Package 'simsurv'

January 13, 2021

Type Package

Title Simulate Survival Data

Version 1.0.0

Date 2021-01-09

Maintainer Sam Brilleman <sam.brilleman@gmail.com>

Description Simulate survival times from standard parametric survival distributions (exponential, Weibull, Gompertz), 2-component mixture distributions, or a user-defined hazard, log hazard, cumulative hazard, or log cumulative hazard function. Baseline covariates can be included under a proportional hazards assumption.
Time dependent effects (i.e. non-proportional hazards) can be included by interacting covariates with linear time or a user-defined function of time.
Clustered event times are also accommodated.
The 2-component mixture distributions can allow for a variety of flexible baseline hazard functions reflecting those seen in practice.
If the user wishes to provide a user-defined
hazard or log hazard function then this is possible, and the resulting cumulative hazard function does not need to have a closed-form solution.
For details see the supporting paper <doi:10.18637/jss.v097.i03>.
Note that this package is modelled on the 'survsim' package available in the 'Stata' software (see Crowther and Lambert (2012) <https://www.stata-journal.com/sjpdf.html?articlenum=st0275> or Crowther and Lambert (2013) <doi:10.1002/sim.5823>).

License GPL (>= 3) | file LICENSE

Depends R (>= 3.3.0)

Imports methods, stats

Suggests BB (>= 2014.10.1), eha (>= 2.4.5), flexsurv (>= 1.1.0), knitr (>= 1.15.1), MASS, rmarkdown, rstpm2 (>= 1.4.1), survival (>= 2.40.1), testthat (>= 1.0.2)

VignetteBuilder knitr

LazyData true

BugReports https://github.com/sambrilleman/simsurv/issues

RoxygenNote 7.1.1

NeedsCompilation no

Author Sam Brilleman [cre, aut, cph],
Alessandro Gasparini [ctb]

Repository CRAN

Date/Publication 2021-01-13 10:50:11 UTC

R topics documented:

`brcancer` *German breast cancer dataset*

Description

A data set containing a subset of variables from the German breast cancer study (Schumacher et al. (1994)).

Usage

```
brcancer
```

Format

A data frame with 686 rows and 4 variables.

Details

- `id` Subject ID
- `hormon` Treatment indicator for hormone therapy (0 = no, 1 = yes)
- `rectime` Time to recurrence or censoring (in days)
- `censrec` Event indicator (0 = censored, 1 = recurrence)

References

Sauerbrei W, Royston P. Building multivariable prognostic and diagnostic models: transformation of the predictors by using fractional polynomials. *Journal of the Royal Statistics Society Series A*. 1999;**162**(1):71–94.

Schumacher M, et al. for the German Breast Cancer Study Group. Randomized 2x2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*. 1994;**12**:2086–2093.

simsurv *Simulate survival data*

Description

Simulate survival times from standard parametric survival distributions, 2-component mixture distributions, or a user-defined hazard or log hazard function.

Usage

```
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),
  ...
)
```

Arguments

dist	Character string specifying the parametric survival distribution. Can be "weibull" (the default), "exponential", or "gompertz". Note that this is ignored if a user-defined baseline hazard function is provided via the hazard or loghazard argument.
lambdas	A numeric vector corresponding to the scale parameters for the exponential, Weibull or Gompertz distributions. This vector should be length one for any of the standard parametric distributions, or length two for a 2-component mixture distribution when mixture = TRUE.

gammas — A numeric vector corresponding to the shape parameters for the Weibull or Gompertz distributions. This vector should be length one for the standard Weibull or Gompertz distributions, or length two for a 2-component mixture distribution when `mixture = TRUE`.

x — A data frame containing the covariate values for each individual. Each row of the data frame should supply covariate data for one individual. The column names should correspond to the named elements in `betas`. If no covariates are being used to simulate the event times, then this data frame should just include an ID value for each individual; this is necessary since the number of individuals to simulate event times for is taken to be the number of rows in `x` (unless `idvar` and `ids` are specified, in which case the number of individuals to simulate event times for is taken to be `length(ids)`).

betas — A named vector, a data frame, or a list of data frames, containing the "true" parameters (e.g. log hazard ratios). If a standard exponential, Weibull, or Gompertz distribution, or a 2-component mixture distribution is being used then `betas` should provide the log hazard ratios for each of the baseline covariates that is to be included in the linear predictor of the proportional hazards model. This is most easily specified as a named vector (see the **Examples**). Alternatively, if a user-defined baseline hazard (or log hazard) function is provided via the `hazard` (or `loghazard`) argument, then `betas` can be specified as a vector, a data frame, or a list of data frames, and the of user-defined hazard (or log hazard) function should extract named elements `betas` however necessary in order to calculate the hazard (or log hazard) for each individual. See the **Examples**.

