# gems: An R Package for Simulating from Disease Progression Models

**Nello Blaser**[*]
University of Bern

**Luisa Salazar Vizcaya**[*]
University of Bern

**Janne Estill**
University of Bern

**Cindy Zahnd**
University of Bern

**Bindu Kalesan**
University of Bern
Columbia University

**Matthias Egger**
University of Bern
University of Cape Town

**Thomas Gsponer**[†]
University of Bern

**Olivia Keiser**[†]
University of Bern

**Abstract**

Mathematical models of disease progression predict disease outcomes and are useful epidemiological tools for planners and evaluators of health interventions. The R package **gems** is a tool that simulates disease progression in patients and predicts the effect of different interventions on patient outcome. Disease progression is represented by a series of events (e.g., diagnosis, treatment and death), displayed in a directed acyclic graph. The vertices correspond to disease states and the directed edges represent events. The package **gems** allows simulations based on a generalized multistate model that can be described by a directed acyclic graph with continuous transition-specific hazard functions. The user can specify an arbitrary hazard function and its parameters. The model includes parameter uncertainty, does not need to be a Markov model, and may take the history of previous events into account. Applications are not limited to the medical field and extend to other areas where multistate simulation is of interest. We provide a technical explanation of the multistate models used by **gems**, explain the functions of **gems** and their arguments, and show a sample application. This manuscript was published in the Journal of Statistical Software Blaser *et al.* (2015).

*Keywords*: Monte Carlo simulation, multistate model, R, survival analysis, prediction, compartmental model.

## 1. Introduction

We present a simulation algorithm and the R package **gems** (Salazar Vizcaya *et al.* 2013) for simulating from a multistate model with arbitrary transition-specific hazard functions.

In epidemiology, mathematical models of disease progression are useful for predicting disease

[*]The first two authors listed on the paper contributed equally to the development of **gems** and to the writing of this manuscript.

[†]The last two authors listed on the paper contributed equally to this manuscript.

outcomes and for planning and evaluating interventions (Garnett *et al.* 2011). Disease progression is often characterized by a series of events, such as diagnosis, treatment and death. From this characterization, disease progression can be displayed in a directed acyclic graph (DAG) (Pearl 2009), where disease states are denoted by vertices and the directed edges connecting them correspond to the events.

Traditional compartmental models of infectious diseases assume that transition times between the different stages of a disease are exponentially distributed (Anderson and May 1992). The use of exponential transition times has the advantage that models can be formulated deterministically with ordinary differential equations. Exponential times can also be simulated using the Gillespie algorithm (Gillespie 1977). However, the distribution of transition times between states is often not exponential (Lloyd 2001). Although it is possible to divide states into substates, so that an exponential transition-specific hazard fits the data for those substates, this approach is inflexible. Typical model structures using non-exponential transition times are agent-based stochastic simulation models (Estill *et al.* 2012; Phillips *et al.* 2011). For instance one study used history-dependent Weibull distributed transition times to investigate the effect on HIV transmission of bringing patients lost to follow-up back into care (Estill *et al.* 2014). This study found that 116 tracing efforts were needed to prevent one new infection. Agent-based models usually apply to one specific disease and include a limited number of interventions. We are not aware of any agent-based model structure that can be applied simultaneously to different diseases and interventions. We therefore propose a more flexible simulation algorithm that can simulate from any DAG.

We developed a multistate model that allows disease progression to be monitored in a cohort of individual patients, and takes into account the history of previous events. The R package **gems** allows simulation from a directed acyclic multistate model with general transition-specific hazard functions. The package simplifies definition of the multistate model, its relevant transition-specific functions, its parameters, and their uncertainty. It also calculates transition probabilities and cumulative incidences, and thus facilitates analysis of the simulated cohorts. The R package **gems** is used for simulation and not parameterization of multistate models. To parameterize the transition-specific hazard functions, the R packages **survival** (Therneau 2014), **mstate** (de Wreede *et al.* 2011) and **muhaz** (Hess and Gentleman 2010) can be used.

In Section 2 we present a mathematical description of the multistate model. We present the simulation from this model and demonstrate the inclusion of parameter uncertainty. In Section 3, we describe the use of **gems** in detail, providing explanations for and examples of all the important package functions. In Section 4 we present a case study in cardiology. Finally, in Section 5, we discuss the strengths and limitations of the package.

## 2. Technical description of the simulation model

We describe a directed acyclic multistate model and the algorithm used in **gems** to simulate from it. For a general introduction to multistate models, see Putter *et al.* (2007).

### 2.1. General setup of the multistate model

A multistate model consists of a set of states and the transitions between them. The states can be divided into three groups: initial states, intermediate states and absorbing states.

**gems** only considers multistate models without loops, that is models, which can be written as a directed acyclic graph (DAG) (Pearl 2009). A DAG consists of states and the directed edges that connect them, so that no sequence of directed edges can connect back to a previous state.

Consider a directed acyclic multistate model with $n$ states $E_1, \ldots, E_n$, where a transition from state $E_i$ to state $E_j$ is only possible if $i < j$. Let $(X_t)_{t \geq 0}$ be the stochastic process that describes the progression through the different states. It is an $\mathcal{E} = \{E_1, \ldots, E_n\}$-valued jump process with jump times given by

$$S_i = \inf \{t \geq 0 \mid X_t = E_i\} \tag{1}$$

for states $E_i$ that are visited, and $S_i = \infty$ otherwise. Transition times to state $E_j$ from the previous state are defined by $T_j = S_j - S_{\max\{k \mid k<j, S_k<\infty\}}$, where $S_0 = 0$ by convention.

The entire process is determined by transition times $T_j$ to state $E_j$, described by transition-specific hazard functions $h_{ij}$ as

$$T_{ij} \sim F_{ij}(t) = 1 - \exp\left\{-\int_0^t h_{ij}(u)\, du\right\}, \tag{2}$$

$$T_j = \min_{i \in \{1, \ldots j-1 \mid T_i < \infty\}} T_{ij}, \tag{3}$$

where $F_{ij}$ is the cumulative distribution function of the transition time from state $E_i$ to state $E_j$. See Figure 1 for a graphical representation of these hazard functions and transition times. Unless all hazards are constant, $X$ does not have a Markovian structure.

*Hazards and transition probabilities*

Consider the relatively simple multistate model described in the DAG in Figure 1.

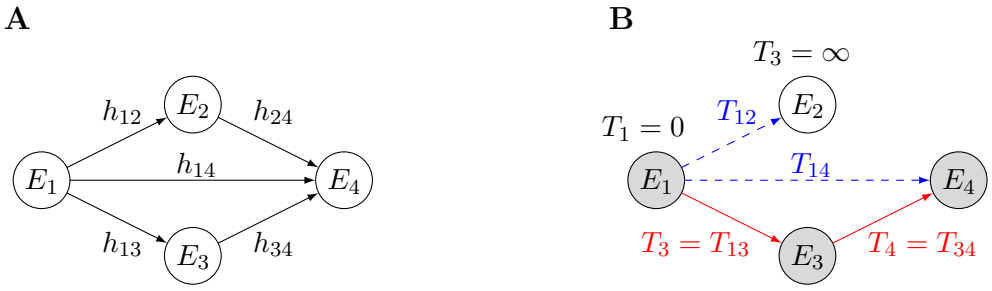


Figure 1: Sample DAG of states and transitions. Panel A shows the states and transition hazards. Panel B shows the potential transition times $T_{ij}$ and the transition times $T_j$ of a simulation where $T_{13} < T_{12}$. The red solid lines depict the path taken in this particular simulation and the gray circles represent the states visited by the individual. The blue dashed lines represent the potential transitions that never occurred.

This exemplary model consists of an initial state $E_1$, two intermediate states $E_2$, $E_3$ and one absorbing state $E_4$. The transition probabilities from state $E_i$ at time $s$ to state $E_j$ at time $t$

$$p_{ij}(s,t) = \mathsf{P}\left[X_t = E_j \mid X_s = E_i\right], \qquad \text{for } s \leq t \tag{4}$$

can then be calculated from the transition-specific hazard functions as follows.

