Flexible parametric multi-state modelling with flexsurv

Christopher H. Jackson
MRC Biostatistics Unit, Cambridge, UK

Abstract

A *multi-state model* represents how an individual moves between multiple states in continuous time. Survival analysis is a special case with two states, "alive" and "dead". *Competing risks* are a further special case, where there are multiple causes of death, that is, one starting state and multiple possible destination states.

This vignette describes the two forms of multi-state or competing risks models that can be implemented in **flexsurv**. Section 2 describes models based on *cause-specific hazards*. Section 3 describes a less commonly-used approach based on *mixture models*. Cause-specific hazards models tend to be faster to fit, whereas the parameters of the mixture models are easier to interpret.

Keywords: multi-state models, multistate models, competing risks.

1. Overview

This vignette describes multi-state models for data where there are a series of event times $t_1, \ldots, t_n$ for an individual, corresponding to changes of state. The last of these may be an observed or right-censored event time. Note *panel data* are not considered here — that is, observations of the state of the process at an arbitrary set of times (Kalbfleisch and Lawless 1985). In panel data, we do not necessarily know the time of each transition, or even whether transitions of a certain type have occurred at all between a pair of observations. Multi-state models for that type of data (and also exact event times) can be fitted with the **msm** package for R (Jackson 2011), but are restricted to (piecewise) exponential event time distributions. Knowing the exact event times enables much more flexible models, which **flexsurv** can fit.

The **flexsurv** package provides two general frameworks for multi-state modelling.

1. The most common framework is based on *cause-specific hazards* of competing risks. This is an extension of standard survival modelling, and is explained in Section 2.

2. An alternative approach is based on *mixture models*. For example, suppose there are three competing states that an individual in state r may move to next. People are then classified into three *mixture components*, defined by the state the person moves to. The probabilities of each next state, and the distribution of the time of moving to each state, are estimated jointly. These quantities are easier to interpret than cause specific hazards. This approach was originally described by Larson and Dinse (1985). In Section 3 we explain how to implement it using the `flexsurvmix` function.

2. Multi-state modelling using cause-specific hazards

Given that an individual is in state $X(t)$ at time t, their next state, and the time of the change, are governed by a set of *transition intensities* or *transition hazards*

$$q_{rs}(t, \mathbf{z}(t), \mathcal{F}_t) = \lim_{\delta t \to 0} \mathsf{P}(X(t + \delta t) = s | X(t) = r, \mathbf{z}(t), \mathcal{F}_t)/\delta t$$

for states $r, s = 1, \ldots, R$, which for a survival model are equivalent to the hazard $h(t)$. The intensity represents the instantaneous risk of moving from state r to state s, and is zero if the transition is impossible. It may depend on covariates $\mathbf{z}(t)$, the time t itself, and possibly also the "history" of the process up to that time, $\mathcal{F}_t$: the states previously visited or the length of time spent in them.

Alternative time scales In semi-Markov ("clock-reset") models, $q_{rs}(t)$ is defined as a function of the time t since entry into the current state. Any software to fit survival models can also fit this kind of multi-state model, as the following sections will explain.

In an inhomogeneous Markov model, t represents the time since the beginning of the process (that is, a "clock-forward" scale is used), but the intensity $q_{rs}(t)$ does not depend further on $\mathcal{F}_t$. Again, standard survival modelling software can be used, with the additional requirement that it can deal with left-truncation or *counting process* data, which `survreg`, for example, does not currently support.

These approaches are equivalent for competing risks models, since there is at most one transition for each individual, so that the time since the beginning of the process equals the time spent in the current state. Therefore no left-truncation is necessary.

Note also that in a clock-reset model, the time since the beginning of the process may enter the model as a covariate. Likewise, in a clock-forward model, the time spent in the current state may enter as a covariate, in which case the model is no longer Markov.

Example For illustration, consider a simple three-state example, previously studied by Heng *et al.* (1998). Recipients of lung transplants are are risk of bronchiolitis obliterans syndrome (BOS). This was defined as a decrease in lung function to below 80% of a baseline value defined in the six months following transplant. A three-state "illness-death" model represents the risk of developing BOS, the risk of dying before developing BOS, and the risk of death after BOS. BOS is assumed to be irreversible, so there are only three allowed transitions (Figure 1), each with an intensity function $q_{rs}(t)$.

2.1. Representing multi-state data as survival data

Andersen and Keiding (2002) and Putter *et al.* (2007) explain how to implement multi-state models by manipulating the data into a suitable form for survival modelling software — an overview is given here. For each permitted $r \to s$ transition in the multi-state model, there is a corresponding "survival" (time-to-event) model, with hazard rates defined by $q_{rs}(t)$. For a patient who enters state r at time t_j, their next event at t_{j+1} is defined by the model structure to be one of a set of competing events $s_1, \ldots, s_{n_r}$. This implies there are n_r corresponding survival models for this state r, and $\sum_r n_r$ models over all states r. In the BOS example, there are $n_1 = 2, n_2 = 1$ and $n_3 = 0$ possible transitions from states 1, 2 and 3 respectively.

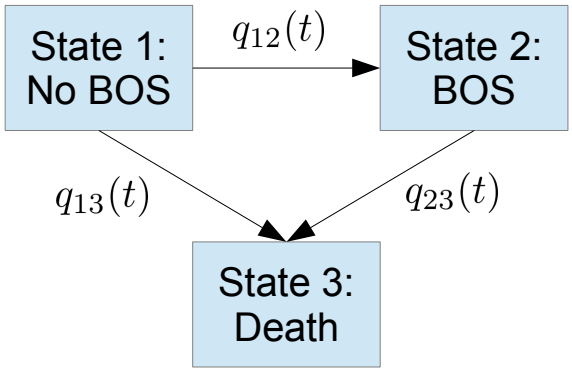


Figure 1: Three-state multi-state model for bronchiolitis obliterans syndrome (BOS).

The data to inform the n_r models from state r consists firstly of an indicator for whether the transition to the corresponding state $s_1, \ldots, s_{n_r}$ is observed or censored at t_{j+1}. If the individual moves to state s_k, the transitions to all other states in this set are censored at this time. This indicator is coupled with:

- (for a semi-Markov model) the time elapsed $dt_j = t_{j+1} - t_j$ from state r entry to state s entry. The "survival" model for the $r \to s$ transition is fitted to this time.
- (for an inhomogeneous Markov model) the start and stop time (t_j, t_{j+1}), as in §**??**. The $r \to s$ model is fitted to the right-censored time t_{j+1} from the *start of the process*, but is conditional on not experiencing the $r \to s$ transition until after the state r entry time. In other words, the $r \to s$ transition model is *left-truncated* at the state r entry time.

In this form, the outcomes of two patients in the BOS data are

```
R> library(flexsurv)
R> bosms3[18:22, ]

An object of class 'msdata'

Data:
   id from to    Tstart     Tstop     years status trans
18  7    1  2 0.0000000 0.1697467 0.1697467      1     1
19  7    1  3 0.0000000 0.1697467 0.1697467      0     2
20  7    2  3 0.1697467 0.6297057 0.4599589      1     3
21  8    1  2 0.0000000 8.1615332 8.1615332      0     1
22  8    1  3 0.0000000 8.1615332 8.1615332      1     2
```

Each row represents an observed (`status = 1`) or censored (`status = 0`) transition time for one of three time-to-event models indicated by the categorical variable `trans` (defined as a factor). Times are expressed in years, with the baseline time 0 representing six months after transplant. Values of `trans` of 1, 2, 3 correspond to no BOS→BOS, no BOS→death and BOS→death respectively. The first row indicates that the patient (`id` 7) moved from state 1 (no BOS) to state 2 (BOS) at 0.17 years, but (second row) this is also interpreted as a

censored time of moving from state 1 to state 3, potential death before BOS onset. This patient then died, given by the third row with `status` 1 for `trans` 3. Patient 8 died before BOS onset, therefore at 8.2 years their potential BOS onset is censored (fourth row), but their death before BOS is observed (fifth row).

