Journal of Statistical Software

January 2021, Volume 97, Issue 3. *doi: 10.18637/jss.v097.i03*

Simulating Survival Data Using the simsurv R Package

Samuel L. Brilleman
Monash University

Rory Wolfe
Monash University

Margarita Moreno-Betancur
Murdoch Children's Research Institute
University of Melbourne

Michael J. Crowther
University of Leicester

Abstract

The **simsurv** R package allows users to simulate survival (i.e., time-to-event) data from standard parametric distributions (exponential, Weibull, and Gompertz), two-component mixture distributions, or a user-defined hazard function. Baseline covariates can be included under a proportional hazards assumption. Clustered event times, for example individuals within a family, are also easily accommodated. Time-dependent effects (i.e., non-proportional hazards) can be included by interacting covariates with linear time or a user-defined function of time. Under a user-defined hazard function, event times can be generated for a variety of complex models such as flexible (spline-based) baseline hazards, models with time-varying covariates, or joint longitudinal-survival models.

Keywords: survival, time-to-event, simulation, R.

1. Introduction

In survival analysis (also known as time-to-event analysis) one is concerned with the analysis of an outcome variable that corresponds to the time from some defined baseline until the occurrence of an event of interest. For instance, in clinical studies, one may be interested in the time from diagnosis of a disease until the occurrence of death. Due to right censoring, the most common approach for analyzing such data is not to model the time-to-event outcome directly, but instead to model the rate of occurrence of the event. The hazard of the event at time t is

defined as the instantaneous rate of occurrence for the event at time t. Mathematically, it is

$$h_i(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T_i^* < t + \Delta t \mid T_i^* \geq t)}{\Delta t},$$

where T_i^* is the so-called true event time for individual i, which may or may not be observed due to right-censoring, and Δt is the width of some small time interval. In practice, one generally observes a time $T_i = \min(T_i^*, C_i)$ where C_i is a right-censoring time for individual i, and an event status indicator d_i which takes the value 1 if $T_i^* \leq C_i$ and value 0 otherwise. Since one often wishes to assess the effect of covariate(s) on the rate of the event, the proportional hazards model is commonly used for analysis. The proportional hazards model allows covariates to have a multiplicative effect on the hazard through the following model formulation

$$h_i(t) = h_0(t) \exp(X_i^\top \beta), \tag{1}$$

where $h_0(t)$ is the baseline hazard at time t, X_i is a vector of covariates for individual i, and β is a vector of population-level (i.e., fixed effect) parameters. The baseline hazard $h_0(t)$ can be left unspecified (as in the Cox proportional hazards model) or specified using a parametric form. If a parametric baseline hazard is specified, then this should have sufficient flexibility to model the underlying changes in the hazard over time given the context of the application.

When conducting simulation studies to evaluate the performance of new and existing statistical methods for analyzing survival data, one is required to simulate event times under a known data generating model. Similarly, one may need to simulate event times for the purpose of power calculations when designing new studies.

Traditionally, a common approach has been to make simplifying parametric assumptions about the distribution of the event times. For example, many authors limit their simulation studies to settings in which the simulated event times are drawn from exponential or Weibull distributions. However, the exponential and Weibull distributions may be unrealistic in many settings. This is because the former assumes a constant hazard (i.e., instantaneous rate of the event) over time, whilst the latter assumes a monotonically increasing or decreasing hazard over time. These limiting forms for the hazard function are likely to be implausible in many applications where the underlying hazard function may have one or more turning points. Moreover, one may wish to accommodate the effects of covariates (for example, a treatment effect) in the data generating model when simulating the event times, either through a proportional hazards assumption or through a more complicated time-dependent association.

A general method, known as the cumulative hazard inversion method (Leemis 1987; Bender, Augustin, and Blettner 2005), allows one to simulate event times under a proportional hazards data generating model with any parametric formulation for the baseline hazard. The cumulative hazard inversion method allows one to easily simulate an event time T_i^s for individual i by drawing a uniform random variable $U_i \sim U(0, 1)$ and evaluating

$$T_i^s = H_0^{-1}\left(-\log(U_i) \exp(-X_i^\top \beta)\right),$$

where $H_0^{-1}(.)$ corresponds to the inverted cumulative baseline hazard. Therefore, all that is required to generate simulated event times under the cumulative hazard inversion method is an invertible cumulative baseline hazard function and access to independent draws of the random uniform variable U_i. The latter is easily obtainable, since standard statistical software

packages have built in functions to simulate uniform random variables. However, access to the former will depend on how complex the definition of the baseline hazard is. If an analytical form for the inverted cumulative baseline hazard can be obtained, then the cumulative hazard inversion method is simple and computationally efficient.

In situations where the cumulative baseline hazard is not invertible, or where there is no closed form solution to the cumulative hazard function, simulating event times is less straightforward. To overcome these difficulties, Crowther and Lambert (2013) proposed an algorithm which nested numerical integration inside numerical root finding. The numerical integration is used to evaluate the cumulative hazard in situations where it does not have a closed form, while the numerical root finding is used to invert the cumulative hazard in situations where it is not analytically invertible. Using their algorithm it becomes possible to simulate event times under any data generating model for which it is possible to write down the formulation of the hazard or log hazard function. This includes, for example, models with time-varying covariates or models with time-dependent effects (i.e., non-proportional hazards).

The **simsurv** package (Brilleman 2021) described in this article, written in the R software (R Core Team 2020), and available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=simsurv`, is an implementation of the methods described by Crowther and Lambert (2013). The package is based on a corresponding package introduced in Crowther and Lambert (2012), which is available in the Stata software (StataCorp. 2015). The **simsurv** package allows one to simulate event times from standard parametric distributions (exponential, Weibull, and Gompertz), two-component mixture distributions, or a user-defined hazard, log hazard, cumulative hazard, or log cumulative hazard function. The two-component mixture distributions can allow for a variety of flexible baseline hazard functions. The effects of baseline covariates can be included under a proportional hazards assumption. Moreover, time-dependent effects (i.e., non-proportional hazards) can be included by interacting covariates with linear time or some user-defined transformation of time. Lastly, the option to provide a user-defined hazard function means that one can simulate under complex model formulations, for example, Royston and Parmar's flexible parametric model (Royston and Parmar 2002) or a joint longitudinal-survival model (Henderson, Diggle, and Dobson 2000; Tsiatis and Davidian 2004).

Other packages currently exist in R for simulating event times. The `simulate()` function in the **mlt** package (Hothorn 2020) allows the user to simulate event times under models where the conditional cumulative distribution function has a closed form expression. As special cases, this includes parametric proportional hazard models where the baseline hazard can be integrated analytically, as well as parametric hazard models with time-dependent effects (i.e., non-proportional hazards) as long as they are specified on the cumulative hazard or log cumulative hazard scales (e.g., Royston and Parmar 2002, models). The **prodlim** (Gerds 2019), **SimHaz** (Xiong, Pokaprakarn, Udagawa, and Rabbee 2015), and **SimSCRPiecewise** (Chapple 2016) packages each provide functions for simulating event times under a limited set of data generating models. The **survsim** package (Moriña and Navarro 2014) allows users to simulate event times from standard parametric distributions, specifically the Weibull, log-normal, and log-logistic distributions with extensions that focus primarily on multiple (i.e., competing) or recurrent events. The **PermAlgo** package (Sylvestre, Evans, MacKenzie, and Abrahamowicz 2017) allows users to simulate event times conditional on time-varying covariates using an extension of the permutational algorithm described by Abrahamowicz, MacKenzie, and Esdaile (1996) and Sylvestre and Abrahamowicz (2008).

This article is structured as follows. In Section 2 we describe the methodological framework for simulating event times under a general hazard model formulation, potentially incorporating the effects of time-fixed or time-varying covariates. In Section 3 we describe the implementation of the methods in the **simsurv** package, including a description of the arguments to the package's main function. In Section 4 we describe usage of the package through a series of examples. Specifically, we replicate examples from Crowther and Lambert (2012) and Crowther and Lambert (2013). These include simulating event times from a standard parametric distribution, a survival model with clustered event times, a flexible parametric Royston-Parmar survival model, a survival model with time-dependent effects (i.e., non-proportional hazards), and a joint longitudinal-survival model (i.e., with a time-varying covariate) (Royston and Parmar 2002). In Section 5 we close with a discussion.

2. Methodological framework

2.1. Simulating event times under a proportional hazards assumption

The survival function for individual i is the probability that their so-called true event time T_i^* is greater than the current time t. That is, the survival function can be defined as

$$S_i(t) = P(T_i^* > t).$$

Moreover, the corresponding probability of having failed at or before time t (i.e., having not survived up to time t) is the complement to the survival function. That is, the probability of failure is defined as

$$F_i(t) = P(T_i^* \leq t) = 1 - S_i(t),$$

which is equivalent to the definition of the cumulative distribution function (CDF) for the distribution of event times.