The probabilities from state $E_4$ are $p_{44}(s,t) = 1$ and $p_{4j}(s,t) = 0$ for all $j \neq 4$.

From states $E_2, E_3$, the only two possibilities are to remain in the current state or move to state $E_4$, so the transition probabilities are

$$p_{ii}(s,t) = \exp\left\{-\int_{s-S_i}^{t-S_i} h_{i4}(u)\,du\right\}, \qquad \text{for } i \in \{2,3\}, \tag{5}$$

$$p_{i4}(s,t) = 1 - \exp\left\{-\int_{s-S_i}^{t-S_i} h_{i4}(u)\,du\right\}, \qquad \text{for } i \in \{2,3\}, \tag{6}$$

$$p_{ij}(s,t) = 0, \qquad \text{for } i \in \{2,3\},\ j \notin \{i,4\}. \tag{7}$$

The transition probabilities from the initial state are already difficult to solve analytically. Assuming $S_1 = 0$, the transition probabilities can be calculated from the integrals

$$p_{11}(s,t) = \exp\left\{-\int_s^t h_{12}(u)\,du - \int_s^t h_{13}(u)\,du - \int_s^t h_{14}(u)\,du\right\}, \tag{8}$$

$$p_{12}(s,t) = \int_s^t p_{11}(s,u)h_{12}(u)p_{22}(u,t)\,du, \tag{9}$$

$$p_{13}(s,t) = \int_s^t p_{11}(s,u)h_{13}(u)p_{33}(u,t)\,du, \tag{10}$$

$$p_{14}(s,t) = 1 - p_{11}(s,t) - p_{12}(s,t) - p_{13}(s,t). \tag{11}$$

Intuitively, these formulas express that the process $X$ remains in state $E_1$ from time $s$ to time $u$. Then it moves to state $E_2$ or $E_3$ respectively, where it remains until time $t$. The transition probabilities $p_{12}$ and $p_{13}$ can then be calculated as the integral over $u$. These integrals become increasingly difficult to solve when there are more states, and they cannot usually be solved analytically. The **gems** package uses Monte Carlo methods to simulate the transition times associated to those probabilities.

## 2.2. Simulating from hazard functions

In this Section we describe the methods used in the package **gems** to simulate from a transition-specific hazard function for one agent. For each state $E_i$, all transition-specific hazard functions and their parameters must be specified. For instance, an exponentially distributed transition with mean $\mu = 2$ can be specified as a constant function $h(t) = \frac{1}{\mu}$ with parameter $\mu = 2$, or equivalently if specifying $h(t) = r$ with parameter $r = \frac{1}{2}$. For the description in this Section, the choice of parameterization is arbitrary, but it will be relevant in Section 2.3 where we consider parameter uncertainty.

It is possible to simulate the times $T_{ik}$ from the hazard functions, as explained below. By taking the minimum over all $k$, we get the transition time $T_j$, and the corresponding state $E_j$. To simulate $X$, we therefore start by simulating from the initial state $T_{1k}$ and calculate the first transition time by taking the minimum. Then we continue the simulation from the corresponding state $E_j$. This procedure is iterated until an absorbing state is reached, at which point the simulation ends.

In order to simulate from a hazard function, we approximate the specified hazard function $h(t)$ by a piecewise constant function $h_{pc}(t)$. Then we use the `rpexp` function of the **msm** package

(Jackson 2011) to simulate from the piecewise constant approximation of the hazard function $h_{pc}(t)$. The `rpexp` function generates random variables from an exponential distribution with piecewise constant rates.

To calculate the transition probabilities from the initial state at time $t = 0$, the process $X$ is simulated for $N$ agents (i.e., patients in the context of a cohort). At each time point the proportion of simulated patients that are in any state at that time is calculated. For large enough $N$ these proportions approximate the transition probabilities from the initial state at time $t = 0$.

### 2.3. Including uncertainty into the multistate model

The exact parameters of transition-specific hazard functions are often not known. This uncertainty should thus be included in the model's parameter values. Parameters estimated from data are often asymptotically normally distributed for a suitable parameterization. We therefore included parameter uncertainty in the model by sampling the parameters of the transition-specific hazard functions from a multivariate normal distribution. Therefore the transition-specific hazard functions need to be parameterized so that parameters are multivariate normally distributed.

For each simulated patient, all parameters are first drawn from the specified distribution. Then the simulation for this patient is performed, as described in Section 2.2. This procedure allows the direct inclusion of uncertainty in the estimated parameters into the model, and obtains confidence intervals in the statistical analyses of the hypothetical cohorts. These confidence intervals reflect both sampling and parameter uncertainty.

In order to include uncertainties in the transition probabilities, the $N$ simulations are split into $M$ groups. Then the above-mentioned proportions for each of these groups is calculated. Finally, the 2.5% and 97.5% quantiles are computed to get a 95% confidence interval for the transition probabilities at each time point. This procedure requires $N$ to be fairly large.

## 3. Using gems

In this section we illustrate how to use **gems** (Salazar Vizcaya *et al.* 2013). Figure 2 shows a flowchart of the steps to take to use **gems** and indicates where these steps are described in detail. First, Section 3.1 shows how to specify all the necessary input (number of states, hazard functions and parameters) to run a simulation. Then Section 3.2 shows how to simulate from this input. Section 3.3 describes how to include parameter covariances in the model and Section 3.4 shows how to add baseline covariates. Finally Section 3.5 describes how to include history dependence in the hazard functions and Section 3.6 describes an alternative to using hazard functions for specifying the transitions.

The package is available at CRAN and can be loaded by

```
R> require("gems")
```

The package **gems** uses three classes to encode all model inputs and outputs.

1. A `transition.structure` contains the number of model states and a `matrix` with elements that are used to specify transition-specific hazard functions, their parameters and covariances.