tde — A named vector, containing the "true" parameters that will be used to create time dependent effects (i.e. non-proportional hazards). The values specified in `tde` are used as coefficients (in the linear predictor of the proportional hazards model) on an interaction term between the corresponding covariate and time (or some function of time, for example log time, if `tdefunction` is not `NULL`).

tdefunction — An optional function of time to which covariates specified in `tde` will be interacted, in order to generate time dependent effects (i.e. non-proportional hazards). If `NULL` then the covariates specified in `tde` will be interacted with linear time. This can be the character string corresponding to a standard function (such as "log") or it can be a user-defined function with a single argument (for example `function(x) x ^ 2`).

mixture — Logical specifying whether to use a 2-component mixture model for the survival distribution. If `TRUE`, then the distribution of the mixture components is determined by the `dist` argument.

pmix — Scalar between 0 and 1 defining the mixing parameter when `mixture = TRUE`. The baseline survival at time t is taken to be $S(t) = pS_1(t) + (1-p)S_2(t)$ where $S_1(t)$ and $S_2(t)$ are the baseline survival under each component of the mixture distribution.

hazard — Optionally, a user-defined hazard function, with arguments `t`, `x`, and `betas`. This function should return the hazard at time `t` for an individual with covariates supplied via `x` and parameters supplied via `betas`. See the **Examples**.

loghazard — Optionally, a user-defined log hazard function, with arguments `t`, `x`, and `betas`. This function should return the log hazard at time `t` for an individual with covariates supplied via `x` and parameters supplied via `betas`. See the **Examples**.

`cumhazard` Optionally, a user-defined cumulative hazard function, with arguments `t`, `x`, and `betas`. This function should return the cumulative hazard at time `t` for an individual with covariates supplied via `x` and parameters supplied via `betas`. See the **Examples**.

`logcumhazard` Optionally, a user-defined log cumulative hazard function, with arguments `t`, `x`, and `betas`. This function should return the log cumulative hazard at time `t` for an individual with covariates supplied via `x` and parameters supplied via `betas`. See the **Examples**.

`idvar` The name of the ID variable identifying individuals. This is only required when `x` (and `betas` if it is supplied as a data frame) contains multiple rows per individual. Otherwise, if `idvar = NULL` then each row of `x` (and `betas` if it is supplied as a data frame) is assumed to correspond to a different individual.

`ids` A vector containing the unique values of `idvar` (i.e. the unique individual IDs). This is only required when `x` (and `betas` if it is supplied as a data frame) contain multiple rows per individual. Otherwise, if `idvar = NULL` then each row of `x` (and `betas` if it is supplied as a data frame) is assumed to correspond to a different individual.

`nodes` Integer specifying the number of quadrature nodes to use for the Gauss-Kronrod quadrature. Can be 7, 11, or 15.

`maxt` The maximum event time. For simulated event times greater than `maxt`, the event time (`"eventtime"`) returned in the data frame will be truncated at `maxt` and the event indicator (`"status"`) will be set to zero indicating that the individual was right-censored.

`interval` The interval over which to search for the `uniroot` corresponding to each simulated event time.

`rootsolver` Character string specifying the function to use for univariate root finding when required. This can currently be `"uniroot"` for using the `uniroot` function, or `"dfsane"` for using the `dfsane` function.

`rootfun` A function to apply to each side of the root finding equation when numerical root finding is used to solve for the simulated event time. An appropriate function helps to improve numerical stability. The default is to use a log transformation; that is, to solve $log(S(T)) - log(U) = 0$ where $S(T)$ is the survival probability at the event time and U is a uniform random variate. Suitable alternatives might be to specify `rootfun = NULL`, which corresponds to $S(T) - U = 0$, or `rootfun = sqrt`, which corresponds to $sqrt(S(T)) - sqrt(U) = 0$. It is unexpected that the user should need to change this argument from its default value, except perhaps in the extreme case that the numerical root finding fails.

`seed` The `seed` to use.

`...` Other arguments passed to `hazard`, `loghazard`, `cumhazard`, or `logcumhazard`.

Details

The `simsurv` function simulates survival times from standard parametric survival distributions (exponential, Weibull, Gompertz), 2-component mixture distributions, or a user-defined hazard or log hazard function. Baseline covariates can be included under a proportional hazards assumption. Time dependent effects (i.e. non-proportional hazards) can be included by interacting covariates

with time (by specifying them in the `tde` argument); the default behaviour is to interact the covariates with linear time, however, they can be interacted with some other function of time simply by using the `tdefunction` argument.