The probability integral transformation tells us that transforming a continuous random variable by its own CDF leads a new random variable that must follow a uniform distribution on the range 0 to 1 (Mood, Graybill, and Boes 1974). That is, $F_X(X) \sim U(0,1)$ where $F_X(.)$ denotes the CDF for the continuous random variable X. Similarly, a new random variable obtained by taking the complement of X transformed by its own CDF must also follow a uniform distribution on the range 0 to 1, that is, $1 - F_X(X) \sim U(0,1)$. Therefore, under an assumption that the event times T_i^* $(i = 1, \ldots, N)$ occur in continuous time, the survival probability for individual i at their true event time will be a uniform random variable on the range 0 to 1. That is,

$$S_i(T_i^*) = U_i \sim U(0,1).$$

It is then possible to extend these results to the setting of a proportional hazards model. Under a proportional hazards model the survival probability for individual i at their event time T_i^* can be written as

$$S_i(T_i^*) = \exp\left(-H_0(T_i^*)\exp(X_i^\top \beta)\right), \tag{2}$$

where $H_0(t) = \int_0^t h_0(s)ds$ is the cumulative baseline hazard evaluated at time t, and X_i is a vector of covariates with associated population-level (i.e., fixed effect) parameters β.

This is because the proportional hazards assumption also implies proportional cumulative hazards. Of course, using Equation 2, the survival probability $S_i(T_i^*)$ can be replaced by the uniform random variable U_i. Moreover, since the objective is to simulate a new event time for individual i ($i = 1, \ldots, N$), rather than to evaluate the survival probability at the known true event time, one can replace T_i^* with the simulated event time for individual i, T_i^s. This leads to

$$U_i = \exp\left(-H_0(T_i^s)\exp(X_i^\top \beta)\right). \tag{3}$$

To obtain the formula for the cumulative hazard inversion method, one simply needs to rearrange Equation 3 to solve for the event time. That is,

$$T_i^s = H_0^{-1}\left[-\log(U_i)\exp(-X_i^\top \beta)\right]. \tag{4}$$

2.2. Extending to complex data generating processes

If one can obtain an algebraic closed-form solution for the inverse cumulative baseline hazard, $H_0^{-1}(.)$, then a major benefit of the cumulative hazard inversion method is that it is simple and computationally efficient. Moreover, it can be used to generate event times for a variety of parametric baseline hazards, including standard choices such as the exponential, Weibull or Gompertz distributions. However, two potential hurdles can be encountered when applying the method to complex data generating processes. First, one may not be able to obtain a closed-form solution to the cumulative baseline hazard $H_0(t)$. Second, the cumulative baseline hazard may not be invertible.

Crowther and Lambert (2013) describe an extension to overcome these two difficulties. Their extension incorporates numerical root finding and/or numerical quadrature.

Root finding is used to numerically solve for T_i^s in situations where the cumulative baseline hazard function cannot be inverted analytically. A convenient choice of algorithm is Brent's univariate root finder (Brent 1973), which can effectively find a solution to the equation

$$S_i(T_i^s) - U_i = 0$$

by treating T_i^s as the single unknown. By default, the **simsurv** package uses Brent's univariate root finder as implemented in the `uniroot()` function from the base package **stats** (R Core Team 2020). However, as an alternative (e.g., if `uniroot` fails to converge), the user has the option to instead use the `dfsane()` function from the **BB** (Varadhan and Gilbert 2009) package.

To improve numerical stability, one may wish to solve the root finding equation after applying an appropriate transformation. By default, **simsurv** attempts to solve the root finding equation after applying a log transformation, that is

$$-H_i(T_i^s) - \log(U_i) = 0.$$

But another reasonable alternative might be a square root transformation, that is

$$(S_i(T_i^s))^{0.5} - (U_i)^{0.5} = 0.$$

The **simsurv** package allows the transformation function for the rooting equation to be specified by the user.

Quadrature is used to numerically evaluate the cumulative hazard function in settings where it does not have a tractable form. A standard choice of algorithm is Gauss-Kronrod quadrature, whereby the cumulative hazard $H_i(T_i^s) = \int_{s=0}^{T_i^s} h_i(s)ds$ can be approximated by

$$H_i(T_i^s) \approx \frac{T_i^s}{2} \sum_{q=1}^{Q} w_q h_i \left(\frac{T_i^s(1+z_q)}{2} \right),$$

where w_q and z_q are, respectively, the standardized weights and locations (abscissa) for the $q = 1, \ldots, Q$ quadrature nodes (Laurie 1997). When quadrature is required, the **simsurv** package uses $Q = 15$ nodes by default (but this can be altered by the user).

In some situations, when the form of the baseline hazard function is complex, both the root finding and quadrature steps may be required. The approach then involves nesting the numerical quadrature inside the root finding and iterating between the two until an appropriate solution to the root finding equation is obtained, under a tolerance, of say 1E-8.

2.3. Two-component mixture distributions

To provide greater flexibility in the specification of the baseline hazard function, the **simsurv** package allows event times to be generated from two-component mixture distributions. The two-component mixture distributions are additive on the survival scale, with a parameter $0 \leq \pi \leq 1$ defining the mixing proportions, that is

$$S_0(t) = \pi S_{01}(t) + (1 - \pi) S_{02}(t),$$

where $S_0(t)$ is the baseline survival function, $S_{01}(t)$ and $S_{02}(t)$ are baseline survival functions for the two component distributions. In **simsurv**, the component distributions can be specified as either exponential, Weibull, or Gompertz distributions. These choices of distribution lead to closed forms for the hazard, cumulative hazard, and survival functions and therefore numerical quadrature is not required (reducing computation time); these formulas are shown in Appendix B. However, the survival function is not analytically invertible under each of the two-component mixture distributions and therefore the cumulative hazard inversion method as specified in Equation 4 cannot be applied directly. Instead, numerical root finding must be used to solve for the event time.

The two-component mixture distributions allow for a wide range of shapes for the hazard function. Some example hazard functions for the two-component Weibull distribution are shown in Figure 1.

2.4. Time-dependent effects (i.e., non-proportional hazards)

The methods for simulating event times as described in Section 2.1 were based on a situation in which the effects of any covariates were incorporated under a proportional hazards assumption. However, the **simsurv** package also allows one to simulate event times conditional on covariates that have time-dependent effects (i.e., non-proportional hazards).

Consider a hazard function of the form

$$h_i(t) = h_0(t) \exp(X_{i1}^\top \beta_1 + X_{i2}^\top \beta_2 f(t)), \tag{5}$$

where X_{i1} is a vector of covariates for individual i with an associated vector of population-level (i.e., fixed effect) parameters β_1, and X_{i2} is a vector of covariates for individual i with

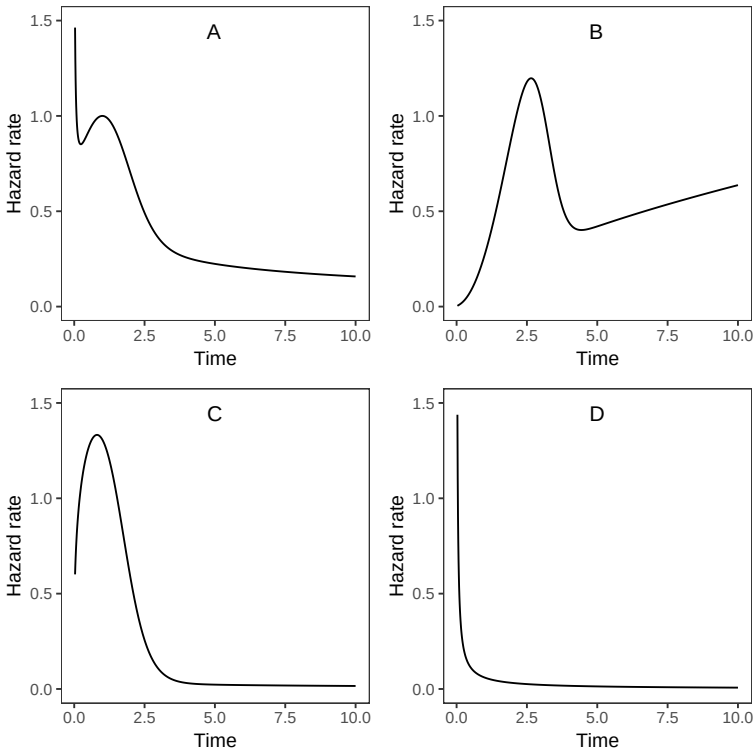

Figure 1: Examples of hazard functions that can be generated from a two-component Weibull distribution. The following parameter values were used for each panel. Panel A: $\lambda_1 = 1, \lambda_2 = 1, \gamma_1 = 1.5, \gamma_2 = 0.5, \pi = 0.5$. Panel B: $\lambda_1 = 0.1, \lambda_2 = 0.1, \gamma_1 = 3, \gamma_2 = 1.6, \pi = 0.8$. Panel C: $\lambda_1 = 1.4, \lambda_2 = 0.1, \gamma_1 = 1.3, \gamma_2 = 0.5, \pi = 0.9$. Panel D: $\lambda_1 = 1.5, \lambda_2 = 0.5, \gamma_1 = 0.2, \gamma_2 = 0.1, \pi = 0.1$

an associated vector of population-level parameters β_2 that are interacted with some scalar valued function of time, $f(t)$.