The **mstate** R package (de Wreede *et al.* 2010, 2011) has a utility `msprep` to produce data of this form from "wide-format" datasets where rows represent individuals, and times of different events appear in different columns. **msm** has a similar utility `msm2Surv` for producing the required form given longitudinal data where rows represent state observations.

2.2. Multi-state model likelihood with cause-specific hazards

After forming survival data as described above, a multi-state model can be fitted by maximising the standard survival model likelihood (given at the start of the main **flexsurv** vignette), $l(\boldsymbol{\theta}|\mathbf{x}) = \prod_i l_i(\boldsymbol{\theta}|x_i)$, where $\mathbf{x}$ is the data, and i now indexes multiple observations for multiple individuals. This can also be written as a product over the $K = \sum_r n_r$ transitions k, and the m_k observations j pertaining to the kth transition. The transition type will typically enter this model as a categorical covariate — see the examples in the next section.

$$l(\boldsymbol{\theta}|\mathbf{x}) = \prod_{k=1}^{K} \prod_{j=1}^{m_k} l_{jk}(\boldsymbol{\theta}|\mathbf{x}_{jk}) \tag{1}$$

Therefore if the parameter vector $\boldsymbol{\theta}$ can be partitioned as $(\boldsymbol{\theta}_1|\ldots|\boldsymbol{\theta}_K)$, independent components for each transition k, the likelihood becomes the product of K independent transition-specific likelihoods (Andersen and Keiding 2002). The full multi-state model can then be fitted by maximising each of these independently, using K separate calls to a survival modelling function such as `flexsurvreg`. This can give vast computational savings over maximising the joint likelihood for $\boldsymbol{\theta}$ with a single fit. For example, Ieva *et al.* (2015) used **flexsurv** to fit a parametric multi-state model with 21 transitions and 84 parameters for over 30,000 observations, which was computationally impractical via the joint likelihood, whereas it only took about a minute to perform 21 transition-specific fits.

On the other hand, if any parameters are constrained between transitions (e.g. if hazards are proportional between transitions, or the effects of covariates on different transitions are the same) then it is necessary to maximise the joint likelihood (1) with a single call.

2.3. Fitting parametric multi-state models with cause-specific hazards

Joint likelihood Three multi-state models are fitted to the BOS data using `flexsurvreg`, firstly using a single likelihood maximisation for each model. The first two use the "clock-reset" time scale. `crexp` is a simple time-homogeneous Markov model where all transition intensities are constant through time, so that the clock-forward and clock-reset scales are identical. The time to the next event is exponentially-distributed, but with a different rate q_{rs} for each transition type `trans`. `crwei` is a semi-Markov model where the times to BOS onset, death without BOS and the time from BOS onset to death all have Weibull distributions, with a different shape and scale for each transition type. `cfwei` is a clock-forward, inhomogeneous Markov version of the Weibull model: the 1→2 and 1→3 transition models are the same, but

the third has a different interpretation, now the time *from baseline* to death with BOS has a Weibull distribution.

```
R> crexp <- flexsurvreg(Surv(years, status) ~ trans, data = bosms3,
+                       dist = "exp")
R> crwei <- flexsurvreg(Surv(years, status) ~ trans + shape(trans),
+                       data = bosms3, dist = "weibull")
R> cfwei <- flexsurvreg(Surv(Tstart, Tstop, status) ~ trans + shape(trans),
+                       data = bosms3, dist = "weibull")
```

Semi-parametric equivalents The equivalent Cox models are also fitted using `coxph` from the **survival** package. These specify a different baseline hazard for each transition type through a function `strata` in the formula, so since there are no other covariates, they are essentially non-parametric. Note that the `strata` function is not currently understood by `flexsurvreg` — the user must say explicitly what parameters, if any, vary with the transition type, as in `crwei`.

```
R> crcox <- coxph(Surv(years, status) ~ strata(trans), data = bosms3)
R> cfcox <- coxph(Surv(Tstart, Tstop, status) ~ strata(trans), data = bosms3)
```

In all cases, if there were other covariates, they could simply be included in the model formula. Typically, covariate effects will vary with the transition type, so that an interaction term with `trans` would be included. Some post-processing might then be needed to combine the main covariate effects and interaction terms into an easily-interpretable quantity (such as the hazard ratio for the r, s transition). Alternatively, **mstate** has a utility `expand.covs` to expand a single covariate in the data into a set of transition-specific covariates, to aid interpretation (see de Wreede *et al.* 2011).

Transition-specific models In this small example, the joint likelihood can be maximised easily with a single function call, but for larger models and datasets, this may be unfeasible. A more computationally-efficient approach is to fit a list of transition-specific models, as follows.

```
R> mod_nobos_bos <- flexsurvreg(Surv(years, status) ~ 1, subset=(trans==1),
+                               data = bosms3, dist = "weibull")
R> mod_nobos_death <- flexsurvreg(Surv(years, status) ~ 1, subset=(trans==2),
+                                 data = bosms3, dist = "weibull")
R> mod_bos_death <- flexsurvreg(Surv(years, status) ~ 1, subset=(trans==3),
+                               data = bosms3, dist = "weibull")
```

We then define a matrix `tmat` describes the transition structure of the multi-state model , as a matrix of integers whose r, s entry is i if the ith transition type is r, s, and has `NA`s on the diagonal and where the r, s transition is disallowed.

```
R> tmat <- rbind(c(NA, 1, 2), c(NA, NA, 3), c(NA, NA, NA))
R> crfs <- fmsm(mod_nobos_bos, mod_nobos_death, mod_bos_death, trans = tmat)
```

The three transition models are then grouped together, and the transition matrix is attached, using the `fmsm` function. This constructs an R object, `crfs`, that fully describes the multistate model.

The `fmsm` object can be supplied to the output and prediction functions described in the subsequent sections, instead of a single `flexsurvreg` object. However, this approach is not possible if there are constraints in the parameters across transitions, such as common covariate effects.

Any parametric distribution can be employed in a multi-state model, just as for standard survival models, with `flexsurvreg`. Spline models may also be fitted with `flexsurvspline`, and if hazards are assumed proportional, they are expected to give similar results to the Cox model. Since **flexsurv** version 2.0, different parametric families can be used for different transitions, though in earlier versions, the same family had to be used.

2.4. Obtaining cumulative transition-specific hazards

Multi-state models can be characterised by their cumulative $r \to s$ transition-specific hazard functions $H_{rs}(t) = \int_0^t q_{rs}(u)du$. For *semi-parametric* multi-state models fitted with `coxph`, the function `msfit` in **mstate** (de Wreede *et al.* 2010, 2011) provides piecewise-constant estimates and covariances for $H_{rs}(t)$. For the Cox models for the BOS data,

```
R> require("mstate")

Loading required package: mstate

R> mrcox <- msfit(crcox, trans = tmat)
R> mfcox <- msfit(cfcox, trans = tmat)
```

flexsurv provides an analogous function `msfit.flexsurvreg` to produce cumulative hazards from *fully-parametric* multi-state models in the same format. This is a short wrapper around `summary.flexsurvreg(..., type = "cumhaz")`, previously mentioned in §**??**. The difference from **mstate**'s method is that hazard estimates can be produced for any grid of times `t`, at any level of detail and even beyond the range of the data, since the model is fully parametric. The argument `newdata` can be used in the same way to specify a desired covariate category, though in this example there are no covariates in addition to the transition type. The name of the (factor) covariate indicating the transition type can also be supplied through the `tvar` argument, in this case it is the default, `"trans"`.

```
R> tgrid <- seq(0, 14, by = 0.1)
R> mrwei <- msfit.flexsurvreg(crwei, t = tgrid, trans = tmat)
R> mrexp <- msfit.flexsurvreg(crexp, t = tgrid, trans = tmat)
R> mfwei <- msfit.flexsurvreg(cfwei, t = tgrid, trans = tmat)
```

These can be plotted (Figure 2) to show the fit of the parametric models compared to the non-parametric estimates. Both models appear to fit adequately, though give diverging extrapolations after around 6 years when the data become sparse. The Weibull clock-reset model has an improved AIC of 1091, compared to 1099 for the exponential model. For the $2 \to 3$ transition, the clock-forward and clock-reset models give slightly different hazard trajectories.