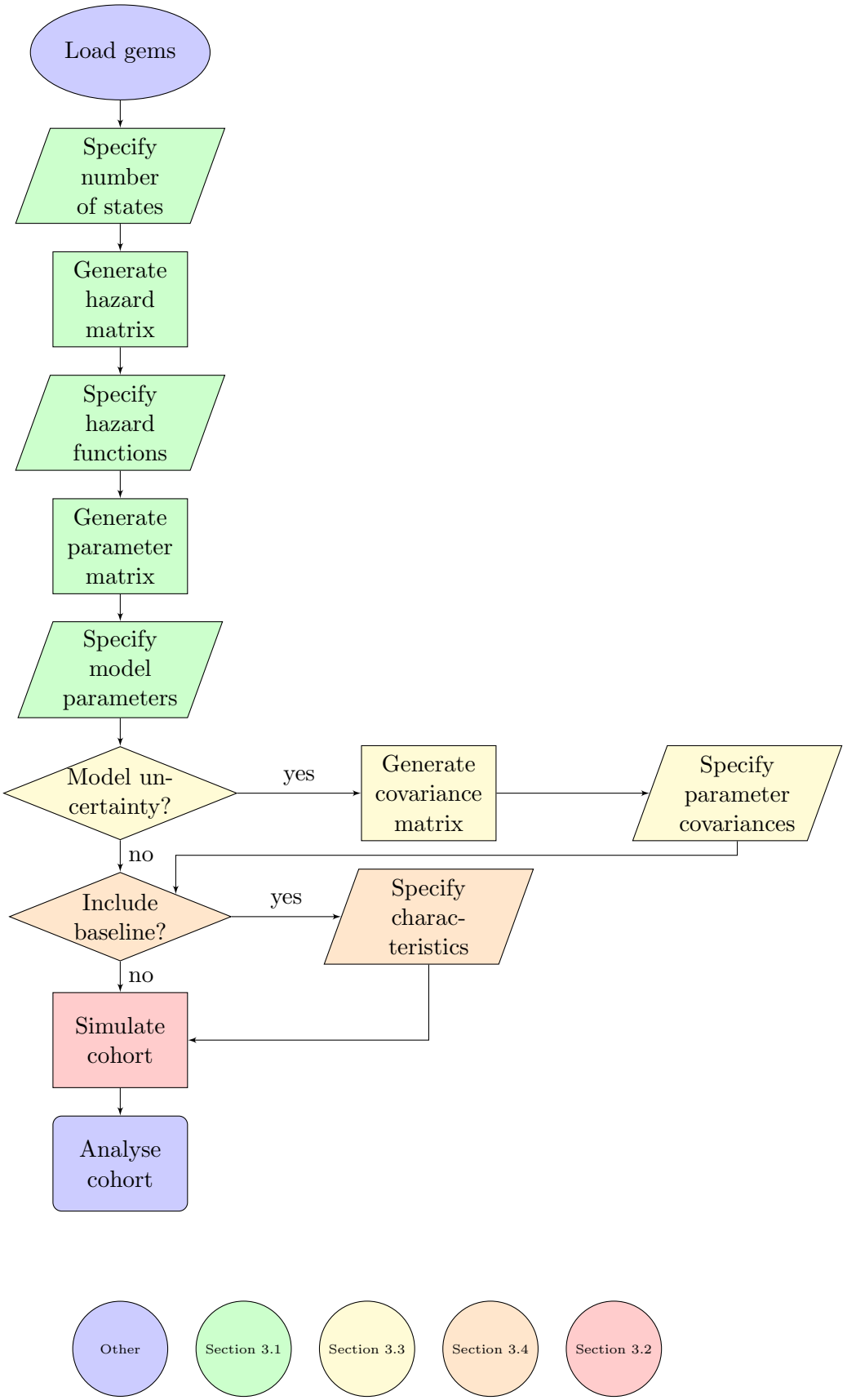


Figure 2: Flow chart indicating the steps users should take to simulate cohorts. The colors indicate where more detailed information is available. The parallelograms represent that user input is required, the rectangles represent that a process of **gems** performs this step and the diamonds represent user decisions.

2. An `ArtCohort` contains all aspects of the simulated cohort, including the model input and a `data.frame` with the entry times for each patient into each of the states.

3. The `PosteriorProbabilities` contain the transition probabilities or cumulative incidence that can be calculated from the `ArtCohort`.

The model has six main functions. The first three are used to specify the model, the fourth is used for simulation and the last two are used to summarize the results.

1. `generateHazardMatrix` creates a template of class `transition.structure` that can be used to specify the transition-specific hazard functions.

2. `generateParameterMatrix` creates a template of class `transition.structure` that can be used to specify the parameters.

3. `generateParameterCovarianceMatrix` creates a `transition.structure` that can be used to specify the parameter covariance.

4. `simulateCohort` simulates the specified artificial cohort and returns an object of class `ArtCohort`.

5. `transitionProbabilities` returns an object of class `posteriorProbabilities` that contains the transition probabilities from the initial state over time.

6. `cumulativeIncidence` returns an object of class `posteriorProbabilities` that contains the cumulative incidence over time.

### 3.1. Specifying the model

Suppose we want to simulate a disease with initial state $E_1$, intermediate state $E_2$ and absorbing state $E_3$ as depicted in the DAG in Figure 3. In order to fully specify the model,

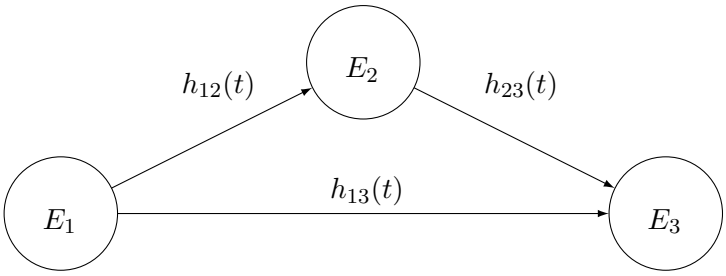


Figure 3: Model specification DAG.

the hazard functions, their parameters and the covariance structure of these parameters must be specified. The hazard functions are specified in a `transition.structure` of dimension $states \times states$.

The function `generateHazardMatrix` can be used to specify such a `transition.structure` that contains only the model structure.

```
R>   hf <- generateHazardMatrix(3)
R>   print(hf)

         to
from      State 1 State 2      State 3
  State 1 NULL    "impossible" "impossible"
  State 2 NULL    NULL         "impossible"
  State 3 NULL    NULL         NULL
```

The argument `statesNumber` specifies the number of states in the multistate model. The resulting `transition.structure` only provides the basic structure for how hazard functions are specified and the desired hazard functions must be entered.

For exponential, Weibull and Weibull mixture distributions, the built-in functions can be specified as `"Exponential"`, `"Weibull"`, and `"multWeibull"` respectively. Arbitrary continuous hazards can also be specified as functions.

For instance, assume that the transition times $T_{12}$ and $T_{13}$ are exponentially distributed and the transition time $T_{23}$ is Weibull distributed. Then the `transition.structure` can be set up using double square brackets as follows. To show different ways of specifying time-to-event distributions, we used a user-supplied function for $T_{12}$ and a built-in function for $T_{13}$, even though they are both exponentially distributed. For the first transition, the hazard function of an exponential is specified as a hazard function with its own parameterization. The parameterization of the built-in functions is explained below. Note that user-supplied functions need to return a result of the same length as the time argument $t$. The required code to specify the hazard functions described above is

```
R>   hf[[1, 2]] <- function(t, mu) rep(1 / mu, length(t))
R>   hf[[1, 3]] <- "Exponential"
R>   hf[[2, 3]] <- "Weibull"
R>   print(hf)

         to
from      State 1 State 2           State 3
  State 1 NULL    "function(t, mu)" "Exponential"
  State 2 NULL    NULL              "Weibull"
  State 3 NULL    NULL              NULL
```

When specifying a function as `"Exponential"` or `"Weibull"`, the parameterization is the same as in the `rexp` or `rweibull` function; that is,

$$h_{exp}(t, rate) = rate, \tag{12}$$

$$h_{Weibull}(t, shape, scale) = \frac{shape}{scale}\left(\frac{t}{scale}\right)^{shape-1}. \tag{13}$$

For the Weibull mixture model, the parameterization is

$$h_{multWeibull}(t, \boldsymbol{\omega}, \mathbf{k}, \boldsymbol{\lambda}) = \frac{\omega_1 f_W(t, k_1, \lambda_1) + \cdots + \omega_n f_W(t, k_n, \lambda_n)}{\omega_1(1 - F_W(t, k_1, \lambda_1)) + \cdots + \omega_n(1 - F_W(t, k_n, \lambda_n))}, \tag{14}$$

where $f_W$ and $F_W$ are the Weibull density and distribution functions in the same parameterization as before, where $k_i$ is the shape and $\lambda_i$ the scale of the $i$-th Weibull distribution. Here $\mathbf{k}$ and $\boldsymbol{\lambda}$ are $n$-dimensional vectors and $\boldsymbol{\omega}$ is an $(n-1)$ dimensional vector with $\omega_n = 1 - \sum_{i=1}^{n-1} \omega_i$ being defined automatically. Mixed Weibull distributions can be used when there are multiple modes of failure that result in the same state and can be estimated using maximum likelihood or non-linear least squares methods (Ling *et al.* 2009).

Specifying built-in functions is more efficient than specifying the hazard function of a Weibull distribution, because the simulation internally uses `rweibull` instead of using piecewise approximation of the Weibull hazard function and `rpexp` to generate Weibull distributed random numbers.