In the **simsurv** package, when time-dependent effects are specified (via the `tde` argument) the default is to use $f(t) = t$. That is, the time-dependent effects are formed by an interaction between the coefficient and linear time (on the log hazard scale). However, the user of **simsurv** can specify any scalar valued function for $f(.)$ (via the `tdefunction` argument) so long as it takes a single input value: the current time t. In Section 4.5 we demonstrate simulating event times using time-dependent effects generated through an interaction with log time.

Note that in many instances, $X_{i2}^\top$ will be a subset of $X_{i1}^\top$, but that this is not a requirement.

2.5. Incorporating cluster-specific parameters

A common approach to modeling clustered survival data is to introduce a so-called shared frailty term. The shared frailty term is effectively a cluster-specific random intercept introduced into the linear predictor of the proportional hazards model. Specifically, we can consider the following extension of the proportional hazards model from Equation 1:

$$h_{ij}(t) = h_0(t) \exp(X_{ij}^\top \beta + b_j),$$

where $h_{ij}(t)$ is the hazard of the event at time t, for individual i ($i = 1, \ldots, N_j$) in cluster

j ($j = 1, \ldots, J$), X_{ij} is a vector of covariates for individual i in cluster j with associated population-level parameters β (log hazard ratios), and b_j is a random effect (i.e., cluster-specific) parameter for cluster j capturing both the between-cluster variation, and within-cluster correlation, in the event times. An appropriate distribution must be specified for b_j, with common choices being a log-Gamma or normal distribution.

More generally, one can consider the extension to multiple random effect parameters

$$h_{ij}(t) = h_0(t) \exp(X_{ij}^\top \beta + Z_{ij}^\top b_j),$$

where Z_{ij} is a vector of covariates for individual i in cluster j with an associated vector of cluster-specific parameters b_j.

In Section 3 and Section 4.2, we describe how the `betas` argument can be used to simulate clustered event times by providing a data frame (rather than a vector) of parameters.

3. Implementation

The **simsurv** R package is implemented around one function, `simsurv()`. The function signature for `simsurv()` is

```
simsurv(dist = c("weibull", "exponential", "gompertz"), lambdas, gammas, x,
  betas, tde, tdefunction = NULL, mixture = FALSE, pmix = 0.5, hazard,
  loghazard, cumhazard, logcumhazard, idvar = NULL, ids = NULL,
  nodes = 15, maxt = NULL, interval = c(1e-08, 500),
  rootsolver = c("uniroot", "dfsane"), rootfun = log,
  seed = sample.int(.Machine$integer.max, 1), ...)
```

The first argument, `dist`, determines the parametric distribution for the event times when no user-defined hazard, log hazard, cumulative hazard, or log cumulative hazard function is provided. The standard parametric distributions that are available are the exponential, Weibull, and Gompertz distributions. If no covariates are specified, then event times are simulated using an analytical form for the inverted survival function. If time-fixed covariate(s) are specified in the data generating model, but no time-dependent effects, then the simulated event times are generated under a proportional hazards assumption (assuming the baseline hazard follows one of the three parametric distributions) and using an analytical form for the inverted survival function. Appendix A provides the analytical formulas for the inverted survival function under the assumption of an exponential, Weibull, or Gompertz baseline hazard. Where relevant, the true values for the shape and scale parameters for these parametric distributions need to be specified by the user via the `lambdas` and `gammas` arguments.

The user must provide covariate data via the `x` argument. This should be in the form of a data frame. At a minimum, this must include a variable with the subject IDs for each individual, since the number of rows in the `x` data frame is used to determine the number of event times to generate (assuming that `ids = NULL`). For instance, if one wishes to simulate event times for $N = 1000$ individuals from a model without any covariate effects then the argument could be specified as `x = data.frame(id = 1:1000)`, thereby providing information on the number of individuals to simulate event times for.

The `betas` argument is used to specify the true parameter values for any covariate effects used in the model. It can either be specified as `NULL` (i.e., left missing), a named vector, a

data frame, or a list of data frames. If `betas` is specified as a named vector then the names must coincide with named columns of the data frame of covariates. For example, if we specify `betas = c(treat = 0.5)` then the data frame of covariates specified in `x` must have a column named `"treat"` containing the covariate values for each individual and the true log hazard ratio associated with that covariate will be 0.5. Alternatively, if `betas` is specified as a data frame, then the column names in `betas` must coincide with corresponding column names in `x`. The benefit of specifying `betas` as a data frame is that the true parameter values in the data generating model can differ across individuals (or observational units), for example, if we have individual-specific (i.e., random effect) parameters in the model. An example of this situation is described in Section 4.2 where we simulate clustered event times. Lastly, one can specify `betas` as a list of data frames, however, this would only be necessary when providing a very complex user-defined hazard function via the `hazard` argument.

The `tde` and `tdefunction` arguments allow the easy specification of time-dependent effects (i.e., non-proportional hazards). The `tde` argument is specified in a similar way to the `betas` argument. However, instead of specifying the true parameter values for covariate main effects, it specifies the true parameter values for an interaction between the covariate and time. If the user wants an interaction with some function of time (instead of linear time), then this is specified in the `tdefunction` argument. An example of this functionality is demonstrated in Section 4.5. Alternatively, time-dependent effects can be accommodated in user-defined hazard functions (i.e., arguments `hazard`, `loghazard`, `cumhazard`, or `logcumhazard`). However, the `tde` and `tdefunction` arguments are more convenient for specifying time-dependent effects when using a model with one of the standard parametric baseline hazards provided in the package. Note that when time-dependent effects are specified there is no longer a closed-form solution to the cumulative hazard and, therefore, numerical quadrature and numerical root finding are used to simulate event times. This means that computation time will be slightly longer than when using a standard parametric proportional hazards model for which an analytical form for the inverted survival function is available.

The `mixture` and `pmix` arguments specify whether a two-component mixture distribution is used and the associated mixing parameter, respectively. If a two-component mixture distribution is used (i.e., `mixture = TRUE`) then the distribution for each component is determined by the `dist` argument. The two-component mixture model can be specified under a proportional hazards model formulation or alongside the `tde` argument to generate a model with non-proportional hazards.

The `hazard`, `loghazard`, `cumhazard`, and `logcumhazard` arguments allow for user-defined functions. At most, only one of these arguments can be specified. When a user-defined function is provided the `dist`, `lambdas`, `gammas`, `tde`, `tdefunction`, `mixture`, and `pmix` arguments are all ignored; all of these features need to be handled within the user-defined function, if they are required. The user-defined function must always have the following three arguments

- `t`: Scalar specifying the current time at which to evaluate the hazard.
- `x`: A named list with the covariate data.
- `betas`: A named list with the true parameter values.

Each of these arguments provide information that is used in evaluating the hazard $h_i(t)$, log hazard $\log h_i(t)$, cumulative hazard $H_i(t)$, or log cumulative hazard $\log H_i(t)$ (depending on

which type of user-specified function is being provided). These three arguments (`t`, `x`, `betas`) can then be followed in the function signature by any additional arguments that may be necessary. Examples of user-defined functions are provided in Section 4.3 and Section 4.6. Note that when a user-defined function is provided the simulated event times are generated using numerical root finding (for `cumhazard` and `logcumhazard`) or both numerical root finding and numerical integration (for `hazard` and `loghazard`).

The `idvar` and `ids` arguments are only likely to be required in two situations. First, when one wants to simulate using only a subset of the individuals in the covariate data frame passed to argument `x`. Second, when the covariate data frame passed to `x` or the parameter data frame(s) passed to `betas` contain multiple rows per individual. The latter situation will only be relevant when simulating event times from a user-defined hazard function with, for example, time-varying covariates in the data frame passed to `x`.

The `maxt` argument specifies the maximum follow up time; individuals with a simulated event time larger than `maxt` will be right-censored and their event indicator d_i will be set to 0. If `maxt = NULL` then there is no right-censoring of event times.