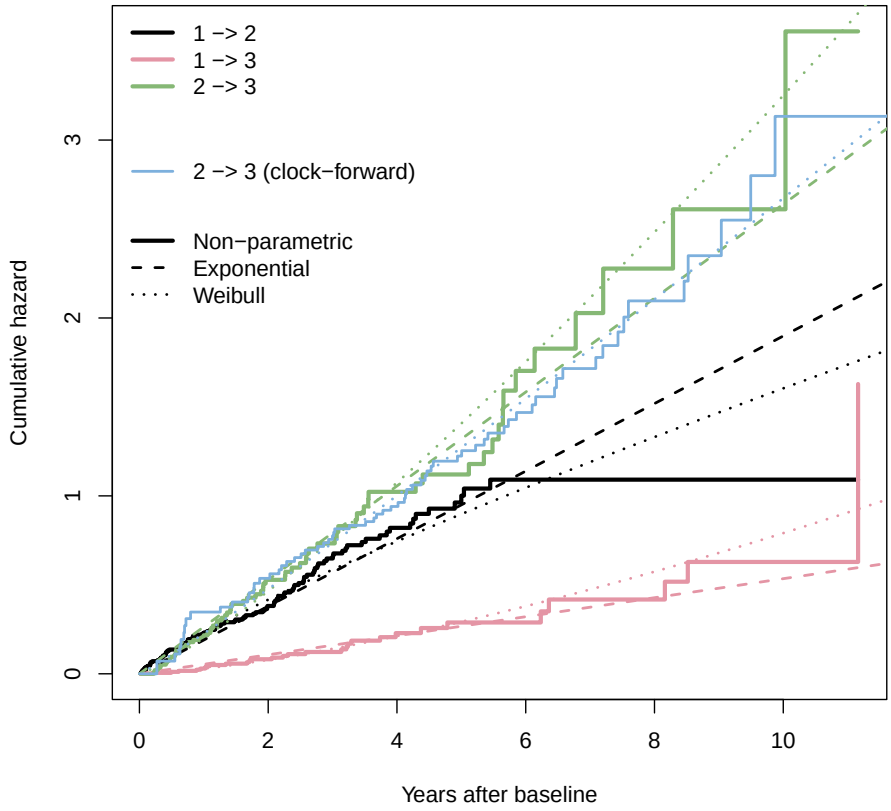


Figure 2: Cumulative hazards for three transitions in the BOS multi-state model (clock-reset), under non-parametric, exponential and Weibull models. For the $2 \to 3$ transition, an alternative clock-forward scale is shown for the non-parametric and Weibull models.

2.5. Prediction from parametric multi-state models with cause-specific hazards

The *transition probabilities* of the multi-state model are the probabilities of occupying each state s at time $t > t_0$, given that the individual is in state r at time t_0.

$$P(t_0, t) = \mathsf{P}(X(t) = s | X(t_0) = r)$$

Markov models For a time-inhomogeneous Markov model, these are related to the transition intensities via the Kolmogorov forward equation

$$\frac{dP(t_0, t)}{dt} = P(t_0, t)Q(t)$$

with initial condition $P(t_0, t_0) = I$ (Cox and Miller 1965). This can be solved numerically, as in Titman (2011). This is implemented in the function `pmatrix.fs`, using the **deSolve** package (Soetaert *et al.* 2010). This returns the full transition probability matrix $P(t_0, t)$ from time $t_0 = 0$ to a time or set of times t specified in the call. Under the Weibull model,

the probability of remaining alive and free of BOS is estimated at 0.3 at 5 years and 0.09 at 10 years:

```
R> pmatrix.fs(cfwei, t = c(5, 10), trans = tmat)
```

```
$`5`
          [,1]      [,2]      [,3]
[1,] 0.3042199 0.2521696 0.4436105
[2,] 0.0000000 0.2804171 0.7195829
[3,] 0.0000000 0.0000000 1.0000000

$`10`
           [,1]       [,2]      [,3]
[1,] 0.09117016 0.12047936 0.7883505
[2,] 0.00000000 0.06903846 0.9309615
[3,] 0.00000000 0.00000000 1.0000000

attr(,"nst")
[1] 3
```

Confidence intervals can be obtained by simulation from the asymptotic distribution of the maximum likelihood estimates — see `help(pmatrix.fs)` for full details. A similar function `totlos.fs` is provided to estimate the expected total amount of time spent in state s up to time t for a process that starts in state r, defined as $\int_{u=0}^{t} P(0,u)_{rs} du$.

Semi-Markov models For semi-Markov models, the Kolmogorov equation does not apply, since the transition intensity matrix $Q(t)$ is no longer a deterministic function of t, but depends on when the transitions occur between time t_0 and t. Predictions can then be made by simulation. The function `sim.fmsm` simulates trajectories from parametric semi-Markov models by repeatedly generating the time to the next transition until the individual reaches an absorbing state or a specified censoring time. This requires the presence of a function to generate random numbers from the underlying parametric distribution — and is fast for built-in distributions which use vectorised functions such as `rweibull`.

`pmatrix.simfs` calculates the transition probability matrix by using `sim.fmsm` to simulate state histories for a large number of individuals, by default 100000. Simulation-based confidence-intervals are also available in `pmatrix.simfs`, at an extra computational cost, and the expected total length of stay in each state is available from `totlos.simfs`.

```
R> pmatrix.simfs(crwei, trans = tmat, t = 5)
R> pmatrix.simfs(crwei, trans = tmat, t = 10)
```

Prediction via mstate Alternatively, predictions can be made by supplying the cumulative transition-specific hazards, calculated with `msfit.flexsurvreg`, to functions in the **mstate** package.

For Markov models, the solution to the Kolmogorov equation (e.g., Aalen *et al.* 2008) is given by a product integral, which can be approximated as

$$P(t_0, t) = \prod_{i=0}^{m-1} \{I + Q(t_i)dt\}$$

where a fine grid of times $t_0, t_1, \ldots, t_m = t$ is chosen to span the prediction interval, and $Q(t_i)dt$ is the increment in the cumulative hazard matrix between times t_i and t_{i+1}. Q may also depend on other covariates, as long as these are known in advance. In **mstate**, these can be calculated with the `probtrans` function, applied to the cumulative hazards returned by `msfit`. For Cox models, the time grid is naturally defined by the observed survival times, giving the Aalen-Johansen estimator (Andersen *et al.* 1993). Here, the probability of remaining alive and free of BOS is estimated at 0.27 at 5 years and 0.17 at 10 years.

```
R> ptc <- probtrans(mfcox, predt = 0, direction = "forward")[[1]]
R> round(ptc[c(165, 193),], 3)
```

```
     time pstate1 pstate2 pstate3   se1   se2   se3
165 4.999   0.273   0.294   0.433 0.037 0.039 0.040
193 9.873   0.174   0.040   0.786 0.040 0.022 0.045
```

For parametric models, using a similar discrete-time approximation was suggested by Cook and Lawless (2014). This is achieved by passing the object returned by `msfit.flexsurvreg` to `probtrans` in **mstate**. It can be made arbitrarily accurate by choosing a finer resolution for the grid of times when calling `msfit.flexsurvreg`.

```
R> ptw <- probtrans(mfwei, predt = 0, direction = "forward")[[1]]
R> round(ptw[ptw$time %in% c(5, 10),], 3)
```

```
    time pstate1 pstate2 pstate3   se1   se2   se3
51     5   0.300   0.254   0.446 0.034 0.033 0.037
101   10   0.089   0.119   0.792 0.027 0.032 0.041
```

`pstate1`–`pstate3` are close to the first rows of the matrices returned by `pmatrix.fs`. The discrepancy from the Cox model is more marked at 10 years when the data are more sparse (Figure 2). A finer time grid would be required to achieve a similar level of accuracy to `pmatrix.fs` for the point estimates, at the cost of a slower run time than `pmatrix.fs`. However, an advantage of `probtrans` is that standard errors are available more cheaply.