Under the 2-component mixture distributions (obtained by setting `mixture = TRUE`) the baseline survival at time t is taken to be $S(t) = p * S_1(t) + (1 - p) * S_2(t)$ where $S_1(t)$ and $S_2(t)$ are the baseline survival under each component of the mixture distribution and p is the mixing parameter specified via the argument `pmix`. Each component of the mixture distribution is assumed to be either exponential, Weibull or Gompertz. The 2-component mixture distributions can allow for a variety of flexible baseline hazard functions (see Crowther and Lambert (2013) for some examples).

If the user wishes to provide a user-defined hazard or log hazard function (instead of using one of the standard parametric survival distributions) then this is also possible via the `hazard` or `loghazard` argument. If a user-defined hazard or log hazard function is specified, then this is allowed to be time-dependent, and the resulting cumulative hazard function does not need to have a closed-form solution. The survival times are generated using the approach described in Crowther and Lambert (2013), whereby the cumulative hazard is evaluated using numerical quadrature and survival times are generated using an iterative algorithm which nests the quadrature-based evaluation of the cumulative hazard inside Brent's (1973) univariate root finder (for the latter the `uniroot` function is used). Not requiring a closed form solution to the cumulative hazard function has the benefit that survival times can be generated for complex models such as joint longitudinal and survival models; the **Examples** section provides an example of this.

Parameterisation for the exponential distribution: For the exponential distribution, with scale parameter $\lambda > 0$, the baseline hazard and survival functions used by `simsurv` are: $h(t) = \lambda$ and $S(t) = \exp(-\lambda)$.

Our parameterisation is equivalent to the one used by Wikipedia, the `dexp` function, the **eha** package, and the **flexsurv** package, except what we call the scale parameter they call the rate parameter.

Parameterisation for the Weibull distribution: For the Weibull distribution, with shape parameter $\gamma > 0$ and scale parameter $\lambda > 0$, the baseline hazard and survival functions used by `simsurv` are: $h(t) = \gamma\lambda t^{\gamma-1}$ and $S(t) = \exp(-\lambda t^{\gamma})$. Setting γ equal to 1 leads to the exponential distribution as a special case.

Our parameterisation differs from the one used by Wikipedia, `dweibull`, the `phreg` modelling function in the **eha** package, and the `flexsurvreg` modelling function in the **flexsurv** package. The parameterisation used in those functions can be achieved by transforming the scale parameter via the relationship $b = \lambda^{\frac{-1}{\gamma}}$, or equivalently $\lambda = b^{-\gamma}$ where b is the scale parameter under their parameterisation of the Weibull distribution.

Parameterisation for the Gompertz distribution: For the Gompertz distribution, with and shape parameter $\gamma > 0$ and scale parameter $\lambda > 0$, the baseline hazard and survival functions used by `simsurv` are: $h(t) = \lambda \exp(\gamma t)$ and $S(t) = \exp(\frac{-\lambda(\exp(\gamma t)-1)}{\gamma})$. Setting γ equal to 0 leads to the exponential distribution as a special case.

Our parameterisation is equivalent to the one used by the `dgompertz` and `flexsurvreg` functions in the **flexsurv** package, except they use slightly different terminology. Their parameterisation can be achieved via the relationship $a = \gamma$ and $b = \lambda$ where a and b are their shape and rate parameters, respectively.

Our parameterisation differs from the one used in the dgompertz and phreg functions in the **eha** package. Their parameterisation can be achieved via the relationship $a = \lambda$ and $b = \frac{1}{\gamma}$ where a and b are their shape and scale parameters, respectively.

Our parameterisation differs from the one used by Wikipedia. Their parameterisation can be achieved via the relationship $a = \frac{\lambda}{\gamma}$ and $b = \gamma$ where a and b are their shape and scale parameters, respectively.

Value

A data frame with a row for each individual, and the following three columns:

- `id` The individual identifier
- `eventtime` The simulated event (or censoring) time
- `status` The event indicator, 1 for failure, 0 for censored

Note

This package is modelled on the user-written `survsim` package available in the Stata software (see Crowther and Lambert (2012)).

Author(s)

Sam Brilleman (<sam.brilleman@gmail.com>)

References

Brilleman SL, Wolfe R, Moreno-Betancur M, and Crowther MJ. (2020) Simulating survival data using the simsurv R package. *Journal of Statistical Software* **96**(9), 1–27. doi: 10.18637/jss.v097.i03.