Several control arguments are also available for controlling the root finding or quadrature, when relevant. The `rootsolver` argument specifies the function that should be used for numerical root finding. Current options are `"uniroot"` or `"dfsane"`, as described in Section 2.2. The `rootfun` argument specifies the transformation function that should be applied to the root finding equation before attempting to solve it. This defaults to a log transformation, but other possibilities are described in Section 2.2. We only expect that the user would need to change the `rootsolver` or `rootfun` arguments from their default values in an extreme situation where the numerical root finding fails to converge. Lastly, the `nodes` argument specifies the number of nodes used in the Gauss-Kronrod quadrature.

4. Usage examples

In this section we demonstrate usage of **simsurv** through a series of examples. We replicate the examples from Crowther and Lambert (2012) and Crowther and Lambert (2013).

4.1. Simulating under a standard parametric survival model

This first example shows how the **simsurv** package can be used to generate event times under a Weibull proportional hazards model. The simulated event times will be generated under the following conditions:

- A monotonically increasing baseline hazard function, achieved by specifying a Weibull baseline hazard with a γ parameter of 1.5.
- The effect of a protective treatment obtained by specifying a binary covariate with log hazard ratio of -0.5.
- A maximum follow up time by censoring any individuals with a simulated event time larger than five years.

We will demonstrate simulating these event times as part of a simple simulation study. The objective of the simulation study will be to assess the bias and coverage probability (of the 95% confidence interval) for the estimated treatment effect. This will be achieved by:

- Generating 100 simulated datasets (ideally it should be more than 100 datasets, but we use less here for the sake of brevity), each containing $N = 200$ individuals.
- Fitting a Weibull proportional hazards model to each simulated dataset using the **flexsurv** package (Jackson 2016).
- Calculating, across the 100 simulated datasets, the mean bias of the estimated treatment effect.
- Calculating, across the 100 simulated datasets, the proportion of 95% confidence intervals containing the true treatment effect.

We first define a function that simulates one dataset, fits the analysis model, and then returns a bias and coverage indicator.

```
R> sim_run <- function() {
+    cov <- data.frame(id = 1:200, trt = rbinom(200, 1, 0.5))
+    dat <- simsurv(lambdas = 0.1, gammas = 1.5, betas = c(trt = -0.5),
+      x = cov, maxt = 5)
+    dat <- merge(cov, dat)
+
+    mod <- flexsurvspline(Surv(eventtime, status) ~ trt, data = dat)
+
+    est <- mod$coefficients[["trt"]]
+    ses <- sqrt(diag(mod$cov))[["trt"]]
+    cil <- est + qnorm(.025) * ses
+    ciu <- est + qnorm(.975) * ses
+
+    c(bias = est - (-0.5), coverage = ((-0.5 > cil) && (-0.5 < ciu)))
+  }
```

We then set a seed for reproducibility, perform 100 replicates in our simulation study, and present the mean bias and the estimate of the coverage probability for the treatment effect.

```
R> set.seed(908070)
R> rowMeans(replicate(100, sim_run()))

      bias   coverage
0.02842414 0.90000000
```

Here we see that there is very little bias in the estimates of the log hazard ratio for the treatment effect, and the 95% confidence intervals are near their intended level of coverage.

4.2. Simulating clustered event times

In this example we show how the **simsurv** package can be used to simulate clustered event times. Clustered event times can appear in a number of settings. One example is when event times are observed for patients who are clustered within clinics. In that setting, we may

wish to allow for the dependence (i.e., correlation) between event times observed for patients clustered within the same clinic. A second example is when performing a meta-analysis of individual participant survival data. In that setting we wish to allow for the dependence between the event times observed for individuals drawn from the same clinical trial. For this example, we focus on the latter; that is, we demonstrate how **simsurv** can be used to simulate event times from an individual participant data meta-analysis.

For our meta-analysis example we will assume that we are meta-analyzing $J = 50$ studies each containing $N_j = 200$ individuals. We assume that each study compared an active treatment to a control. It is assumed that there is a population-level (i.e., average) treatment effect, and that the $J = 50$ studies in our meta-analysis each have a study-specific treatment effect that deviates from the average treatment effect. That is, we assume the following data generating model

$$h_{ij}(t) = h_0(t) \exp(\text{treat}_{ij} \times (\beta + b_j)),$$

where treat_{ij} is a treatment indicator for individual i in study j, taking the value 1 if individual i was randomized to the active treatment and value 0 if they were randomized to the control. We will assume that each study assigned participants to the two treatment arms using a 1:1 randomization ratio. We will assume that the average treatment effect across all studies is $\beta = -0.5$ (i.e., a protective treatment) and that the study-specific deviations around this average treatment effect are drawn from a $b_j \sim N(0, 0.5)$ distribution. For the baseline hazard we will assume a Gompertz distribution with $\lambda = 0.1$ and $\gamma = 0.05$.

It is straightforward to simulate event times under this model using the `simsurv()` function. The pre-processing required is to generate a data frame with each row containing the covariate data (i.e., subject ID and treatment indicator) for one individual in the meta-analysis, and a separate data frame with each row containing the parameters to use in the data generating model for one individual in the meta-analysis. First, let us define the covariate data.

```
R> num_studies <- 50
R> num_patients <- 200
R> tot_patients <- num_studies * num_patients
R> cov <- data.frame(id = 1:tot_patients,
+    study = rep(1:num_studies, each = num_patients),
+    treat = rbinom(tot_patients, 1, 0.5))
```

Then we define the parameters for each individual.

```
R> pop_treat_effect <- -0.5
R> study_treat_effect <- pop_treat_effect + rnorm(num_studies, 0, 0.5)
R> pars <- data.frame(treat = rep(study_treat_effect, each = num_patients))
```

We can now simulate the event times using a relatively straightforward call to the `simsurv()` function.

```
R> set.seed(908070)
R> dat <- simsurv(dist = "gompertz", lambdas = 0.1, gammas = 0.05,
+    x = cov, betas = pars)
R> dat <- merge(cov, dat)
R> head(dat)
```

```
  id study treat   eventtime status
1  1     1     0 18.641158      1
2  2     1     1  7.616510      1
3  3     1     1  7.957204      1
4  4     1     1  0.049017      1
5  5     1     0  1.560224      1
6  6     1     1  4.655177      1
```

This demonstrates one of the advantages of `simsurv()`; the `betas` argument can be specified as a named vector if only population-level parameters are required or, alternatively, it can be specified as a data frame if cluster-specific or individual-specific parameters are required. If the latter approach is used then, by default, each row of the data frame specified in `betas` is assumed to contain the parameters corresponding to each row of the data frame of covariates specified in `x`. That is, `betas[i, ]` should extract the named parameter vector, and `x[i, ]` should extract the named covariate vector, used to generate the event time for the ith individual.

Lastly, let us fit a Cox proportional hazards model that coincides with the data generating model used to simulate our meta-analysis event times (noting that the baseline hazard is left unspecified in the analysis model, whereas we simulated the event times using a parametric Gompertz baseline hazard). We fit this model using the **coxme** package (Therneau 2020):

```
R> mod <- coxme(Surv(eventtime, status) ~ treat + (treat | study),
+    data = dat)
R> summary(mod)

Cox mixed-effects model fit by maximum likelihood
  Data: dat
  events, n = 10000, 10000
  Iterations= 19 82
                    NULL Integrated    Fitted
Log-likelihood -82108.93  -81640.74 -81541.63

                   Chisq    df p     AIC    BIC
Integrated loglik  936.38  2.00 0  932.38 917.96
 Penalized loglik 1134.60 47.62 0 1039.36 696.01

Model:  Surv(eventtime, status) ~ treat + (treat | study)
Fixed coefficients
            coef exp(coef)   se(coef)     z       p
treat -0.2755787 0.7591327 0.06642155 -4.15 3.3e-05

Random effects
 Group Variable Std Dev   Variance
 study treat    0.4471346 0.1999293
```

Here we see that we approximately recovered the standard deviation for the study-specific treatment effects, i.e., $\sigma_b = 0.5$.

4.3. Simulating under a user-defined log cumulative hazard function

Next, we will simulate event times under a slightly more complex parametric survival model. The data generating model will be based on the method of Royston and Parmar (2002); i.e., restricted cubic splines are used to approximate the log cumulative baseline hazard. This will require us to provide a user-defined log cumulative hazard function to `simsurv()`.