Once all hazard functions are suitably specified, the parameter values must be determined. The easiest way to do this is by using the function `generateParameterMatrix`, with the hazard structure `hf` as an argument,

```
R>   par <- generateParameterMatrix(hf)
R>   par[[1, 2]] <- list(mu = 3.1)
R>   par[[1, 3]] <- list(rate = 0.3)
R>   par[[2, 3]] <- list(shape = 3, scale = 3)
```

The `transition.structure` generated by `generateParameterMatrix` is again only a framework; the specific parameter values need to be assigned afterwards. Note that the parameters need to be specified in the order in which they appear in the function.

### 3.2. Simulation and post-processing

Once the model is specified, the simulation can be invoked with the `simulateCohort` function. The arguments for `simulateCohort` are the previously specified `transitionFunction`, `parameters`, as well as the number of patients `cohortSize` to be simulated and the final time `to` of the simulation.

```
R>   cohortSize <- 10000
R>   cohort <- simulateCohort(transitionFunctions = hf,
+                           parameters = par,
+                           cohortSize = cohortSize,
+                           to = 10)
R>   head(cohort)

          State 1   State 2   State 3
Patient 1       0        NA 2.8115242
Patient 2       0 4.3985589 8.7294035
Patient 3       0        NA 2.5803986
Patient 4       0        NA 0.2123987
Patient 5       0 0.5824707 4.5493082
Patient 6       0        NA 2.6388907
```

The output `ArtCohort` contains the entry time into the different states for each patient.

Next, we can calculate and plot the transition probabilities and cumulative incidence including 95% confidence intervals from the initial state using the functions `transitionProbabilities` and `cumulativeIncidence` respectively.

```
R>   post <- transitionProbabilities(cohort, times = seq(0,5, .1))
R>   cinc <- cumulativeIncidence(cohort, times = seq(0,5, .1))
R>   head(post)

         State 1 State 2 State 3
Time 0    1.0000  0.0000  0.0000
Time 0.1  0.9384  0.0344  0.0272
Time 0.2  0.8808  0.0658  0.0534
Time 0.3  0.8272  0.0933  0.0795
Time 0.4  0.7775  0.1187  0.1038
Time 0.5  0.7272  0.1464  0.1264

R>   head(cinc)

         State 1 State 2 State 3
Time 0         1  0.0000  0.0000
Time 0.1       1  0.0344  0.0272
Time 0.2       1  0.0658  0.0534
Time 0.3       1  0.0933  0.0795
Time 0.4       1  0.1188  0.1038
Time 0.5       1  0.1465  0.1264

R>   plot(post, main = "Transition probabilities", ci = TRUE)

R>   plot(cinc, main = "Cumulative incidence", ci = TRUE)
```

For the `transitionProbabilities` function, the argument `times` specifies the timepoints at which the transition probabilities should be returned. The `plot` function admits the argument `ci` in order to add confidence intervals to the figure. Figure 4 shows the transition probabilities and Figure 5 shows the cumulative incidence in the above example.

### 3.3. Parameter uncertainty

Suppose that we want to include parameter uncertainty in the above example. For instance, we estimate the shape and scale parameters for the transition from $E_2$ to $E_3$ be distributed as follows:

$$\begin{pmatrix} shape \\ scale \end{pmatrix} \sim \mathcal{MN}\left(\begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\right). \tag{15}$$

Then the covariance matrix can be specified using the `generateParameterCovarianceMatrix` function with the previously generated parameter `transition.structure` as an argument.

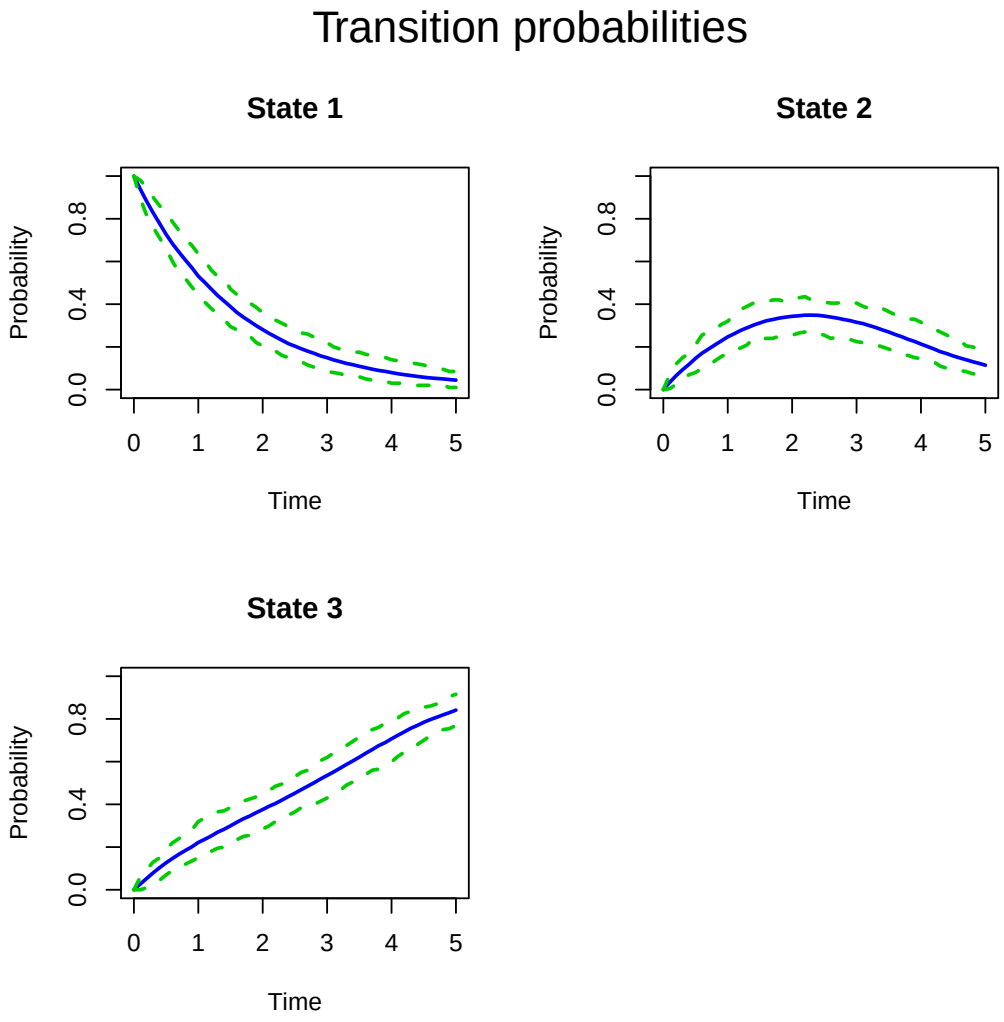


Figure 4: Transition probabilities.

