3-state oncology case might, for instance, be extended to account for adverse events or multiple lines of therapy. One related application is the quality-adjusted time without symptoms or toxicity (Q-TWiST) method (Goldhirsch, Gelber, Simes, Glasziou, and Coates 1989; Lenderking, Gelber, Cotton, Cole, Goldhirsch, Volberding, and Testa 1994), which partitions patients into time spent (i) having toxic side effects, (ii) without symptoms or toxicities, and (iii) after progression. An example four-state model is provided on the package website.

As mentioned in the introduction, `heemod` is the only other general purpose R package for cost-effectiveness modeling. It differs from **hesim** in that it focuses solely on cDTSTMs (i.e., Markov cohort models) and is not designed to facilitate integration of economic models with the statistical models used for parameter estimation. Another important difference is that `hesim` is considerably faster since all simulations are vectorized (i.e, performed at the C++ level). Performance comparisons are provided on the package website. The time-inhomogeneous total hip replacement example from Briggs *et al.* (2006) was run with both packages using 1,000 PSA iterations. A cohort model in **heemod** completed in approximately 85 seconds while the same cohort model took approximately 1 second to run in **hesim** and an equivalent individual-level simulation in `hesim` ran in 9 seconds.

There are a number of multi-state modeling R packages that can both fit models and make predictions. **hesim** leverages some of these for parameters estimation: **flexsurv** is used to fit parametric and flexible parametric survival and multi-state models when exact event times are available while **msm** is used to fit multi-state models when only panel data is available. A key difference is that **hesim** is designed specifically for health-economic modeling whereas the other packages are more general purpose statistical packages.

mstate is another package for multi-state modeling that is designed for estimation of non-parametric and semi-parametric (i.e., Cox proportional hazards) models (de Wreede, Fiocco, Putter *et al.* 2011). It supports prediction of state probabilities and simulation of multi-state trajectories (to facilitate prediction for clock-reset models) from cumulative hazard functions evaluated at specific times. Williams *et al.* (2017) has adapted **mstate** for cost-effectiveness

modeling applications. There are however, some disadvantages of using **mstate** for health economic modeling since it was not designed for it. `mssample()`—the core function for simulating multi-state models—does not natively support heterogeneous target populations, PSA, simultaneous simulation of multiple treatment strategies, or computation of QALYs and costs from the simulated output. It is less computationally efficient than **hesim** for two main reasons. First, simulation of general cumulative hazards functions is considerably slower than simulation of parametric distributions where efficient random number generators are available. Second, it is written completely in R and incorporating PSA, multiple treatment strategies, or heterogeneous patients results in code that is not vectorized. To compare computational performance, we ran 100 PSA iterations where 1 treatment strategy and 1,000 homogeneous patients were simulated through a 6-state Weibull model during each iteration. The model completed in 0.44 second in **hesim** and took 34 minutes in **mstate**, showing that computational considerations are not just an academic concern.

A number of extensions of the current version of **hesim** would be helpful. First, we plan to extend the individual-level simulation so that users can update input data during the simulation as a function of time elapsed and/or the health states transitioned to. This would allow potentially time-varying variables such as age to update each time a patient transitions to a new state. Second, we would like to add more examples to ensure that **hesim** is fit for purpose for a wide range of applications in health economics. Third, more complex models of utility (Kharroubi, Brazier, Roberts, and O'Hagan 2007; Hernández Alava, Wailoo, Wolfe, and Michaud 2013) and costs (Nixon and Thompson 2005) could help better capture heterogeneity in preferences and spending. Fourth, it would be useful to add functionality that automatically creates tunnel states for users in cDTSTMs. Finally, a larger aspiration is to provide support for parameterization via NMA, particularly for estimation of relative treatment effects in survival (e.g., for PSMs) and multi-state models. However, since existing R packages are lacking in this area, this could involve creation of a separate NMA package and a new dependency for **hesim**.