For semi-Markov models, **mstate** provides the function `mssample` to produce both simulated trajectories and transition probability matrices from semi-Markov models, given the estimated piecewise-constant cumulative hazards (Fiocco *et al.* 2008), produced by `msfit` or `msfit.flexsurvreg`, though this is generally less efficient than `pmatrix.simfs`. In this example, 1000 samples from `mssample` give estimates of transition probabilities that are accurate to within around 0.02. However with `pmatrix.simfs`, greater precision is achieved by simulating 100 times as many trajectories in a shorter time.

```
R> mssample(mrcox$Haz, trans = tmat, clock = "reset", M = 1000,
+           tvec = c(5, 10))
R> mssample(mrwei$Haz, trans = tmat, clock = "reset", M = 1000,
+           tvec = c(5, 10))
```

2.6. Next-state probabilities

In a multi-state situation we are usually interested in the probability that a person in one state will move to a specific state next, rather than to the other competing states. In the BOS example we might want to estimate the probability that a patient will die before getting BOS, or get BOS before dying. In a mixture multi-state model (Section 3), these are explicit parameters of the model. In a multi-state model parameterised by cause-specific hazards, these probabilities are related to the hazards through the Kolmogorov forward equation.

To obtain these, we consider the full multi-state model as a set of *competing risks submodels*. There is one submodel for each state a person can transition from (the "transient" states of the model). In the BOS example, there is one submodel for the transitions from no BOS (with two competing risks: BOS and death), and a second submodel for the single transition from BOS to death.

The probability that the the next state after r is s can then be obtained in practice as the transition probability $P(X(t) = s|X(t_0) = r)$, under the submodel for transient state r, for a very large time t. `pmatrix.fs` can be used for this.

```
R> tmat_nobos <- rbind("No BOS"=c(NA,1,2),
+ "BOS"=c(NA,NA,NA),"Death"=c(NA,NA,NA))
R> crfs_nobos <- fmsm(mod_nobos_bos, mod_nobos_death, trans = tmat_nobos)
R> pmatrix.fs(crfs_nobos, from=1, t=100000)["No BOS",c("BOS","Death")]

      BOS     Death
0.7177376 0.2822624
```

In a time-homogeneous Markov model, i.e. where the times from state r to state s are all exponentially distributed with a constant rate λ_{rs}, the probability that the next state after r is s is simply $\lambda_{rs}/\sum_u \lambda_{ru}$, where the sum is taken over all possible next states after r. We can check the result from `pmatrix.fs` agrees with this:

```
R> modexp_nobos_bos <- flexsurvreg(Surv(years, status) ~ 1, subset=(trans==1),
+                                  data = bosms3, dist = "exponential")
R> modexp_nobos_death <- flexsurvreg(Surv(years, status) ~ 1, subset=(trans==2),
+                                    data = bosms3, dist = "exponential")
R> crfs_nobos <- fmsm(modexp_nobos_bos, modexp_nobos_death, trans=tmat_nobos)
R> pmatrix.fs(crfs_nobos, from=1, t=100000)["No BOS",c("BOS","Death")]

      BOS     Death
0.7802948 0.2197052
```

```
R> rate12 <- modexp_nobos_bos$res["rate","est"]
R> rate13 <- modexp_nobos_death$res["rate","est"]
R> rate12 / (rate12  + rate13)

[1] 0.7802948
```

2.7. Distribution of the time to the next state

Under the cause-specific hazards model, the time until the next observed transition can be considered to equal the *minimum* of a set of latent times — one latent time for each potential transition whose cause-specific hazard defines the multi-state model.

The distribution of this time is not generally known analytically, given a set of parametric distributions defining the cause-specific hazards. For example, if there were two competing latent event times T_1, T_2 with cause-specific hazards distributed as Gamma(a_1, b_1) and Weibull(a_2, b_2) respectively, then the equivalent mixture model would be specified by the conditional distributions of $T_1|T_1 < T_2$ and $T_2|T_1 > T_2$, which wouldn't have a standard form. Instead this time can be computed using simulation, via `sim.fmsm`.

The function `simfinal_fmsm` wraps `sim.fmsm` to summarise the distribution of the time to each absorbing state in a multi-state model, conditionally on that state occurring. If this is applied to a *competing-risks submodel* for state r of a multi-state model, this simply produces the time to the *next* state in the full model, since the absorbing states of the submodel define the potential next states after state r.

This is done here to calculate the mean, median and interquartile range for the time to BOS (given survival) and the time to death (given no BOS before death). The function also produces estimates of the probability of each of these states, though as we saw in Section 2.6, this is available more efficiently via `pmatrix.fs`. Note the number of simulated individuals M can be increased for more accuracy, and optionally the B argument can be supplied to obtain simulation-based confidence intervals around the estimates.

```
R> crfs_nobos <- fmsm(mod_nobos_bos, mod_nobos_death, trans = tmat_nobos)
R> simfinal_fmsm(crfs_nobos, probs = c(0.25, 0.5, 0.75), M=10000)

# A tibble: 10 x 3
   state quantity   val
   <chr> <chr>    <dbl>
 1 BOS   25%      0.836
 2 BOS   50%      2.31
 3 BOS   75%      4.95
 4 BOS   mean     3.53
 5 BOS   prob     0.718
 6 Death 25%      2.32
 7 Death 50%      4.62
 8 Death 75%      7.91
 9 Death mean     5.71
10 Death prob     0.282
```

2.8. Obtaining probabilities of, and times to, a final absorbing state

As the previous section mentioned, `simfinal_fmsm` can be applied to any multi-state model with an absorbing state, to estimate the distribution of the time from the start of the process to the absorbing state. If there are multiple absorbing states, it can also estimate the probability that the individual ends up in each one. For the BOS model, there is only one absorbing state, death, so the function returns a probability of 1 that the patient will eventually die, and summaries of the distribution of the time from transplant to death.

```
R> simfinal_fmsm(crfs, probs = c(0.25, 0.5, 0.75), M=10000)
```

```
# A tibble: 5 x 3
  state   quantity   val
  <chr>   <chr>    <dbl>
1 State 3 25%       3.19
2 State 3 50%       5.60
3 State 3 75%       8.95
4 State 3 mean      6.60
5 State 3 prob      1
```

`simfinal_fmsm` requires a semi-Markov multi-state model, or a simple competing risks model, constructed through `fmsm`. Though for a simple competing risks model, `pmatrix.fs` can be used to get the probability of each competing state, as in 2.6.

3. Multi-state modelling using mixtures

3.1. Definitions

A "mixture" multi-state model, first described for competing risks models by Larson and Dinse (1985) is described in terms of:

- the probability π_{rs} that the next state after state r is state s
- the distribution of the time T_{rs} from state r entry to state s entry, given that the next state after r is state s.

In other words, the transition intensity is defined by

$$q_{rs}(t) = \begin{cases} q^*_{rs}(t) & \text{if } I_r = s \\ 0 & \text{if } I_r \neq s \end{cases}$$

where I_r is a latent categorical variable that determines which transition will happen next for an individual in state r, governed by probabilities $\pi_{r,s} = P(I_r = s)$, with $\sum_{s \in \mathcal{S}_r} \pi_{r,s} = 1$ over the potential next state $\mathcal{S}_r$ after state r. The transition intensity $q^*_{rs}(t)$ is defined by the hazard function of a parametric distribution that governs the time S_{rs} from state r entry until the transition to state s, *given that this is the transition that occurs.*

The mixture model describes a mechanism where the transition that happens is determined "in advance" at time zero, whereas in the cause-specific hazards model, the risks of different events "compete" with each other as time proceeds. However, in practice, both models can represent situations where individuals can experience any of the competing events. In the mixture model, we only specify a distribution for the time to the event *that happens first* among the competing risks, and do not model events that don't occur. As discussed by Cox (1959), if the time-to-event distributions are specified correctly in both frameworks, then they can both represent the same process. In practice however, they will differ. As discussed in Section 2.7, given a particular parametric form for a set of cause-specific hazards, the form of the distribution for the *minimum* of the potential event times, the one that actually happens, won't be available. The main advantage of the mixture model is that it is parameterised in terms of quantities that have an easy, direct interpretation.