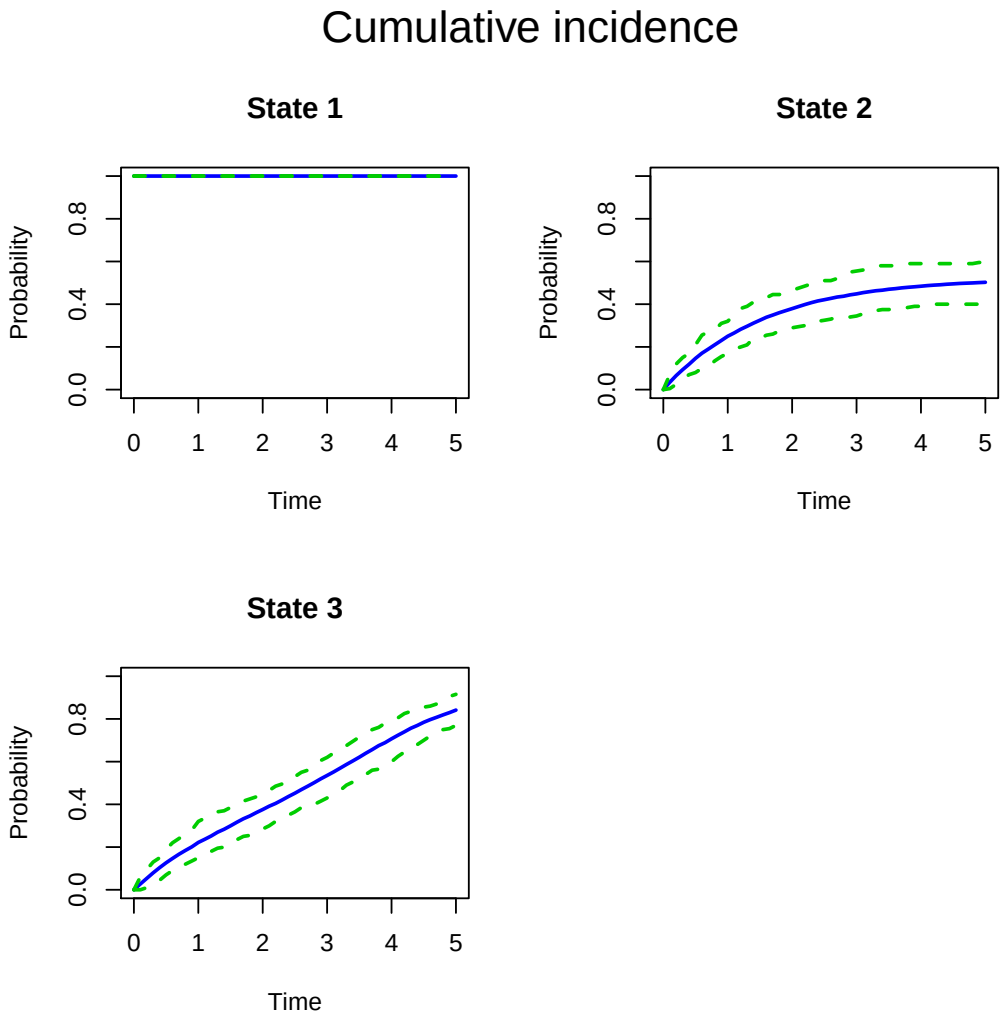


Figure 5: Cumulative incidence.

```
R>   cov <- generateParameterCovarianceMatrix(par)
R>   cov[[2, 3]] <- diag(.5, 2)
```

As with the parameter `transition.structure`, also the values of the parameter covariance `transition.structure` need to be specified after generating the `transition.structure`.

For the simulation, the uncertainty is included in the `parameterCovariance` argument for `simulateCohort` as follow:

```
R>   cohort <- simulateCohort(transitionFunctions = hf,
+                             parameters = par,
+                             parameterCovariances = cov,
+                             cohortSize = cohortSize,
+                             to = 10)
```

### 3.4. Baseline characteristics

Baseline characteristics can be included in the model by letting the hazard depend on the argument `baseline`; for example, if age and sex are important characteristics. Consider sex to be encoded as 0 for males and as 1 for females, and let the baseline age be the age in years. Suppose we want to simulate a cohort of 50% men and 50% women with ages distributed uniformly between 20 and 50. Baseline characteristics should be specified in a `matrix` or `data.frame` as follows.

```
R>   bl <- data.frame(sex = rbinom(cohortSize, 1, .5),
+                    age = runif(cohortSize, 20, 50))
R>   head(bl)
```

```
  sex      age
1   0 21.02408
2   1 44.70358
3   0 20.70234
4   1 36.94189
5   0 48.42227
6   1 29.78793
```

If there is a sex-specific rate, one option would be to record it as a `numeric` of `length` 2, with the first position describing the rate for male and the second position the rate for women. Suppose age is another risk factor, specified as a rate ratio per year. In this case the function would depend on the sex-specific rate `rate`, the rate ratio `rr` per year of age and the baseline characteristics `bl`. The model could then be specified as follows.

```
R>   hf[[1, 2]] <- function(t, bl, rate, rr) {
+     rep(rate[bl["sex"] + 1], length(t)) * rr ^ (bl["age"] - 20)
+   }
R>   par[[1, 2]] <- list(rate = c(0.2,0.3), rr = 1.02)
R>   cov[[1, 2]] <- diag(0, 3)
```

In order to simulate from this model that includes baseline characteristics, an additional argument `baseline` is added to the `simulateCohort` function as follows.

```
R>   cohort <- simulateCohort(transitionFunctions = hf,
+                             parameters = par,
+                             cohortSize = cohortSize,
+                             parameterCovariances = cov,
+                             baseline = bl,
+                             to = 5)
```

### 3.5. History dependence

In many real-world applications, transitions between states may depend both on the current state, and on the history of previous events in the patient history. For instance, in an HIV treatment model, the immune system worsens between failure of first-line therapy and start of second-line treatment and the mortality hazard after starting a second-line therapy depends on how long a person spent on failing first-line treatment (Gazzola *et al.* 2009).

History-dependence of the model can be specified by letting the hazard function depend on the argument `history`. This `history`-argument is a vector indexed by the transition-number

```
R>   gems:::auxcounter(3)
```

```
         to
from      State 1 State 2 State 3
  State 1       0       1       2
  State 2       0       0       3
  State 3       0       0       0
```

and is the transition time $T_i$ for the transitions that have occurred. For transitions that have not yet occurred, the `history` argument is 0.

The `history` argument allows to use the clock-forward approach (time refers to the time since the patient entered the initial state) instead of the clock-reset approach (time refers to time since entry into current state) to multistate modeling (Putter *et al.* 2007). To use the clock-forward approach, `t` can be replaced by `t + sum(history)`. Note that the clock-forward approach does not support built-in function (`"Exponential"`, `"Weibull"` and `"multWeibull"`) and users have to supply their own functions. If the transition-specific hazard for transition 3 was estimated using the clock-forward approach instead of the clock-reset approach, the Weibull function could be specified as follows.

```
R>   hf[[2, 3]] <- function(t, shape, scale, history) {
+     shape/scale * ((t + sum(history)) / scale) ^ (shape - 1)
+   }
```

The `simulateCohort` function can be used with this new function just as it was before.

### 3.6. Time to transition functions

Sometimes it is easier not to specify transitions via their hazards, but to directly specify the time it takes until the transition occurs. For instance, in some cases test results need to be confirmed by a second test three months later (e.g., HIV treatment failure tests (Estill *et al.* 2012)). Then the time would be three months and the hazard function would be a function with infinite point mass in 3. An additional argument `timeToTransition` to the `simulateCohort` function would have to be given by a `matrix`; the position of this kind of transition would be `TRUE` and the rest (usual hazard functions) would be `FALSE`. This procedure and the specification of the `transitionFunction` is as follows:

```
R>   hf[[1, 3]] <- function() 3
R>   par[[1, 3]] <- list()
R>   ttt <- matrix(FALSE, nrow = 3, ncol = 3)
R>   ttt[1, 3] <- TRUE
R>   cohort <- simulateCohort(transitionFunctions = hf,
+                           parameters = par,
+                           cohortSize = cohortSize,
+                           parameterCovariances = cov,
+                           timeToTransition = ttt,
+                           baseline = bl,
+                           to = 5)
```

# 4. Case study: Transcatheter aortic valve implantation

## 4.1. Introduction

Calcific aortic stenosis is a degenerative disease characterized by progressive narrowing of the aortic valve, which compromises oxygenated blood output from the heart. Medical therapy as a sole treatment option has not improved survival among patients with symptomatic severe aortic stenosis. Surgical aortic valve replacement (SAVR) is the treatment of choice and the gold standard for aortic valve disease treatment. In the presence of serious co-morbidities, and in patients considered to be at high-risk for SAVR, transcatheter aortic valve implantation (TAVI) techniques offer less-invasive treatment of valvular aortic stenosis. Older patients who have severe calcific aortic stenosis, characterized by the presence of co-morbidities and compromised left ventricular ejection fraction, have increased risk of complications from the surgical procedure itself. These high risk patients were managed medically until catheter-based treatment TAVI was introduced in 2002. During a TAVI implantation a bio-prosthetic valve is inserted and implanted within the diseased aortic valve through a catheter. The result of increased interest in this catheter-based approach is that this less invasive procedure is now used in patients with less severe disease (Pilgrim *et al.* 2012).