7. Conclusion

hesim is a modular and computationally efficient R package for development and analysis of simulation models for economic evaluation. Discrete time cohort and continuous time individual-level models are supported, encompassing Markov (time-homogeneous and time-inhomogeneous) and semi-Markov processes as well as partitioned survival analysis. To facilitate integration of statistical methods and economic models and use of both patient- and aggregate-level data, models can be parameterized by either fitting statistical models using R, inputting values directly, or from a combination of the two. Simulated costs and QALYs from a PSA can be used for for decision-analysis within a cost-effectiveness framework, allowing users to represent decision uncertainty (e.g., cost-effectiveness planes, cost-effectiveness acceptability curves, and cost-effectiveness acceptability frontiers), and conduct value of information analysis. Algorithms are efficiently coded and written in C++ so that approaches that have been considered computationally intensive such as individual-level simulations and PSA can run very quickly. The modular design makes modeling more flexible and it easier to add new features.

References

Aalen OO, Johansen S (1978). "An empirical transition matrix for non-homogeneous Markov chains based on censored observations." *Scandinavian Journal of Statistics*, pp. 141–150.

Baio G (2012). *Bayesian methods in health economics*. CRC Press.

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(1), 1–47.

Baio G, Berardi A, Heath A (2017). *Bayesian cost-effectiveness analysis with the R package BCEA*. Springer.

Baio G, Heath A (2017). "When simple becomes complicated: why excel should lose its place at the top table."

Brennan A, Chick SE, Davies R (2006). "A taxonomy of model structures for economic evaluation of health technologies." *Health economics*, **15**(12), 1295–1310.

Briggs A, Sculpher M (1998). "An introduction to Markov modelling for economic evaluation." *Pharmacoeconomics*, **13**(4), 397–409.

Briggs A, Sculpher M, Claxton K (2006). *Decision modelling for health economic evaluation*. Oup Oxford.

Claxton K, Sculpher M, McCabe C, Briggs A, Akehurst R, Buxton M, Brazier J, O'Hagan T (2005). "Probabilistic sensitivity analysis for NICE technology assessment: not an optional extra." *Health economics*, **14**(4), 339–347.

Cox DR, Miller HD (1977). *The theory of stochastic processes*, volume 134. CRC press.

Dakin H, Devlin N, Feng Y, Rice N, O'Neill P, Parkin D (2015). "The influence of cost-effectiveness and other factors on nice decisions." *Health economics*, **24**(10), 1256–1271.

de Wreede LC, Fiocco M, Putter H, *et al.* (2011). "mstate: an R package for the analysis of competing risks and multi-state models." *Journal of statistical software*, **38**(7), 1–30.

Dias S, Ades AE, Welton NJ, Jansen JP, Sutton AJ (2018). *Network meta-analysis for decision-making*. John Wiley & Sons.

Filipović-Pierucci A, Zarca K, Durand-Zaleski I (2017). "Markov Models for Health Economic Evaluations: The R Package heemod." *arXiv preprint arXiv:1702.03252*.

Fiocco M, Putter H, van Houwelingen HC (2008). "Reduced-rank proportional hazards regression and simulation-based prediction for multi-state models." *Statistics in Medicine*, **27**(21), 4340–4358.

Glasziou P, Simes R, Gelber R (1990). "Quality adjusted survival analysis." *Statistics in medicine*, **9**(11), 1259–1276.

Goldhirsch A, Gelber RD, Simes RJ, Glasziou P, Coates AS (1989). "Costs and benefits of adjuvant therapy in breast cancer: a quality-adjusted survival analysis." *Journal of Clinical Oncology*, **7**(1), 36–44.

Guyot P, Ades A, Ouwens MJ, Welton NJ (2012). "Enhanced secondary analysis of survival data: reconstructing the data from published Kaplan-Meier survival curves." *BMC medical research methodology*, **12**(1), 9.