To motivate the need for this type of model, we will first fit

- a Weibull survival model, and
- a Royston and Parmar flexible parametric survival model

to a real dataset and compare the fit of each model. We will use the publicly accessible German breast cancer data. This dataset is included with the **simsurv** package (see `help(simsurv::brcancer)` for a description of the dataset). Let us look at the first few rows of the dataset.

```
R> data("brcancer", package = "simsurv")
R> head(brcancer)
```

```
  id hormon rectime censrec
1  1      0    1814       1
2  2      1    2018       1
3  3      1     712       1
4  4      1    1807       1
5  5      0     772       1
6  6      0     448       1
```

For fitting the flexible parametric model, we will use three internal knots (i.e., four degrees of freedom) for the restricted cubic splines with the knot points placed at evenly spaced percentiles of the distribution of observed event time (obtained by specifying the argument `k = 3` in the code below). This model can be estimated using the `flexsurvspline` function from the **flexsurv** package (Jackson 2016). We can also estimate the Weibull proportional hazards model using the `flexsurvspline` function from the **flexsurv** package, by specifying no internal knots (i.e., specifying `k = 0`).

```
R> mod_weib <- flexsurvspline(Surv(rectime, censrec) ~ hormon,
+    data = brcancer, k = 0)
R> mod_flex <- flexsurvspline(Surv(rectime, censrec) ~ hormon,
+    data = brcancer, k = 3)
```

Now let us compare the fit of the two models by plotting in Figure 2 each of the fitted survival functions on top of the Kaplan-Meier survival curve.

There is evidence in the plots that the flexible parametric model fits the data better than the standard Weibull model. Therefore, if we wanted to simulate event times from a data generating process similar to that of the breast cancer data, then using a Weibull distribution may not be adequate. Rather, it would be more appropriate to simulate event times under the flexible parametric model. We will demonstrate how the **simsurv** package can be used

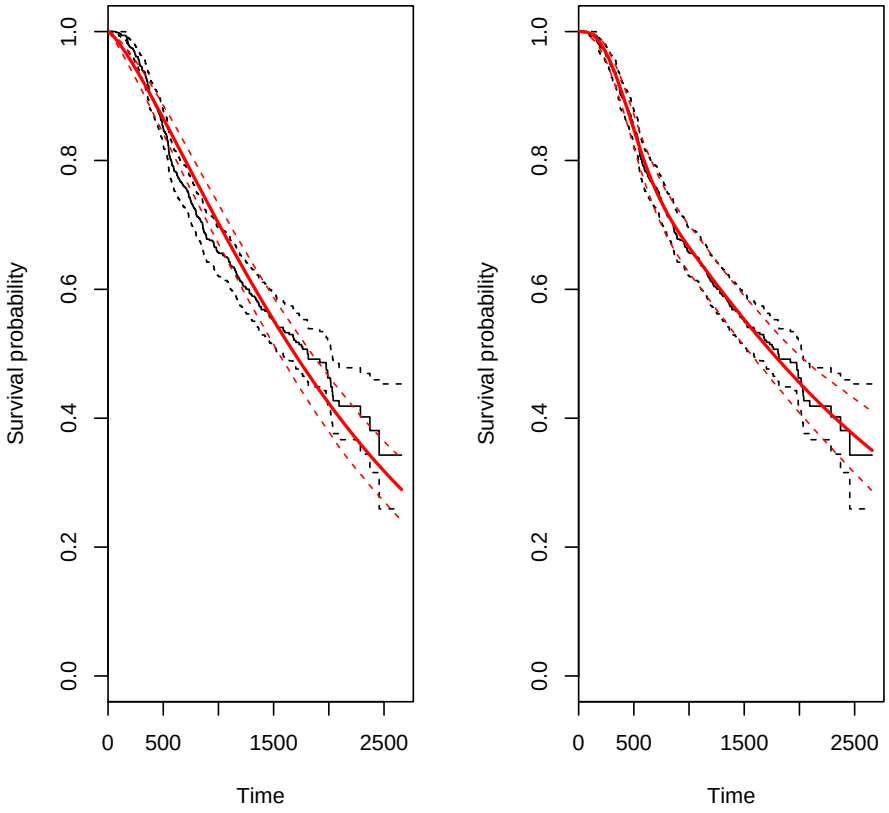

Figure 2: Predicted survival curve under Weibull and flexible parametric models (red), with overlaid Kaplan-Meier curve (black) together with the corresponding 95% confidence intervals.

to do this. The estimated parameters from the flexible parametric model will be used as the true parameters for the simulated event times.

The event times can be generated under a user-specified log cumulative hazard function that is equivalent to the Royston and Parmar specification used by the **flexsurv** package (Jackson 2016). First, the log cumulative hazard function for this model needs to be defined as a function in the R session. For example, in the function definition below, the first three compulsory arguments (`t`, `x`, `betas`) are followed by an additional argument `knots`, which allows the calculation of the log cumulative hazard at time t to depend on the knot locations for the splines.

```
R> logcumhaz <- function(t, x, betas, knots) {
+    basis <- flexsurv::basis(knots, log(t))
+    res <- betas[["gamma0"]] * basis[[1]] +
+      betas[["gamma1"]] * basis[[2]] + betas[["gamma2"]] * basis[[3]] +
+      betas[["gamma3"]] * basis[[4]] + betas[["gamma4"]] * basis[[5]] +
+      betas[["hormon"]] * x[["hormon"]]
+    res
+  }
```

Next, we will show how to use the `simsurv` function to simulate event times under the flexible

parametric model. To demonstrate this, we will again generate the event times as part of a simulation study. The objective of the simulation study will be to assess the bias in the estimated log hazard ratio for hormone therapy. This will be achieved by:

- Generating 100 simulated datasets (ideally it should be more than 100 datasets, but we use less here for the sake of brevity), each containing $N = 200$ individuals. The simulated event times will be generated under our flexible parametric model (with the "true" parameter values taken from fitting a model to the German breast cancer data).
- Fitting both a Weibull model and a flexible parametric model to each simulated dataset.
- Calculating, across the 100 simulated datasets, the mean bias in the log hazard ratio for hormone therapy under each of the Weibull and flexible parametric models.

First, we fit the flexible parametric model to the `brcancer` dataset to obtain the parameter estimates that we will use as the true parameter values in our data generating model:

```
R> true_mod <- flexsurvspline(Surv(rectime, censrec) ~ hormon,
+    data = brcancer, k = 3)
```

Then we define a function to generate one simulated dataset, fit our two analysis models (i.e., Weibull and flexible), and then return the bias in the estimated effect of hormone therapy under each fitted model.

```
R> sim_run <- function(true_mod) {
+    cov <- data.frame(id = 1:200, hormon = rbinom(200, 1, 0.5))
+    dat <- simsurv(betas = true_mod$coefficients, x = cov,
+      knots = true_mod$knots, logcumhazard = logcumhaz, maxt = NULL,
+      interval = c(1E-8, 100000))
+    dat <- merge(cov, dat)
+
+    weib_mod <- flexsurvspline(Surv(eventtime, status) ~ hormon,
+      data = dat, k = 0)
+    flex_mod <- flexsurvspline(Surv(eventtime, status) ~ hormon,
+      data = dat, k = 3)
+
+    true_loghr <- true_mod$coefficients[["hormon"]]
+    weib_loghr <- weib_mod$coefficients[["hormon"]]
+    flex_loghr <- flex_mod$coefficients[["hormon"]]
+
+    c(true_loghr = true_loghr, weib_bias = weib_loghr - true_loghr,
+      flex_bias = flex_loghr - true_loghr)
+  }
```

We then set a seed for reproducibility and generate 100 replicates in our simulation study.

```
R> set.seed(543543)
R> rowMeans(replicate(100, sim_run(true_mod = true_mod)))
```

```
  true_loghr    weib_bias    flex_bias
-0.364039328 -0.029244642 -0.008417815
```

4.4. Simulating under a user-defined piecewise hazard function

In this example we demonstrate how we can simulate under a user-defined piecewise constant hazard function. Specifically, we will use a piecewise constant hazard that resembles the "bathtub" curve. The bathtub hazard consists of a decreasing hazard at the lower end of the time frame, and an increasing hazard at the upper end of the time frame. This type of hazard is often exhibited by the failure rate of consumer products over their life cycle (i.e., high but decreasing failure rate early on due to defects, and an increasing failure rate later on due to wear and tear).

First, we use an exponential distribution (with rate parameter of 0.1) to generate a sequence of random cutpoints for our piecewise hazard function.

```
R> set.seed(1729)
R> ncuts <- 19
R> cuts <- sort(rexp(ncuts, rate = 0.1))
```

Then we use those cutpoints to define a vector containing the lower limits of the time intervals for our piecewise hazard function.

```
R> pw_times <- c(0, cuts)
```

Next, we generate the underlying "true" hazard within each time interval. We first generate an increasing piecewise constant hazard function by drawing variates from a standard half-normal distribution and sorting them.

```
R> N <- length(pw_times)
R> pw_haz <- sort(abs(rnorm(N)))
```

And we use those random variates to create a bathtub-shaped hazard function by centering them around their median and taking the absolute value.

```
R> pw_haz <- abs(pw_haz - median(pw_haz))
```

Figure 3 shows the piecewise constant hazard function we will use to simulate our event times.