## 4.2. Statistical analysis

The `tavi` data set contains data on kidney injuries, bleeding complications and the combined endpoint of stroke or death for 194 patients. The variables `kidney, bleeding, death` are indicator variables that show if an event has occurred; the variables `kidney.dur, bleeding.dur, death.dur` are the times at which the events occurred or the patients were censored.

```
R>   data("tavi")
R>   head(tavi)
```

```
  id kidney kidney.dur bleeding bleeding.dur death death.dur
1 P1      0        779        0          779     0       779
2 P2      0          8        1            8     0       379
3 P3      0        342        0          342     0       342
4 P4      1          3        0            3     0        36
5 P5      1          7        0            7     1       131
6 P6      0          9        1            9     0       154
```

In the following discussion, the DAG depicted in Figure 6 is assumed. Since no patients experience both kidney injury and bleeding complications, we assume these events to be mutually exclusive.

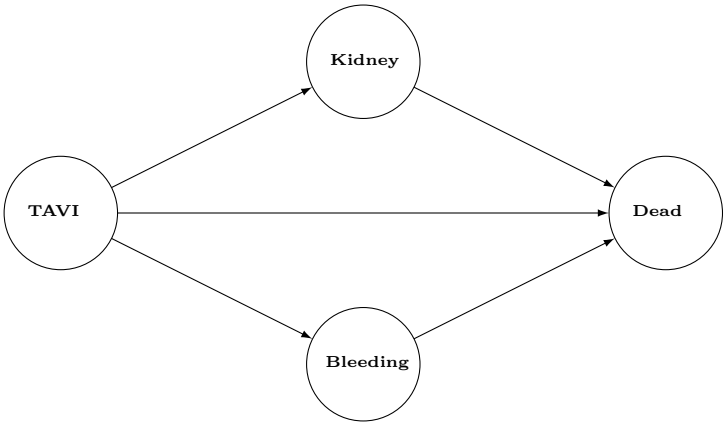


Figure 6: DAG for case study on TAVI.

We then create the transition matrix using the **mstate** package. According to the DAG the transition matrix is given by

```
R>   tmat <- transMat(x = list(c(2, 3, 4), c(4), c(4), c()),
+                   names = c("TAVI",
+                             "Kidney Injury",
+                             "Bleeding",
+                             "Stroke/Death"))
R>   tmat
```

```
              to
from           TAVI Kidney Injury Bleeding Stroke/Death
  TAVI           NA             1        2           3
  Kidney Injury  NA            NA       NA           4
  Bleeding       NA            NA       NA           5
  Stroke/Death   NA            NA       NA          NA
```

In order to estimate the transition-specific hazard functions, we prepare the data using the **mstate** package. We use `msprep` to get the data into long format, and `split` to split the data according to the transition.

```
R>   mstavi <- msprep(data = tavi, trans = tmat,
+                   time = c(NA, "kidney.dur", "bleeding.dur", "death.dur"),
+                   status = c(NA, "kidney", "bleeding",  "death"))
R>   head(mstavi)

An object of class 'msdata'

Data:
  id from to trans Tstart Tstop time status
1  1    1  2     1      0   779  779      0
2  1    1  3     2      0   779  779      0
3  1    1  4     3      0   779  779      0
4  2    1  2     1      0     8    8      0
5  2    1  3     2      0     8    8      1
6  2    1  4     3      0     8    8      0

R>   mstavi$time[mstavi$time == 0] <- .Machine$double.eps
R>   msplit <- split(mstavi, mstavi$trans)
R>   head(msplit[[5]])

An object of class 'msdata'

Data:
    id from to trans Tstart Tstop time status
7    2    3  4     5      8   379  371      0
22   6    3  4     5      9   154  145      0
44  13    3  4     5      7     8    1      1
84  25    3  4     5     10   154  144      0
99  29    3  4     5      6   847  841      1
103 30    3  4     5      3     8    5      1
```

As a first step we fit an exponential distribution to all transition times. For each transition, we estimate the rate and the variance.

```
R>   exp.fit <- sapply(msplit, function(x)
+     summary(survreg(Surv(time, status) ~ 1,
+                     data = x,
+                     dist = "exponential")))
R>   exp.coef <- unlist(exp.fit["coefficients", ])
R>   exp.var <- unlist(exp.fit["var", ])
```

Next we specify the model, simulate from it and compare the simulated mortality to a Kaplan-Meier graph of mortality.

```
R>   states <- 4
R>   maxtime <- max(mstavi$time)
R>   ind <- which(!is.na(tmat), arr.ind = TRUE)
R>   hm <- generateHazardMatrix(states)
R>   for (i in 1:dim(ind)[1]){
+     hm[[ind[i, 1], ind[i, 2]]] <- "Weibull"
+   }
R>   par <- generateParameterMatrix(hm)
R>   for (i in 1:dim(ind)[1]){
+     par[[ind[i, 1], ind[i, 2]]] <- list(shape = 1,
+                                         scale = exp(exp.coef[i]))
+   }
R>   cov <- generateParameterCovarianceMatrix(par)
R>   for (i in 1:dim(ind)[1]){
+     cov[[ind[i, 1], ind[i, 2]]] <- matrix(c(0, 0, 0, exp.var[i]), nrow=2)
+   }
R>   ds <- simulateCohort(transitionFunctions = hm,
+                        parameters = par,
+                        cohortSize = 100 * nrow(tavi),
+                        parameterCovariances = cov,
+                        to = maxtime)
R>   cinc <- cumulativeIncidence(ds, 0:maxtime, colnames(tmat), M = 100)
```

Figure 7 shows the overall mortality from the simulated cohort. Because the purpose of this study is to illustrate the use and flexibility of the package, we split time into monthly intervals and calculate piecewise constant hazard functions using the `pehaz` function from the **muhaz** package.

```
R>   timeStep <- 30
R>   pwexp <- sapply(msplit, function(x) pehaz(x$time,
+                                             x$status,
+                                             width = timeStep,
+                                             min.time = 0,
+                                             max.time = max(mstavi$time)))
R>   cuts <- pwexp["Cuts", ]
R>   pwhazard <- pwexp["Hazard", ]
```

We parameterize the hazard functions with piecewise constant hazards and simulate again.

```
R>   hm2 <- generateHazardMatrix(states)
R>   for (i in 1:dim(ind)[1]){
+     hm2[[ind[i, 1], ind[i, 2]]] <- function(t, rates) {
+       rates[t / timeStep + 1]
+     }
+   }
R>   par2 <- generateParameterMatrix(hm2)
R>   for (i in 1:dim(ind)[1]){
```

```
+     par2[[ind[i, 1], ind[i, 2]]] <- list(rates = pwhazard[[i]])
+   }
R>   ds2 <- simulateCohort(transitionFunctions = hm2,
+                         parameters = par2,
+                         cohortSize = 100 * nrow(tavi),
+                         to = maxtime)
R>   cinc2 <- cumulativeIncidence(ds2, 0:maxtime, colnames(tmat), M = 100)
```

The plot function also admits an argument `states`, which can be used in order to only `plot` certain states as shown in the following example.

```
R>   plot(cinc, states = 4, axes = FALSE, frame = TRUE, col = 2,
+        xlab = "Time (in months)", main = "Mortality", ci = TRUE)
R>   lines(survfit(Surv(death.dur, death) ~ 1, data = tavi),
+         fun = "event", lwd = 2)
R>   lines(survfit( Surv(death.dur, death)~1, data=tavi),
+         fun="event", lwd=2, conf.int=TRUE, lty=2)
R>   par(new = TRUE)
R>   plot(cinc2, states = 4, axes = FALSE, frame = TRUE, col = 3,
+        xlab = "", main = "", ci = TRUE)
R>   axis(2); axis(4)
R>   axis(1, at = (0:13*90)[0:6*2+1], labels = (0:13*3)[0:6*2+1])
R>   legend(200, .8, c("Data",
+                    "Simulation: exponential",
+                    "Simulation: piecewise exponential"),
+          lty = 1, col = c(1:3), lwd = 2)
```

Figure 7 shows how the simulated cumulative incidence depends on the statistical model. The package **gems** admits the choice of any transition-specific hazard function. We will now use the second model with piecewise constant hazard functions to estimate the effect of an intervention on mortality.