Crowther MJ, and Lambert PC. (2013) Simulating biologically plausible complex survival data. *Statistics in Medicine* **32**, 4118–4134. doi: 10.1002/sim.5823

Bender R, Augustin T, and Blettner M. (2005) Generating survival times to simulate Cox proportional hazards models. *Statistics in Medicine* **24**(11), 1713–1723.

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

Crowther MJ, and Lambert PC. (2012) Simulating complex survival data. *The Stata Journal* **12**(4), 674–687. https://www.stata-journal.com/sjpdf.html?articlenum=st0275

Examples

```
#-------------- Simpler examples

# Generate times from a Weibull model including a binary
# treatment variable, with log(hazard ratio) = -0.5, and censoring
# after 5 years:
set.seed(9911)
covs <- data.frame(id = 1:100, trt = stats::rbinom(100, 1L, 0.5))
s1 <- simsurv(lambdas = 0.1, gammas = 1.5, betas = c(trt = -0.5),
              x = covs, maxt = 5)
head(s1)
```

```
# Generate times from a Gompertz model:
s2 <- simsurv(dist = "gompertz", lambdas = 0.1, gammas = 0.05, x = covs)

# Generate times from a 2-component mixture Weibull model:
s3 <- simsurv(lambdas = c(0.1, 0.05), gammas = c(1, 1.5),
              mixture = TRUE, pmix = 0.5, x = covs, maxt = 5)

# Generate times from user-defined log hazard function:
fn <- function(t, x, betas, ...)
  (-1 + 0.02 * t - 0.03 * t ^ 2 + 0.005 * t ^ 3)
s4 <- simsurv(loghazard = fn, x = covs, maxt = 1.5)

# Generate times from a Weibull model with diminishing treatment effect:
s5 <- simsurv(lambdas = 0.1, gammas = 1.5, betas = c(trt = -0.5),
              x = covs, tde = c(trt = 0.05), tdefunction = "log")

#-------------- Complex examples

# Here we present an example of simulating survival times
# based on a joint longitudinal and survival model

# First we define the hazard function to pass to simsurv
# (NB this is a Weibull proportional hazards regression submodel
# from a joint longitudinal and survival model with a "current
# value" association structure).
haz <- function(t, x, betas, ...) {
    betas[["shape"]] * (t ^ (betas[["shape"]] - 1)) * exp(
      betas[["betaEvent_intercept"]] +
      betas[["betaEvent_binary"]] * x[["Z1"]] +
      betas[["betaEvent_continuous"]] * x[["Z2"]] +
      betas[["betaEvent_assoc"]] * (
        betas[["betaLong_intercept"]] +
        betas[["betaLong_slope"]] * t +
        betas[["betaLong_binary"]] * x[["Z1"]] +
        betas[["betaLong_continuous"]] * x[["Z2"]]
      )
    )
}

# Then we construct data frames with the true parameter
# values and the covariate data for each individual
set.seed(5454) # set seed before simulating data
N <- 20        # number of individuals

# Population (fixed effect) parameters
betas <- data.frame(
  shape                = rep(2,    N),
  betaEvent_intercept  = rep(-11.9,N),
  betaEvent_binary     = rep(0.6,  N),
  betaEvent_continuous = rep(0.08, N),
  betaEvent_assoc      = rep(0.03, N),
  betaLong_binary      = rep(-1.5, N),
```

```
  betaLong_continuous  = rep(1,    N),
  betaLong_intercept   = rep(90,   N),
  betaLong_slope       = rep(2.5,  N)
)

# Individual-specific (random effect) parameters
b_corrmat <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
b_sds     <- c(20, 3)
b_means   <- rep(0, 2)
b_z       <- MASS::mvrnorm(n = N, mu = b_means, Sigma = b_corrmat)
b         <- sapply(1:length(b_sds), function(x) b_sds[x] * b_z[,x])
betas$betaLong_intercept <- betas$betaLong_intercept + b[,1]
betas$betaLong_slope     <- betas$betaLong_slope     + b[,2]

# Covariate data
covdat <- data.frame(
  Z1 = stats::rbinom(N, 1, 0.45), # a binary covariate
  Z2 = stats::rnorm(N, 44, 8.5)   # a continuous covariate
)

# Then we simulate the survival times based on the
# hazard function, covariates, and true parameter values
times <- simsurv(hazard = haz, x = covdat, betas = betas, maxt = 10)
head(times)
```

Index