Since we are planning to simulate our event times under a user-defined hazard function, the next step is to define a function in our R session that evaluates the hazard at any time t. As in our previous example, this function definition must have the first three arguments `t`, `x`, and `betas`, followed by any additional arguments needed to evaluate the hazard at time t. In this case, our additional arguments will be

- `lb_interval`: A vector with the lower limits for each the time interval in our piecewise hazard function.
- `haz_interval`: A vector containing the hazard rate within each time interval.

Therefore, our user-defined hazard function looks like

```
R> haz <- function(t, x, betas, lb_interval, haz_interval, ...) {
+    haz_interval[findInterval(t, lb_interval)]
+  }
```

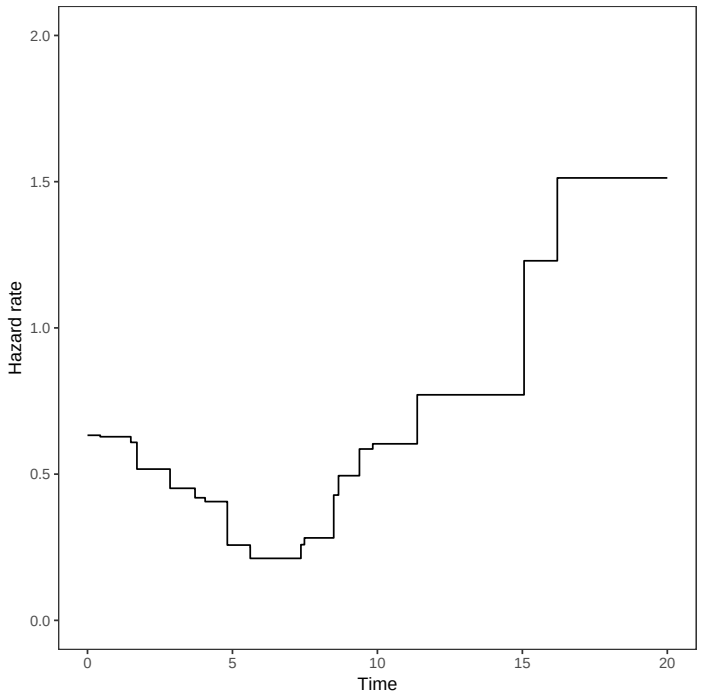

Figure 3: The piecewise constant hazard function we will use to simulate the event times.

Note that although the arguments `x` and `betas` are included in the function signature, they are not actually used within the function body since there are no covariate effects or additional parameters used in the calculation of the hazard rate.

We can now call the `simsurv` function to simulate $N = 5000$ event times under this model, as follows.

```
R> cov <- data.frame(id = 1:5000)
R> dat <- simsurv(x = cov, hazard = haz, lb_interval = pw_times,
+    haz_interval = pw_haz)
R> summary(dat$eventtime)
```

```
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
 0.000061   0.462052   1.107855   1.843625   2.320564 15.451911
```

4.5. Simulating under a Weibull model with time-dependent effects

This example shows how to simulate data under a standard Weibull survival model that incorporates a time-dependent effect (i.e., non-proportional hazards). For the time-dependent effect we will include a single binary covariate (e.g., a treatment indicator) with a protective effect (i.e., a negative log hazard ratio), but we will allow the effect of the covariate to diminish over time. The data generating model will be

$$h_i(t) = \gamma \lambda t^{\gamma-1} \exp(\beta_0 X_i + \beta_1 X_i \times \log(t)),$$

where X_i is the binary treatment indicator for individual i, λ and γ are the scale and shape parameters for the Weibull baseline hazard, β_0 is the log hazard ratio for treatment when $t = 1$ (i.e., when $\log(t) = 0$), and β_1 quantifies the amount by which the log hazard ratio for treatment changes for each one unit increase in $\log(t)$. Here we are assuming the time-dependent effect is induced by interacting the log hazard ratio with log time, but we could have used some other function of time (for example linear time, t, or time squared, t^2, if we had wanted to).

We will simulate data for $N = 10000$ individuals under this model, with a maximum follow up time of five years, and using the following true parameter values for the data generating model: $\beta_0 = -0.5$, $\beta_1 = 0.15$, $\lambda = 0.1$, and $\gamma = 1.5$. The code required to generate the simulated event times is as follows.

```
R> set.seed(9898)
R> cov <- data.frame(id = 1:10000, trt = rbinom(10000, 1, 0.5))
R> dat <- simsurv(dist = "weibull", lambdas = 0.1, gammas = 1.5,
+    betas = c(trt = -0.5), x = cov, tde = c(trt = 0.15),
+    tdefunction = "log", maxt = 5)
R> dat <- merge(cov, dat)
R> head(dat)

  id trt eventtime status
1  1   1  3.838829      1
2  2   0  5.000000      0
3  3   0  1.159572      1
4  4   1  5.000000      0
5  5   0  5.000000      0
6  6   1  5.000000      0
```

Then let us fit a flexible parametric model with two internal knots (i.e., 3 degrees of freedom) for the baseline hazard, and a time-dependent hazard ratio for the treatment effect. For the time-dependent hazard ratio we will use an interaction with log time (the same as used in the data generating model); this can be easily achieved using the `stpm2` function from the **rstpm2** package Clements and Liu (2019) and specifying the `tvc` option. Note that the **rstpm2** package and **flexsurv** packages can both be used to fit the Royston and Parmar flexible parametric survival model, however, they differ slightly in their post-estimation functionality and other possible extensions. Here, we use the **rstpm2** package because it allows us to easily specify time-dependent effects and then plot the time-dependent hazard ratio after fitting the model (as shown in the code below).

The model with the time-dependent effect for treatment can be estimated using the following code.

```
R> mod_tvc <- stpm2(Surv(eventtime, status) ~ trt, data = dat,
+    tvc = list(trt = 1))
```

And for comparison we can fit the corresponding model, but without the time-dependent effect for treatment (i.e., assuming proportional hazards instead).

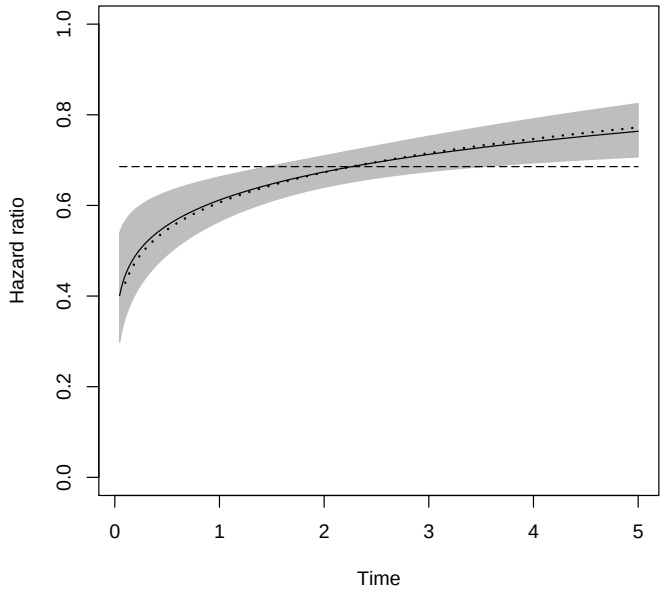

Figure 4: Time-dependent hazard ratio for the estimated treatment effect (solid line) with 95% confidence bounds (shaded region), the time-fixed hazard ratio for the estimated treatment effect (dashed line), and the hazard ratio for the true treatment effect under the data generating model (dotted line).

```
R> mod_ph <- stpm2(Surv(eventtime, status) ~ trt, data = dat)
```

Figure 4 shows, on the same plot region, the true time-dependent hazard ratio, the estimated time-dependent hazard ratio, and the estimated time-fixed hazard ratio. We can see the diminishing effect of treatment under the model with the time-dependent hazard ratio; as time increases the hazard ratio approaches a value of 1. Moreover, note that the estimated mean for the time-dependent hazard ratio is approximately equal to the true time-dependent hazard ratio under the data generating model.

4.6. Simulating under a joint model for longitudinal and survival data

This example shows how the **simsurv** package can be used to simulate event times under a shared parameter joint model for longitudinal and survival data. This is therefore also an example of simulating event times with a time-varying covariate.

We will simulate event times according to the following model formulation for the longitudinal submodel

$$
\begin{aligned}
Y_i(t) &= \mu_i(t) + \epsilon_i(t) \\
\mu_i(t) &= \beta_{0i} + \beta_{1i}t + \beta_2 x_{1i} + \beta_3 x_{2i} \\
\beta_{0i} &= \beta_{00} + b_{0i} \\
\beta_{1i} &= \beta_{10} + b_{1i} \\
(b_{0i}, b_{1i})^\top &\sim N(0, \Sigma) \\
\epsilon_i(t) &\sim N(0, \sigma_y^2)
\end{aligned}
$$

and the event submodel

$$h_i(t) = \delta t^{\delta-1} \exp(\gamma_0 + \gamma_1 x_{1i} + \gamma_2 x_{2i} + \alpha \mu_i(t)),$$

where x_{1i} is an indicator variable for a binary covariate, x_{2i} is a continuous covariate, b_{0i} and b_{1i} are individual-level parameters (i.e., random effects) for the intercept and slope for individual i, the β and γ terms are population-level parameters (i.e., fixed effects), δ is the shape parameter for the Weibull baseline hazard, and the residual errors $\epsilon_i(t)$ are assumed to be uncorrelated with b_{0i} and b_{1i}.