### 4.3. Intervention modeling

Suppose there is a new intervention that dramatically reduces the probability of getting bleeding complications, and we are interested in the impact of this intervention on mortality. For simplicity, we assume that the intervention reduces the transition-specific hazard of bleeding complications by 80%. Then

```
R>   hm3 <- hm2
R>   par3 <- par2
R>   par3[[1, 3]]$rates <- par3[[1, 3]]$rates / 5
R>   ds3 <- simulateCohort(transitionFunctions = hm3,
+                         parameters = par3,
+                         cohortSize = 100 * nrow(tavi),
+                         to = maxtime)
R>   cinc3 <- cumulativeIncidence(ds3, 0:maxtime, colnames(tmat), M = 100)
```

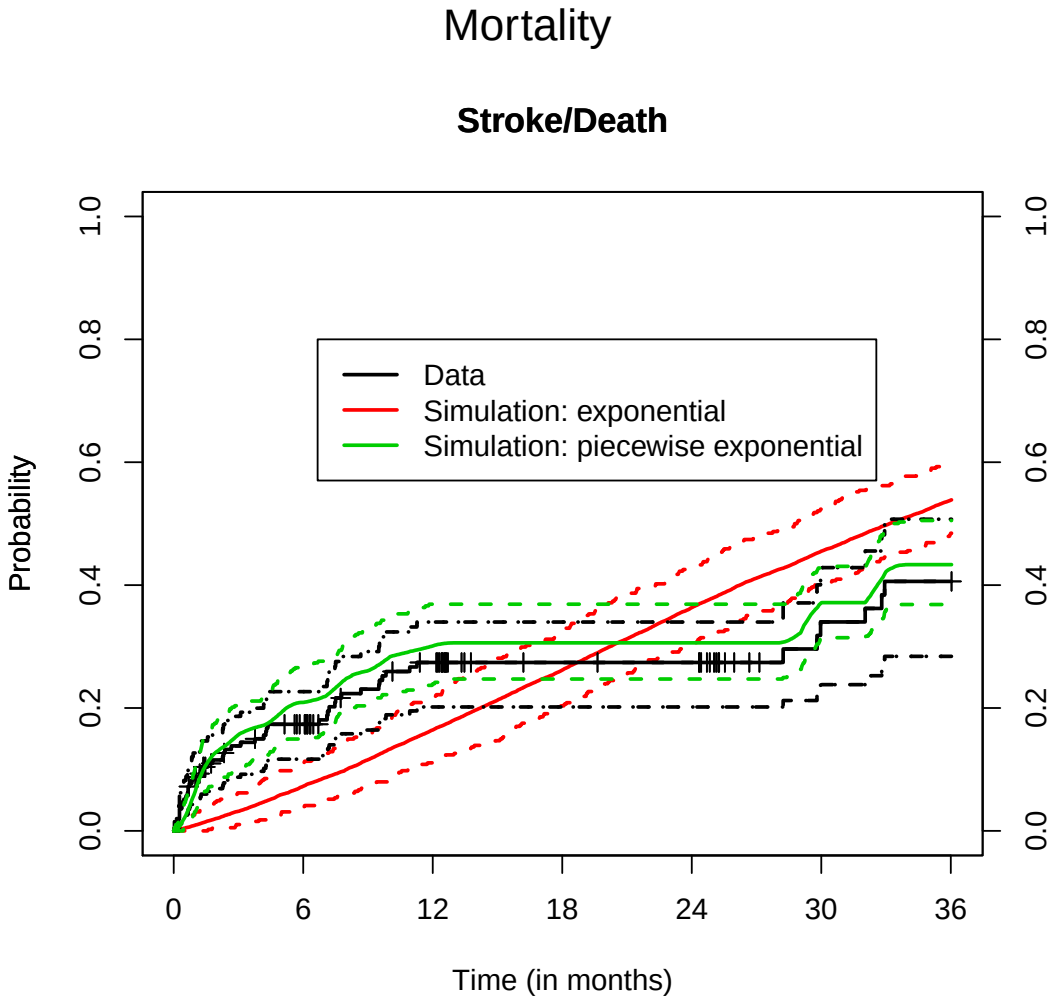


Figure 7: Cumulative incidence with constant and piecewise constant transition-specific hazard functions.

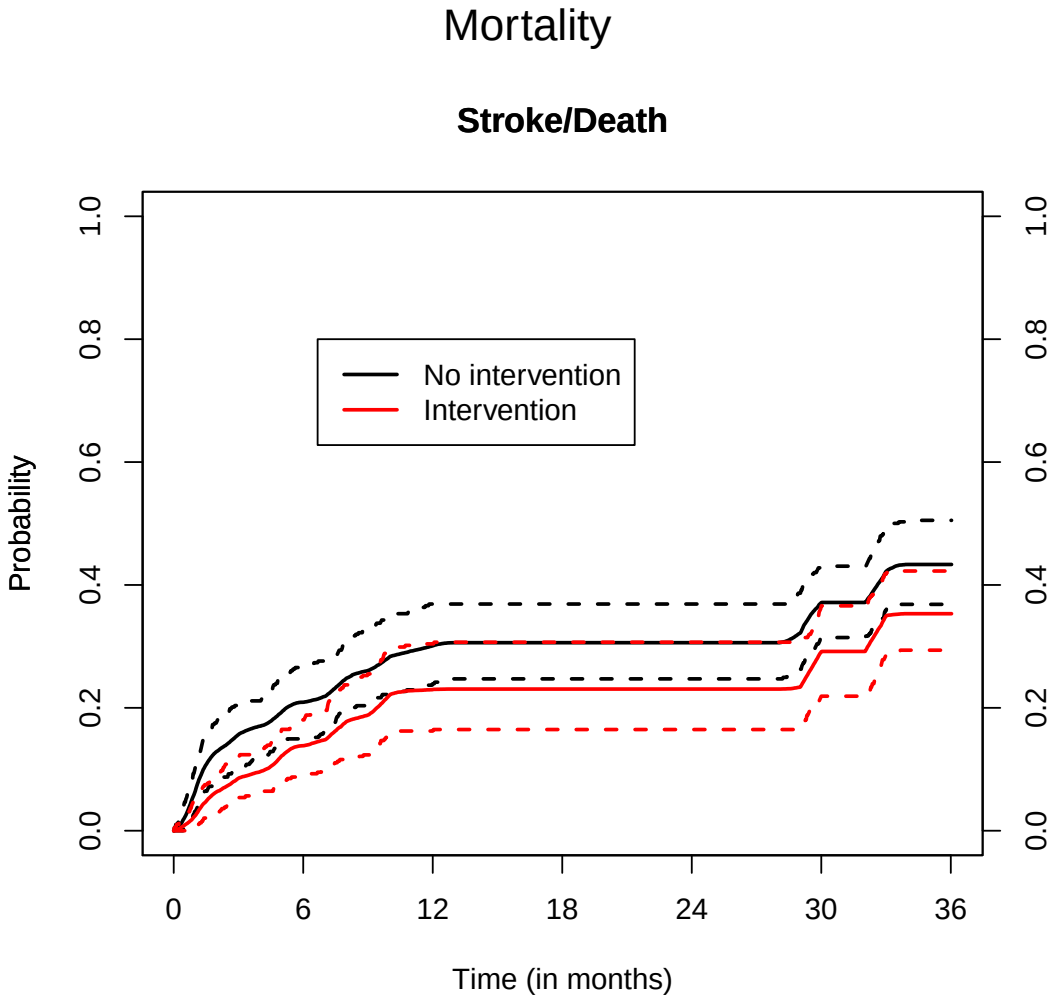


Figure 8: Effect of reducing bleeding complications on mortality.

```
R>   plot(cinc2, states = 4, axes = FALSE, frame = TRUE, col = 1, ci = TRUE,
+        xlab = "Time (in months)", main = "Mortality")
R>   par(new = TRUE)
R>   plot(cinc3, states = 4, axes = FALSE, frame = TRUE, col = 2, ci = TRUE,
+        xlab = "", main = "")
R>   axis(2); axis(4)
R>   axis(1, at = (0:13 * 90)[0:6*2 + 1], labels = (0:13 * 3)[0:6 * 2 + 1])
R>   legend(200, .8, c("No intervention", "Intervention"),
+          lty = 1, col = 1:2, lwd = 2)
```

Figure 8 shows that reducing bleeding complications by 80% decreases three-year mortality by 18.5% from 43.3% to 35.3%.