In the mixture model, we specify a parametric distribution with density $f_{r,s}(|\theta_{r,s})$ (and CDF $F_{r,s}()$) for the time of transition to state s for a person in state r, conditionally on this transition being the one that occurs. We have data consisting of state transitions, indexed by j, experienced by individuals i. This can either be

- *an exact transition time:* $\{y_{i,j}, r_{i,j}, s_{i,j}, \delta_{i,j} = 1\}$, where a transition to state $s_{i,j}$ is known to occur at a time $y_{i,j}$ after entry to state $r_{i,j}$.
- *right censoring* $\{y_{i,j}, r_{i,j}, \delta_{i,j} = 2\}$, where an individual's follow-up ends while they are in state $r_{i,j}$, at time $y_{i,j}$ after entering this state, thus the next state and the time of transition to it are unknown.

Therefore, for an exact time of transition to a known state $s_{i,j}$, i.e. $\delta_{i,j} = 1$, the likelihood contribution is simply

$$l_{i,j} = \pi_{r_{i,j}s_{i,j}} f_{r_{i,j},s_{i,j}}(y_{i,j}|\theta_{r_{i,j},s_{i,j}})$$

For observations j of right-censoring at $y_{i,j}$, the state that the person will move to, and the time of that transition, is unknown. Thus it is unknown which of the distributions $f_{r,s}$ the transition time will obey, and the likelihood contribution is of the form of a mixture model, that sums over the set $\mathcal{S}_r$ of potential next states after r:

$$l_{i,j} = \sum_{s \in \mathcal{S}_r} \pi_{r_{i,j}s}(1 - F_{r_{i,j},s}(y_{i,j}|\theta_{r_{i,j},s}))$$

3.2. Data for mixture competing risks models

The required data format for the mixture model is shown for the BOS data in the object `bosmx3`. This has

- one row for each observed event time
- one row for each "end of follow-up" time where we know an individual was still at risk of making a transition, but we don't know which transition will occur.

Recall that in the data `bosms3` used to implement the cause-specific hazards model, a individual "censored" at time t contributed a row in the data *for each* of the events that might have occurred after t. The difference from `bosmx3` is that for these individuals, `bosmx3` only has *one* row per "censored" individual. For example, patient 5 is censored at 11 years, when they were still at risk of either BOS or death. Also we do not include censoring times for events competing with events that were observed, e.g. when patient 4 below got BOS (state 2) at year 2, thus were "censored" at the same time for the transition from state 1 (no BOS) to state 3 (death).

The process of transforming the cause-specific dataset to the mixture dataset is shown below. Multiple censoring records, for different destination states, are condensed to a single censoring record. Each row of `bosmx3` then corresponds to a transition. A new column called `event` is created, which should be a factor that identifies the terminal event of the transition, which takes the value `NA` in cases of censoring where the transition that will happen is unknown. Columns not needed for the mixture model are removed. Informative labels for the factor levels of `event` are added to identify the states.

```
R> bosms3[bosms3$id  %in% c(4,5),]

An object of class 'msdata'

Data:
   id from to   Tstart     Tstop     years status trans
10  4    1  2 0.000000  2.086242  2.086242      1     1
11  4    1  3 0.000000  2.086242  2.086242      0     2
12  4    2  3 2.086242 12.120465 10.034223      1     3
13  5    1  2 0.000000 10.976044 10.976044      0     1
14  5    1  3 0.000000 10.976044 10.976044      0     2

R> bosmx3 <- bosms3
R> bosmx3$Tstart <- bosmx3$Tstop <- bosmx3$trans <- NULL
R> bosmx3 <- bosmx3[!(bosmx3$status==0 & duplicated(paste(bosmx3$id, bosmx3$from))),]
R> bosmx3$event <- ifelse(bosmx3$status==0, NA, bosmx3$to)
R> bosmx3$event <- factor(bosmx3$event, labels=c("BOS","Death"))
R> bosmx3$to <- NULL
R> bosmx3[bosmx3$id %in% c(4,5),]

An object of class 'msdata'

Data:
   id from     years status event
10  4    1  2.086242      1   BOS
12  4    2 10.034223      1 Death
13  5    1 10.976044      0  <NA>
```

3.3. Fitting mixture competing risks models

The `flexsurvmix` function fits a mixture model to competing risks data. To fit a full multi-state mixture model, we fit one mixture competing risks model for each transient state in the

multi-state structure. The models are then grouped together (Section 3.5) to form the full multi-state model.

The mixture model for the event following transplant (BOS or death before BOS, the 1-2 and 1-3 transitions in the multi-state structure) is fitted as follows. A Weibull distribution is fitted for the time to BOS given BOS onset, and an exponential distribution is fitted for the time to death given death before BOS onset. These distributions are specified in the `dists` argument. The same distributions as in `flexsurvreg` are supported (including custom distributions, in the same way).

```
R> bosfs <- flexsurvmix(Surv(years, status) ~ 1, event=event,
+                       data=bosmx3[bosmx3$from==1,],
+             dists = c(BOS="weibull", Death="exponential"))
R> bosfs
```

```
Call:
flexsurvmix(formula = Surv(years, status) ~ 1, data = bosmx3[bosmx3$from ==
    1, ], event = event, dists = c(BOS = "weibull", Death = "exponential"))

Estimates:
   component         dist  terms     est    est.t     se
1        BOS              prob1  0.5889       NA     NA
2      Death              prob2  0.4111  -0.3596  0.210
3        BOS      weibull shape  1.0474   0.0463  0.100
4        BOS      weibull scale  2.5689   0.9435  0.155
5      Death  exponential  rate  0.0784  -2.5454  0.207

Log-likelihood = -401.4207, df = 4
AIC = 810.8414
```

The printed results object shows the two event probabilities π_{rs} in the first two rows, and the parameters of the time-to-event distributions in the remaining rows. The maximum likelihood estimates are in column `est`, and estimates on a (multinomial) logit or log transformed scale are in column `est.t`, with standard errors on the transformed scale in `se`.

Covariates can be included on the event probabilities through multinomial logistic regression. For illustration, we just simulate some fake covariate data in the variable `x`. The log odds ratio for this covariate on the odds of death (with the first category, BOS here, always taken as the baseline) is shown in the row with `terms` labelled `prob2(x)`.

```
R> set.seed(1)
R> bosmx3$x <- rnorm(nrow(bosmx3),0,1)
R> bosfsp <- flexsurvmix(Surv(years, status) ~ 1, event=event,
+                        data=bosmx3[bosmx3$from==1,],
+             dists = c(BOS="weibull", Death="exponential"),
+             pformula = ~ x)
R> bosfsp
```

```
Call:
flexsurvmix(formula = Surv(years, status) ~ 1, data = bosmx3[bosmx3$from ==
    1, ], event = event, dists = c(BOS = "weibull", Death = "exponential"),
    pformula = ~x)

Estimates:
   component         dist     terms      est    est.t
1        BOS                  prob1   0.5837       NA
2      Death                  prob2   0.4163  -0.3378
3      Death               prob2(x)  -0.1221  -0.1221
4        BOS      weibull     shape   1.0530   0.0516
5        BOS      weibull     scale   2.5336   0.9296
6      Death  exponential      rate   0.0779  -2.5523
      se
1     NA
2  0.205
3  0.183
4  0.099
5  0.150
6  0.204

Log-likelihood = -401.1932, df = 5
AIC = 812.3865
```

Covariates may also be included on any parameter of any time-to-event distribution, as in `flexsurvreg`. If the covariate is named on the right hand side of the `Surv` formula in the first argument, then it is included on the location parameter of all distributions.

```
R> bosfst <- flexsurvmix(Surv(years, status) ~ x, event=event,
+                        data=bosmx3[bosmx3$from==1,],
+             dists = c(BOS="weibull", Death="exponential"))
```