This specification allows for an individual-specific linear trajectory for the longitudinal submodel, a Weibull baseline hazard in the event submodel, a current value association structure for the time-dependent covariate, and the effects of a binary and a continuous time-fixed covariate in both the longitudinal and event submodels.

To simulate from this model using **simsurv**, we need to first explicitly define the hazard function. The code defining a function that returns the hazard for this joint model is

```
R> haz <- function(t, x, betas, ...) {
+    betas[["delta"]] * (t ^ (betas[["delta"]] - 1)) * exp(
+      betas[["gamma_0"]] + betas[["gamma_1"]] * x[["x1"]] +
+      betas[["gamma_2"]] * x[["x2"]] + betas[["alpha"]] * (
+        betas[["beta_0i"]] + betas[["beta_1i"]] * t +
+        betas[["beta_2"]]  * x[["x1"]] +
+        betas[["beta_3"]]  * x[["x2"]]))
+  }
```

The next step is to define the true parameter values and covariate data for each individual. This is achieved by specifying two data frames: one for the parameter values, and one for the covariate data. Each row of the data frame will correspond to a different individual. The code to achieve this is

```
R> set.seed(5454)
R> N <- 200
R> betas <- data.frame(delta = rep(2, N), gamma_0 = rep(-11.9, N),
+    gamma_1 = rep(0.6, N), gamma_2 = rep(0.08, N), alpha = rep(0.03, N),
+    beta_0 = rep(90, N), beta_1 = rep(2.5, N), beta_2 = rep(-1.5, N),
+    beta_3 = rep(1, N))
R> b_corrmat <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
R> b_sds <- c(20, 3)
R> b_means <- rep(0, 2)
R> b_z <- MASS::mvrnorm(n = N, mu = b_means, Sigma = b_corrmat)
R> b <- sapply(1:length(b_sds), FUN = function(x) b_sds[x] * b_z[,x])
R> betas$beta_0i <- betas$beta_0 + b[, 1]
R> betas$beta_1i <- betas$beta_1 + b[, 2]
R> covdat <- data.frame(x1 = stats::rbinom(N, 1, 0.45),
+    x2 = stats::rnorm(N, 44, 8.5))
```

The final step is to then generate the simulated event times using a call to the `simsurv` function. The only arguments that need to be specified are the user-defined hazard function,

the true parameter values, and the covariate data. In this example we will also specify a maximum follow up time of ten units (for example, ten years, after which individuals will be censored if they have not yet experienced the event). The code to generate the simulated event times is

```
R> times <- simsurv(hazard = haz, x = covdat, betas = betas, maxt = 10)
```

We can then examine the first few rows of the resulting data frame, to see the simulated event times and event indicator.

```
R> head(times)
```

```
  id eventtime status
1  1  1.549822      1
2  2  6.023793      1
3  3 10.000000      0
4  4  3.313708      1
5  5  8.317876      1
6  6  7.385959      1
```

Of course, we have only simulated the event times here; we haven't simulated any observed values for the longitudinal outcome. Moreover, although the **simsurv** package can be used for simulating joint longitudinal and time-to-event data, it did take a bit of work and several lines of code to achieve. Therefore, it is worth noting that the **simjm** package (Brilleman 2018), which acts as a wrapper for the **simsurv** package, is designed specifically for this purpose. It can make the process simpler for the end-user, since it shields them from much of the work described in this example. Instead, the user can simulate joint longitudinal and time-to-event data using one function call to `simjm()` and a number of optional arguments are available to alter the exact specification of the shared parameter joint model.

5. Summary

In this article we have introduced the **simsurv** R package. The package facilitates the simulation of survival (i.e., time-to-event) data under any data generating model for which the hazard, log hazard, cumulative hazard, or log cumulative hazard function can be written down. The package is implemented as one function; `simsurv()`. The arguments to `simsurv()` allow straightforward simulation of event times under standard parametric distributions or two-component mixture distributions. Moreover, covariate effects can be introduced under proportional or non-proportional hazards assumptions. Additional scenarios, for example, time-varying covariates or complex baseline hazards are accommodated through user-defined hazard functions. Clustered event times are easily generated through the ability to specify individual-specific or cluster-specific coefficients. Informative censoring times (e.g., that depend on individual-level covariates) can also be easily generated by simulating an event distribution, simulating a censoring distribution, and taking the minimum.

In future work, we will add additional arguments to the `simsurv()` function that will facilitate the simulation of multiple (i.e., competing) events and recurrent events. Lastly, we will add arguments to more easily allow the specification of an informative censoring process.

Acknowledgments

SLB is funded by an Australian National Health and Medical Research Council (NHMRC) Postgraduate Scholarship (ref: APP1093145), with additional support from an NHMRC Centre of Research Excellence grant (ref: 1035261) awarded to the Victorian Centre for Biostatistics (ViCBiostat). MJC is partly funded by a UK Medical Research Council (MRC) New Investigator Research Grant (ref: MR/P015433/1).

References

Abrahamowicz M, MacKenzie T, Esdaile JM (1996). "Time-Dependent Hazard Ratio: Modelling and Hypothesis Testing with Application in Lupus Nephritis." *Journal of the American Statistical Association*, **91**(436), 1432–1439. doi:10.1080/01621459.1996.10476711.

Bender R, Augustin T, Blettner M (2005). "Generating Survival Times to Simulate Cox Proportional Hazards Models." *Statistics in Medicine*, **24**(11), 1713–1723. doi:10.1002/sim.2059.

Brent R (1973). *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs.

Brilleman S (2018). ***simjm**: Simulate Joint Longitudinal and Survival Data*. R package version 0.0.1, URL https://github.com/sambrilleman/simjm.

Brilleman S (2021). ***simsurv**: Simulate Survival Data*. R package version 1.0.0, URL https://CRAN.R-project.org/package=simsurv.

Chapple AG (2016). ***SimSCRPiecewise**: Simulates Univariate and Semi-Competing Risks Data Given Covariates and Piecewise Exponential Baseline Hazards*. R package version 0.1.1, URL https://CRAN.R-project.org/package=SimSCRPiecewise.

Clements M, Liu XR (2019). ***rstpm2**: Generalized Survival Models*. R package version 1.5.1, URL https://CRAN.R-project.org/package=rstpm2.

Crowther MJ, Lambert PC (2012). "Simulating Complex Survival Data." *The Stata Journal*, **12**(4), 674–687. doi:10.1177/1536867x1201200407.

Crowther MJ, Lambert PC (2013). "Simulating Biologically Plausible Complex Survival Data." *Statistics in Medicine*, **32**(23), 4118–4134. doi:10.1002/sim.5823.

Gerds TA (2019). ***prodlim**: Product-Limit Estimation for Censored Event History Analysis*. R package version 2019.11.13, URL https://CRAN.R-project.org/package=prodlim.

Henderson R, Diggle P, Dobson A (2000). "Joint Modelling of Longitudinal Measurements and Event Time Data." *Biostatistics*, **1**(4), 465–480. doi:10.1093/biostatistics/1.4.465.

Hothorn T (2020). "Most Likely Transformations: The **mlt** Package." *Journal of Statistical Software*, **92**(1), 1–68. doi:10.18637/jss.v092.i01.

Jackson C (2016). "**flexsurv**: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Laurie DP (1997). "Calculation of Gauss-Kronrod Quadrature Rules." *Mathematics of Computation*, **66**(219), 1133–1146. doi:10.1090/s0025-5718-97-00861-2.

Leemis LM (1987). "Variate Generation for Accelerated Life and Proportional Hazards Models." *Operations Research*, **35**(6), 892–894. doi:10.1287/opre.35.6.892.

Mood AM, Graybill FA, Boes DC (1974). *Introduction to the Theory of Statistics*. McGraw-Hill, New York.

Moriña D, Navarro A (2014). "The R Package **survsim** for the Simulation of Simple and Complex Survival Data." *Journal of Statistical Software*, **59**(2), 1–20. doi:10.18637/jss.v059.i02.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Royston P, Parmar MK (2002). "Flexible Parametric Proportional-Hazards and Proportional-Odds Models for Censored Survival Data, with Application to Prognostic Modelling and Estimation of Treatment Effects." *Statistics in Medicine*, **21**(15), 2175–2197. doi:10.1002/sim.1203.