Hernández Alava M, Wailoo A, Wolfe F, Michaud K (2013). "The relationship between EQ-5D, HAQ and pain in patients with rheumatoid arthritis." *Rheumatology*, **52**(5), 944–950.

Incerti D, Thom H, Baio G, Jansen JP (2019). "R you still using excel? The advantages of modern software tools for health technology assessment." *Value in Health*, **22**(5), 575–579.

Jackson CH (2016). "flexsurv: a platform for parametric survival modeling in R." *Journal of statistical software*, **70**.

Jalal H, Pechlivanoglou P, Krijkamp E, Alarid-Escudero F, Enns E, Hunink MM (2017). "An overview of R in health decision sciences." *Medical decision making*, **37**(7), 735–746.

Jansen JP, Incerti D, Trikalinos T (2020). "Multi-state network meta-analysis of cause-specific survival data." *medRxiv*.

Kharroubi SA, Brazier JE, Roberts J, O'Hagan A (2007). "Modelling SF-6D health state preference data using a nonparametric Bayesian method." *Journal of health economics*, **26**(3), 597–612.

Lenderking WR, Gelber RD, Cotton DJ, Cole BF, Goldhirsch A, Volberding PA, Testa MA (1994). "Evaluation of the quality of life associated with zidovudine treatment in asymptomatic human immunodeficiency virus infection." *New England Journal of Medicine*, **330**(11), 738–743.

Nixon RM, Thompson SG (2005). "Methods for incorporating covariate adjustment, subgroup analysis and between-centre differences into cost-effectiveness evaluations." *Health economics*, **14**(12), 1217–1229.

O'Hagan A, Stevenson M, Madan J (2007). "Monte Carlo probabilistic sensitivity analysis for patient level simulation models: efficient estimation of mean and variance using ANOVA." *Health economics*, **16**(10), 1009–1023.

Pahuta MA, Werier J, Wai EK, Patchell RA, Coyle D (2019). "A technique for approximating transition rates from published survival analyses." *Cost Effectiveness and Resource Allocation*, **17**(1), 12.

Price MJ, Welton NJ, Ades A (2011). "Parameterization of treatment effects for meta-analysis in multi-state Markov models." *Statistics in Medicine*, **30**(2), 140–151.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in biostatistics: competing risks and multi-state models." *Statistics in medicine*, **26**(11), 2389–2430.

Strong M, Oakley JE, Brennan A (2014). "Estimating multiparameter partial expected value of perfect information from a probabilistic sensitivity analysis sample: a nonparametric regression approach." *Medical Decision Making*, **34**(3), 311–326.

Torrance GW (1986). "Measurement of health state utilities for economic appraisal: a review." *Journal of health economics*, **5**(1), 1–30.

Trosman JR, Van Bebber SL, Phillips KA (2011). "Health technology assessment and private payers' coverage of personalized medicine." *Journal of oncology practice*, **7**(3S), 18s–24s.

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

Williams C, Lewsey JD, Briggs AH, Mackay DF (2017). "Cost-effectiveness analysis in R using a multi-state modeling survival analysis framework: a tutorial." *Medical Decision Making*, **37**(4), 340–352.

Woods B, Sideris E, Palmer S, Latimer N, Soares M (2018). "NICE DSU technical support document 19. Partitioned survival analysis for decision modelling in health care: a critical review. 2017." *Available from: www.. nicedsu. org. uk/wp-content/uploads/2017/06/Partitioned-Survival-Analysisfinal-report. pdf*.

Woods BS, Sideris E, Palmer S, Latimer N, Soares M (2020). "Partitioned Survival and State Transition Models for Healthcare Decision Making in Oncology: Where Are We Now?" *Value in Health*.

Affiliation:

Devin Incerti
Genentech
South San Francisco
E-mail: incerti.devin@gene.com

Jeroen Jansen
University of California, San Francisco
San Francisco
E-mail: jeroen.jansen@ucsf.edu