The first argument may be a list of formulae, to indicate that different covariates are included on the location parameter for different events.

```
R> flexsurvmix(list(Surv(years, status) ~ x,
+                   Surv(years, status) ~ 1),
+                   event=event, data=bosmx3[bosmx3$from==1,],
+             dists = c(BOS="weibull", Death="exponential"))

Call:
flexsurvmix(formula = list(Surv(years, status) ~ x, Surv(years,
    status) ~ 1), data = bosmx3[bosmx3$from == 1, ], event = event,
    dists = c(BOS = "weibull", Death = "exponential"))

Estimates:
```

```
   component         dist  terms      est    est.t     se
1        BOS              prob1   0.5979       NA     NA
2      Death              prob2   0.4021  -0.3967  0.227
3        BOS      weibull shape   1.0373   0.0366  0.103
4        BOS      weibull scale   2.6705   0.9823  0.172
5        BOS      weibull     x  -0.0981  -0.0981  0.132
6      Death  exponential  rate   0.0802  -2.5238  0.215

Log-likelihood = -401.1442, df = 5
AIC = 812.2884
```

An argument `anc` is used to supply covariates on ancillary parameters (e.g. shape parameters). This must be a list of length matching the number of competing events, each component being a named list in the format used for the `anc` argument in `flexsurvreg`. If there are no covariates for a particular component, just specify a null formula on the location parameter (as below, `rate=~1`).

```
R> flexsurvmix(Surv(years, status) ~ 1,
+              event=event, data=bosmx3[bosmx3$from==1,],
+              dists = c(BOS="weibull", Death="exponential"),
+              anc = list(BOS=list(shape=~x),
+                         Death=list(rate=~1)))

Call:
flexsurvmix(formula = Surv(years, status) ~ 1, data = bosmx3[bosmx3$from ==
    1, ], event = event, dists = c(BOS = "weibull", Death = "exponential"),
    anc = list(BOS = list(shape = ~x), Death = list(rate = ~1)))

Estimates:
   component         dist     terms     est    est.t     se
1        BOS                  prob1  0.5898       NA     NA
2      Death                  prob2  0.4102  -0.3633  0.208
3        BOS      weibull     shape  1.0413   0.0405  0.100
4        BOS      weibull     scale  2.5657   0.9422  0.152
5        BOS      weibull  shape(x)  0.0514   0.0514  0.102
6      Death  exponential      rate  0.0787  -2.5419  0.206

Log-likelihood = -401.2939, df = 5
AIC = 812.5877
```

3.4. Predictions from mixture competing risks models

A few functions have been supplied to extract estimates and confidence intervals for interpretable quantities, for given covariate values. Here we extract estimates of various quantities for covariate values of `x=0` and `x=1`. These values are provided in a data frame `nd` that will be supplied as the `newdata` argument to various extractor functions.

The first extractor function `probs_flexsurvmix` gives the fitted event probabilities, by event and covariate value. This is applied here to the fitted model `bosfsp` that included a covariate on the event probabilities. All these functions take an argument `B` which specifies the number of samples to use for calculating bootstrap-like confidence limits — increase this for more accurate limits.

```
R> nd <-  data.frame(x=c(0,1))
R> probs_flexsurvmix(bosfsp, newdata=nd, B=50)
```

```
# A tibble: 4 x 5
      x event   val lower upper
  <dbl> <chr> <dbl> <dbl> <dbl>
1     0 BOS   0.584 0.482 0.680
2     0 Death 0.416 0.320 0.518
3     1 BOS   0.613 0.494 0.696
4     1 Death 0.387 0.304 0.506
```

`mean_flexsurvmix` extracts the mean time to each event conditionally on that event occurring, from the mean of the fitted time-to-event distribution. `quantile_flexsurvmix` extracts the quantiles of these distributions, e.g. the interquartile range in this example summarises the variability between individuals for this time.

```
R> mean_flexsurvmix(bosfst, newdata=nd)
```

```
    event x       val
1     BOS 0  2.562386
1.1 Death 0 13.005489
2     BOS 1  2.410818
2.1 Death 1  9.429885
```

```
R> quantile_flexsurvmix(bosfst, B=50, newdata=nd, probs=c(0.25, 0.5, 0.75))
```

```
      event    p x        val      lower     upper
1       BOS 0.25 0  0.7898311  0.5850319  1.113396
2       BOS 0.25 1  0.7431118  0.4994530  1.124525
1.1     BOS 0.50 0  1.8345086  1.3328861  2.365270
2.1     BOS 0.50 1  1.7259954  1.1782982  2.542817
1.2     BOS 0.75 0  3.5644760  2.4455657  4.836203
2.2     BOS 0.75 1  3.3536334  2.3152695  4.937997
1.3   Death 0.25 0  3.7414461  2.7042976  5.657102
2.3   Death 0.25 1  2.7128089  1.7079124  4.659759
1.1.1 Death 0.50 0  9.0147182  6.5157911 13.630340
2.1.1 Death 0.50 1  6.5362984  4.1150798 11.227320
1.2.1 Death 0.75 0 18.0294365 13.0315822 27.260679
2.2.1 Death 0.75 1 13.0725968  8.2301596 22.454640
```

`p_flexsurvmix` extracts the probability of being in each of the states at a time t in the future, shown for $t = 5, 10$ here. The `startname` argument supplies an informative name to the "starting" state that the model `bosfst` describes transitions from.

```
R> p_flexsurvmix(bosfst, t=c(5, 10), B=10, startname="No BOS")
```

```
# A tibble: 6 x 6
      t     x state    val lower upper
  <dbl> <dbl> <chr>  <dbl> <dbl> <dbl>
1     5     0 No BOS 0.360 0.317 0.411
2     5     0 BOS    0.510 0.446 0.558
3     5     0 Death  0.130 0.102 0.187
4    10     0 No BOS 0.199 0.142 0.251
5    10     0 BOS    0.582 0.517 0.626
6    10     0 Death  0.219 0.176 0.287
```

3.5. Forming a mixture multi-state model

We supplement the mixture model for the event following state 1 (no BOS) with a second model `bosfst_bos` for the events following BOS (the other transient state in the multi-state model). Although this is fitted with `flexsurvmix`, there is only one potential next state after BOS, which is death, so internally this just fits a standard survival model using `flexsurvreg`.

The two mixture models are then coupled together using the `fmixmsm` function, which returns an object containing the entire multi-state model, here named `bosfmix`. This object contains attributes listing the potential pathways that an individual can take through the states. These pathways are identified automatically by naming the arguments to `fmixmsm` with the label of the state that each model describes transitions from — here the first model `bosfst` describes transitions from "No BOS" and the second describes transitions from BOS. (Note we redefine the `event` factor so that empty levels are dropped)

```
R> bdat <- bosmx3[bosmx3$from==2,]
R> bdat$event <- factor(bdat$event)
R> bosfst_bos <- flexsurvmix(Surv(years, status) ~ x, event=event, data=bdat,
+                            dists = c(Death="exponential"))
R> bosfmix <- fmixmsm("No BOS"=bosfst, "BOS"=bosfst_bos)
```

Now we can predict outcomes from the full multi-state model. All these functions work by simulating a large population of individuals from the model, by default `M=10000`, so increase this for more accuracy. A cut-off time `t=1000` is also applied to the simulated data that assumes people have all reached the absorbing state by this time – increase this if necessary. These functions currently require models to have at least one absorbing state, and no "cycles" in their transition structure.