StataCorp (2015). *Stata Data Analysis Statistical Software: Release 14*. StataCorp LP, College Station. URL http://www.stata.com/.

Sylvestre MP, Abrahamowicz M (2008). "Comparison of Algorithms to Generate Event Times Conditional on Time-Dependent Covariates." *Statistics in Medicine*, **27**(14), 2618–2634. doi:10.1002/sim.3092.

Sylvestre MP, Evans T, MacKenzie T, Abrahamowicz M (2017). ***PermAlgo**: Permutational Algorithm to Generate Event Times Conditional on a Covariate Matrix Including Time-Dependent Covariates*. R package version 1.1, URL https://CRAN.R-project.org/package=PermAlgo.

Therneau TM (2020). ***coxme**: Mixed Effects Cox Models*. R package version 2.2-16, URL https://CRAN.R-project.org/package=coxme.

Tsiatis AA, Davidian M (2004). "Joint Modeling of Longitudinal and Time-to-Event Data: An Overview." *Statistica Sinica*, **14**(3), 809–834.

Varadhan R, Gilbert P (2009). "**BB**: An R Package for Solving a Large System of Nonlinear Equations and for Optimizing a High-Dimensional Nonlinear Objective Function." *Journal of Statistical Software*, **32**(4), 1–26. doi:10.18637/jss.v032.i04.

Xiong D, Pokaprakarn T, Udagawa H, Rabbee N (2015). ***SimHaz**: Simulated Survival and Hazard Analysis for Time-Dependent Exposure*. R package version 0.1, URL https://CRAN.R-project.org/package=SimHaz.

A. Parameterizations for standard parametric distributions

The following shows the analytical forms for the hazard function $h_i(t)$, cumulative hazard function $H_i(t)$, survival function $S_i(t)$, and inverted survival function $S_i^{-1}(u)$ for each of the standard parametric distributions under the assumption of proportional hazards.

For the exponential distribution we have the following:

$$\begin{aligned} h_i(t) &= \lambda \exp(X_i^\top \beta) \\ H_i(t) &= \lambda t \exp(X_i^\top \beta) \\ S_i(t) &= \exp\left(-\lambda t \exp(X_i^\top \beta)\right) \\ S_i^{-1}(u) &= \left(\frac{-\log(u)}{\lambda \exp(X_i^\top \beta)}\right) \end{aligned}$$

where $\lambda > 0$ is the rate parameter.

For the Weibull distribution we have the following:

$$\begin{aligned} h_i(t) &= \gamma\lambda(t^{\gamma-1}) \exp(X_i^\top \beta) \\ H_i(t) &= \lambda(t^\gamma) \exp(X_i^\top \beta) \\ S_i(t) &= \exp\left(-\lambda(t^\gamma) \exp(X_i^\top \beta)\right) \\ S_i^{-1}(u) &= \left(\frac{-\log(u)}{\lambda \exp(X_i^\top \beta)}\right)^{1/\gamma} \end{aligned}$$

where $\lambda > 0$ and $\gamma > 0$ are the scale and shape parameters, respectively.

For the Gompertz distribution we have the following:

$$\begin{aligned} h_i(t) &= \lambda \exp(\gamma t) \exp(X_i^\top \beta) \\ H_i(t) &= \frac{\lambda(\exp(\gamma t) - 1)}{\gamma} \exp(X_i^\top \beta) \\ S_i(t) &= \exp\left(\frac{-\lambda(\exp(\gamma t) - 1)}{\gamma} \exp(X_i^\top \beta)\right) \\ S_i^{-1}(u) &= \frac{1}{\gamma} \log\left[\left(\frac{-\gamma \log(u)}{\lambda \exp(X_i^\top \beta)}\right) + 1\right] \end{aligned}$$

where $\lambda > 0$ and $\gamma > 0$ are the shape and scale parameters, respectively.

B. Parameterizations for two-component mixture distributions

The following shows the analytical forms for the hazard function $h_i(t)$, cumulative hazard function $H_i(t)$, survival function $S_i(t)$, and inverted survival function $S_i^{-1}(u)$ for each of the two-component mixture distributions under the assumption of proportional hazards. The two-component mixture distributions are additive on the survival scale, with a parameter $0 \leq \pi \leq 1$ defining the mixing proportions, i.e.,

$$S_0(t) = \pi S_{01}(t) + (1 - \pi) S_{02}(t),$$

where $S_0(t)$ is the baseline survival function, and $S_{01}(t)$ and $S_{02}(t)$ are baseline survival functions for the two component distributions. The following shows the analytical forms for the hazard function $h_i(t)$, cumulative hazard function $H_i(t)$, and survival function $S_i(t)$ for each of the two-component mixture distributions under the assumption of proportional hazards.

For the two-component exponential mixture distribution we have the following:

$$
\begin{aligned}
h_i(t) &= \left[\frac{\pi\lambda_1 \exp(-\lambda_1 t) + (1-\pi)\lambda_2 \exp(-\lambda_2 t)}{\pi \exp(-\lambda_1 t) + (1-\pi)\exp(-\lambda_2 t)}\right] \exp(X_i^\top \beta) \\
H_i(t) &= -\log\left[\pi \exp(-\lambda_1 t) + (1-\pi)\exp(-\lambda_2 t)\right] \exp(X_i^\top \beta) \\
S_i(t) &= \left[\pi \exp(-\lambda_1 t) + (1-\pi)\exp(-\lambda_2 t)\right]^{\exp(X_i^\top \beta)}
\end{aligned}
$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are the rate parameters for the component exponential distributions.

For the two-component Weibull mixture distribution we have the following:

$$
\begin{aligned}
h_i(t) &= \left[\frac{\pi\gamma_1\lambda_1(t^{\gamma_1-1}) \exp(-\lambda_1(t^{\gamma_1})) + (1-\pi)\gamma_2\lambda_2(t^{\gamma_2-1}) \exp(-\lambda_2(t^{\gamma_2}))}{\pi \exp(-\lambda_1(t^{\gamma_1})) + (1-\pi)\exp(-\lambda_2(t^{\gamma_2}))}\right] \exp(X_i^\top \beta) \\
H_i(t) &= -\log\left[\pi \exp(-\lambda_1(t^{\gamma_1})) + (1-\pi)\exp(-\lambda_2(t^{\gamma_2}))\right] \exp(X_i^\top \beta) \\
S_i(t) &= \left[\pi \exp(-\lambda_1(t^{\gamma_1})) + (1-\pi)\exp(-\lambda_2(t^{\gamma_2}))\right]^{\exp(X_i^\top \beta)}
\end{aligned}
$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are the scale parameters, and $\gamma_1 > 0$ and $\gamma_2 > 0$ are the shape parameters, for the component Weibull distributions.

For the two-component Gompertz mixture distribution we have the following:

$$
\begin{aligned}
h_i(t) &= \left[\frac{\pi\lambda_1 \exp(\gamma_1 t) \exp\left(\frac{-\lambda_1(\exp(\gamma_1 t)-1)}{\gamma_1}\right) + (1-\pi)\lambda_2 \exp(\gamma_2 t) \exp\left(\frac{-\lambda_2(\exp(\gamma_2 t)-1)}{\gamma_2}\right)}{\pi \exp\left(\frac{-\lambda_1(\exp(\gamma_1 t)-1)}{\gamma_1}\right) + (1-\pi)\exp\left(\frac{-\lambda_2(\exp(\gamma_2 t)-1)}{\gamma_2}\right)}\right] \\
&\qquad\qquad \exp(X_i^\top \beta) \\
H_i(t) &= -\log\left[\pi \exp\left(\frac{-\lambda_1(\exp(\gamma_1 t)-1)}{\gamma_1}\right) + (1-\pi)\exp\left(\frac{-\lambda_2(\exp(\gamma_2 t)-1)}{\gamma_2}\right)\right] \exp(X_i^\top \beta) \\
S_i(t) &= \left[\pi \exp\left(\frac{-\lambda_1(\exp(\gamma_1 t)-1)}{\gamma_1}\right) + (1-\pi)\exp\left(\frac{-\lambda_2(\exp(\gamma_2 t)-1)}{\gamma_2}\right)\right]^{\exp(X_i^\top \beta)}
\end{aligned}
$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are the shape parameters, and $\gamma_1 > 0$ and $\gamma_2 > 0$ are the scale parameters, for the component Gompertz distributions.

Affiliation:

Samuel L. Brilleman
School of Public Health and Preventive Medicine
Monash University
553 St Kilda Road, Melbourne
Victoria 3004, Australia
E-mail: sam.brilleman@monash.edu
URL: http://www.sambrilleman.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2021, Volume 97, Issue 3 *Submitted:* 2018-06-05
doi:10.18637/jss.v097.i03 *Accepted:* 2019-11-10