## 5. Conclusion

In this paper we have presented the R package **gems**, which allows simulation from a directed acyclic multistate model. The R package **gems** is a flexible tool for investigating and evaluating health interventions. We have given detailed examples of the use of each function of **gems**, and an example of its use to evaluate the effect of reduced bleeding complications in TAVI patients on mortality.

Several packages estimate and simulate Markov models, but we are not aware of any other packages that allow simulation from a multistate model with arbitrary transition-specific hazard functions. This flexibility in hazard functions improves its fit to data and allows to more accurately estimate the effects of different interventions. This package's inclusion of history-dependent transitions is also a major improvement on many traditional model structures.

The **gems** package has some limitations, including the fact that a DAG is required. Sometimes it is useful to have models in which patients can return to a previous state. If this feature is not frequently required, the problem can be resolved by repeating the state in a DAG. Otherwise a different model structure is needed. A further limitation is that higher flexibility requires more intensive computation, compared to traditional models. Longitudinal processes could be incorporated in a joint model, and evaluations of the artificial cohorts could be further automated in future expansions.

The **gems** package has useful functions for simulating hypothetical cohorts of patients based on a multistate model with general transition-specific hazard functions, and is a flexible and user-friendly tool for planning and evaluating public-health interventions.

## Acknowledgments

We would like to thank the Swiss Cardiovascular Center Bern at the Bern University Hospital (S. Windecker, B. Meier, P. Wenaweser, T. Pilgrim, S. Stortecky, L. Buellesfeld) for providing data on transcatheter aortic valve implantation for the case study. We also thank K. Tal for her editorial assistance. This study was supported by UNITAID; a PROSPER fellowship to O. Keiser and a PRODOC PhD grant for N. Blaser from the Swiss National Science Foundation.

## References

Anderson RM, May RM (1992). *Infectious Diseases of Humans Dynamics and Control*. Oxford University Press. URL http://www.oup.com/uk/catalogue/?ci=9780198540403.

Blaser N, Salazar Vizcaya L, Estill J, Zahnd C, Kalesan B, Egger M, Keiser O, Gsponer T (2015). "gems: An R Package for Simulating from Disease Progression Models." *Journal of Statistical Software*, **64**(10), 1–22. URL http://www.jstatsoft.org/v64/i10/.

de Wreede LC, Fiocco M, Putter H (2011). ***mstate**: An R Package for the Analysis of Competing Risks and Multi-State Models*. URL http://www.jstatsoft.org/v38/i07/.

Estill J, Aubrière C, Egger M, Johnson L, Wood R, Garone D, Gsponer T, Wandeler G, Boulle A, Davies MA, Hallett TB, Keiser O (2012). "Viral Load Monitoring of Antiretroviral Therapy, Cohort Viral Load and HIV Transmission in Southern Africa: a Mathematical Modelling Analysis." *AIDS*, **26**(11), 1403–13.

Estill J, Tweya H, Egger M, Wandeler G, Feldacker C, Johnson LF, Blaser N, Vizcaya LS, Phiri S, Keiser O (2014). "Tracing of Patients Lost to Follow-up and HIV Transmission: Mathematical Modeling Study Based on 2 Large ART Programs in Malawi." *J. Acquir. Immune Defic. Syndr.*, **65**(5), e179–186. ISSN 1944-7884. doi:10.1097/QAI.0000000000000075.

Garnett GP, Cousens S, Hallett TB, Steketee R, Walker N (2011). "Mathematical Models in the Evaluation of Health Programmes." *Lancet*, **378**(9790), 515–25.

Gazzola L, Tincati C, Bellistri GM, Monforte A, Marchetti G (2009). "The Absence of CD4+ T Cell Count Recovery Despite Receipt of Virologically Suppressive Highly Active Antiretroviral Therapy: Clinical Risk, Immunological Gaps, and Therapeutic Options." *Clin Infect Dis*, **48**(3), 328–37.

Gillespie DT (1977). "Exact Stochastic Simulation of Coupled Chemical Reactions." *J. Phys. Chem.*, **81**(25), 2340–2361. URL http://dx.doi.org/10.1021/j100540a008.

Hess K, Gentleman R (2010). ***muhaz**: Hazard Function Estimation in Survival Analysis*. R package version 1.2.5, URL http://CRAN.R-project.org/package=muhaz.

Jackson CH (2011). "Multi-State Models for Panel Data: The **msm** Package for R." *Journal of Statistical Software*, **38**(8), 1–29. URL http://www.jstatsoft.org/v38/i08/.

Ling D, Huang HZ, Liu Y (2009). "A method for parameter estimation of Mixed Weibull distribution." In *Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual*, pp. 129–133. ISSN 0149-144X. doi:10.1109/RAMS.2009.4914663.

Lloyd AL (2001). "Realistic Distributions of Infectious Periods in Epidemic Models: Changing Patterns of Persistence and Dynamics." *Theor Popul Biol*, **60**(1), 59–71.

Pearl J (2009). *Causality*. Second edition. Cambridge University Press, Cambridge. ISBN 978-0-521-89560-6; 0-521-77362-8.

Phillips AN, Pillay D, Garnett G, Bennett D, Vitoria M, Cambiano V, Lundgren J (2011). "Effect on Transmission of HIV-1 Resistance of Timing of Implementation of Viral Load Monitoring to Determine Switches from First to Second-line Antiretroviral Regimens in Resource-limited Settings." *AIDS*, **25**(6), 843–50.

Pilgrim T, Kalesan B, Wenaweser P, Huber C, Stortecky S, Buellesfeld L, Khattab AA, Eberle B, Gloekler S, Gsponer T, Meier B, Juni P, Carrel T, Windecker S (2012). "Predictors of Clinical Outcomes in Patients With Severe Aortic Stenosis Undergoing TAVI: A Multistate Analysis." *Circ Cardiovasc Interv*, **5**(6), 856–61.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in Biostatistics: Competing Risks and Multi-state Models." *Stat. Med.*, **26**(11), 2389–2430. ISSN 0277-6715. URL http://dx.doi.org/10.1002/sim.2712.

Salazar Vizcaya L, Blaser N, Gsponer T (2013). ***gems**: General Multistate Simulation Model*. R package version 0.9.1, URL http://CRAN.R-project.org/package=gems.

Therneau TM (2014). *A Package for Survival Analysis in S*. R package version 2.37-7, URL http://CRAN.R-project.org/package=survival.

## Affiliation:

Nello Blaser
Institute of Social and Preventive Medicine
University of Bern
3012 Bern, Switzerland
E-mail: nblaser@ispm.unibe.ch

Luisa Salazar Vizcaya
Institute of Social and Preventive Medicine
University of Bern
3012 Bern, Switzerland
E-mail: lsalazar@ispm.unibe.ch