The function `ppath_fmixmsm` returns the probability of following each of the potential pathways through the model. For this example, they are the same for both covariate values `x=0` and `x=1`. Setting `final=TRUE` aggregates the pathway probabilities by the final (absorbing) state, returning the probability of ending in each of the potential final states, which is trivially 1 in this case for the single absorbing state of death.

```
R> ppath_fmixmsm(bosfmix, B=20)

  final          pathway       val     lower     upper
1 Death No BOS-BOS-Death 0.5917815 0.5313479 0.6827631
2 Death     No BOS-Death 0.4082185 0.3172369 0.4686521

R> ppath_fmixmsm(bosfmix, B=20, final=TRUE)

  final val lower upper
1 Death   1     1     1
```

To estimate the mean time to the final state for individuals following each pathway, use `meanfinal_fmixmsm`, and for the quantiles of the distribution of this time over individuals (by default the median and a 95% interval), use `qfinal_fmixmsm`. Supplying `final=TRUE` in these functions summarises the mean time to the final state, averaged over all potential pathways (according to the probability of following those pathways). Again, covariate values can be supplied in the `newdata` argument if needed.

```
R> meanfinal_fmixmsm(bosfmix, B=10)

  final          pathway       val    lower     upper
1 Death No BOS-BOS-Death  6.368014 5.963077  7.831323
2 Death     No BOS-Death 13.005489 9.218586 15.975117

R> qfinal_fmixmsm(bosfmix, B=10)

           pathway probs        val      lower      upper
1 No BOS-BOS-Death 0.025  0.7519363  0.6641109  0.9695770
2 No BOS-BOS-Death 0.500  5.3042370  4.7615652  6.9108963
3 No BOS-BOS-Death 0.975 17.6039946 16.0993207 24.3280645
4     No BOS-Death 0.025  0.3192801  0.2237027  0.3978458
5     No BOS-Death 0.500  9.1410207  6.2659597 11.0128012
6     No BOS-Death 0.975 47.3588137 33.6342476 57.8299514

R> meanfinal_fmixmsm(bosfmix, B=10, final=TRUE)

  final      val    lower    upper
1 Death 9.077554 7.925123 11.52372
```

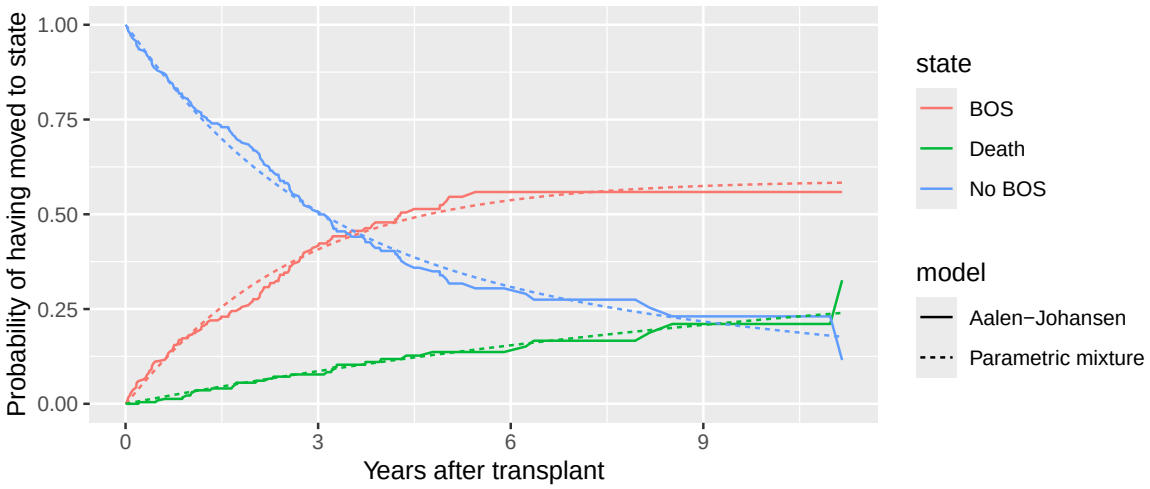
3.6. Goodness of fit checking

The fit of mixture competing risks models can be checked by comparing predictions of the state occupancy probabilities (as returned by `p_flexsurvmix`) with nonparametric estimates of these quantities (Aalen and Johansen 1978). This is only supported for models with no covariates or only factor covariates (where subgroup-specific predictions are compared with

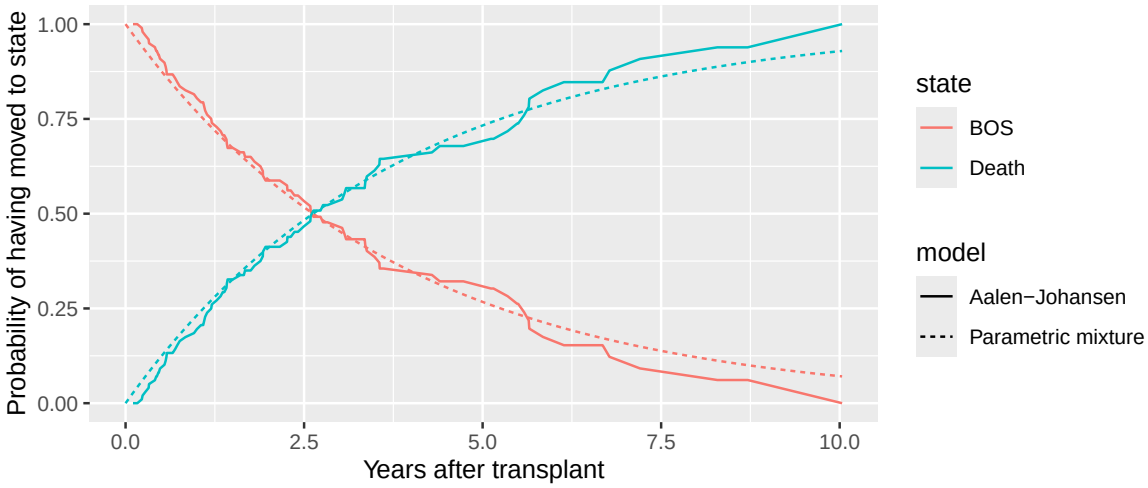
stratified nonparametric estimates). The function `ajfit_flexsurvmix` derives these estimates from both the mixture model and the Aalen-Johansen method, and collects them in a tidy data frame ready for plotting, e.g. with the **ggplot2** package.

The mixture models fit nicely here. A `start` argument is supplied to give an informative label to the starting state in the plots. Note we are plotting not probability of *being in* a state at time t, but the probability of *having made* the respective transition — here we are checking the competing risks submodels one at time.

```
R> aj <- ajfit_flexsurvmix(bosfs, start="No BOS")
R> bosfs_bos <- flexsurvmix(Surv(years, status) ~ 1, event=event, data=bdat,
+                           dists = c(Death="exponential"))
R> aj2 <- ajfit_flexsurvmix(bosfs_bos, start="BOS")
R> library(ggplot2)
R> ggplot(aj, aes(x=time, y=val, lty=model, col=state)) +
+   geom_line() +
+   xlab("Years after transplant") + ylab("Probability of having moved to state")
```

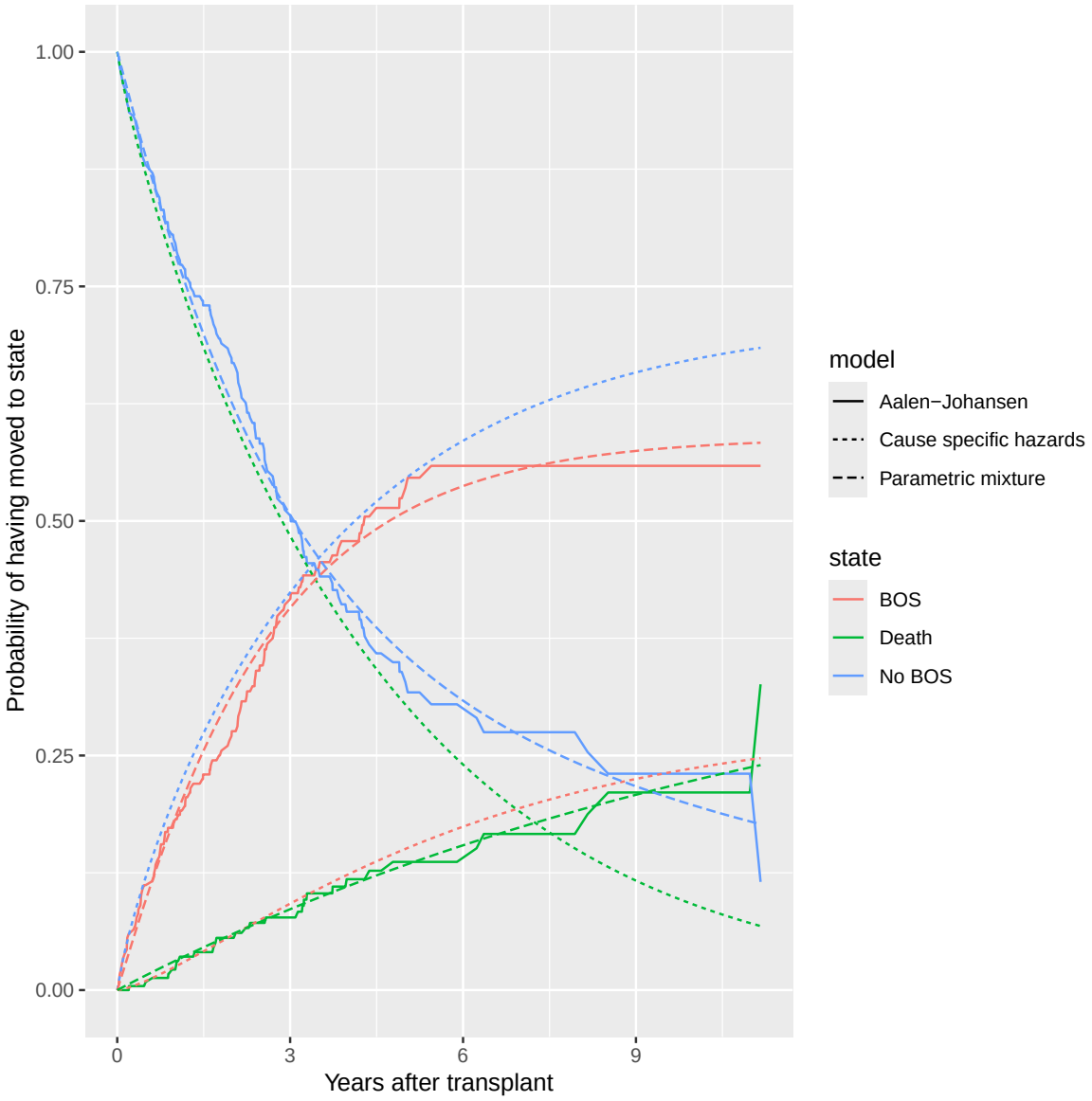



```
R> ggplot(aj2, aes(x=time, y=val, lty=model, col=state)) +
+   xlab("Years after transplant") + ylab("Probability of having moved to state") +
+   geom_line()
```



Checking against Aalen-Johansen estimates is also supported for cause-specific hazards models that are grouped together using `fmsm`. Thus we can compare cause-specific hazards and mixture models that have been fitted to the same data. We do this here for the `fmsm` object `crfs_nobos` created in Section 2.6.

```
R> library(dplyr)
R>
R> aj3 <- ajfit_fmsm(crfs_nobos) %>%
+   filter(model=="Parametric") %>%
+   mutate(model = "Cause specific hazards") %>%
+   mutate(state = factor(state, labels=c("No BOS","BOS","Death"))) %>%
+   full_join(aj)
R>
R> ggplot(aj3, aes(x=time, y=val, lty=model, col=state)) +
+   xlab("Years after transplant") + ylab("Probability of having moved to state") +
+   geom_line()
```



4. Comparison of frameworks for parametric multi-state modelling

As the mixture model is described by the probabilities of observable events and the times to those events, it is somewhat easier to interpret then the cause-specific hazards model. While those quantities can be computed under the cause-specific hazards model, they require potentially-expensive simulation. An advantage of the cause-specific hazards model is that it tends to be faster to *fit*, if the transition-specific models are fitted independently.

Another competing-risks modelling framework not implemented here is typically referred to as *subdistribution* hazard modelling — essentially covariate effects are applied to the probabilities of occupying different states at time t, rather than to the cause-specific hazards. This leads to covariate effects that are easier to interpret compared to, e.g. cause-specific hazard

ratios. The method has been implemented semiparametrically (Fine and Gray 1999) and parametrically (Lambert *et al.* 2017).

A further framework referred to as "vertical" modelling (Nicolaie *et al.* 2010) involves factorising the joint distribution of events and event times as $P(time)P(event|time)$, whereas the mixture modelling framework uses $P(event)P(time|event)$.

References

Aalen O, Borgan O, Gjessing H (2008). *Survival and Event History Analysis: A Process Point of View*. Springer-Verlag.

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.

Andersen PK, Borgan O, Gill RD, Keiding N (1993). *Statistical Models Based On Counting Processes*. Springer-Verlag.

Andersen PK, Keiding N (2002). "Multi-state models for event history analysis." *Statistical Methods in Medical Research*, **11**(2), 91–115.

Cook RJ, Lawless JF (2014). "Statistical Issues in Modeling Chronic Disease in Cohort Studies." *Statistics in Biosciences*, **6**(1), 127–161.

Cox DR (1959). "The analysis of exponentially distributed life-times with two types of failure." *Journal of the Royal Statistical Society: Series B (Methodological)*, **21**(2), 411–421.

Cox DR, Miller HD (1965). *The Theory of Stochastic Processes*. Chapman and Hall.

de Wreede L, Fiocco M, Putter H (2010). "The **mstate** Package for Estimation and Prediction in Non-and Semi-Parametric Multi-State and Competing Risks Models." *Computer Methods and Programs in Biomedicine*, **99**(3), 261–274.

de Wreede LC, Fiocco M, Putter H (2011). "**mstate**: An R Package for the Analysis of Competing Risks and Multi-State Models." *Journal of Statistical Software*, **38**, 1–30.

Fine JP, Gray RJ (1999). "A proportional hazards model for the subdistribution of a competing risk." *Journal of the American Statistical Association*, **94**(446), 496–509.

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.

Heng D, Sharples L, McNeil K, Stewart S, Wreghitt T, Wallwork J (1998). "Bronchiolitis Obliterans Syndrome: Incidence, Natural History, Prognosis, and Risk Factors." *The Journal of Heart and Lung Transplantation*, **17**(12), 1255–1263.

Ieva F, Jackson CH, Sharples LD (2015). "Multi-State modelling of repeated hospitalisation and death in patients with Heart Failure: the use of large administrative databases in clinical epidemiology." *Statistical Methods in Medical Research*. Early view.

Jackson CH (2011). "Multi-State Models for Panel Data: The **msm** Package for R." *Journal of Statistical Software*, **38**(8).

Kalbfleisch J, Lawless J (1985). "The Analysis of Panel Data under a Markov Assumption." *Journal of the American Statistical Association*, **80**(392), 863–871.

Lambert PC, Wilkes SR, Crowther MJ (2017). "Flexible parametric modelling of the cause-specific cumulative incidence function." *Statistics in Medicine*, **36**(9), 1429–1446.

Larson MG, Dinse GE (1985). "A mixture model for the regression analysis of competing risks data." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **34**(3), 201–211.

Nicolaie MA, van Houwelingen HC, Putter H (2010). "Vertical modeling: a pattern mixture approach for competing risks modeling." *Statistics in Medicine*, **29**(11), 1190–1205.

Putter H, Fiocco M, Geskus RB (2007). "Tutorial in Biostatistics: Competing Risks and Multi-State Models." *Statistics in Medicine*, **26**(8), 2389–2430.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package **deSolve**." *Journal of Statistical Software*, **33**(9), 1–25. ISSN 1548-7660. URL http://www.jstatsoft.org/v33/i09.

Titman AC (2011). "Flexible Nonhomogeneous Markov Models for Panel Observed Data." *Biometrics*, **67**(3), 780–787.