Package ‘survobj’

September 20, 2026

Title Simulate Parametric and Semi-Parametric Survival Times with Object-Oriented Design

Version 3.2.0

Description Simulate parametric and semi-parametric survival times through a consistent, reusable interface for each distribution, using an object-oriented design. Supported distributions include Exponential, Weibull, Gompertz, Log-Logistic, Log-Normal, and Piecewise Exponential. Random variates can be generated under Proportional Hazards, Accelerated Failure Time, and Extended Hazards models, as well as under renewal and non-homogeneous Poisson recurrent event processes, following the methods described by Bender (2003) <doi:10.5282/UBM/EPUB.1716> and Leemis (1987) in Operations Research, 35(6), 892-894.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Language en-US

Config/testthat/edition 3

Imports dplyr, tidyr, ggplot2, survival

VignetteBuilder knitr

URL https://johnaponte.github.io/survobj/, https://github.com/johnaponte/survobj

BugReports https://github.com/johnaponte/survobj/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Aponte John [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-3014-3673>)

Maintainer Aponte John <john.j.aponte@gmail.com>

Repository CRAN

Date/Publication 2026-09-20 18:30:10 UTC

Contents

`censor_event` *Censor of events*

Description

if censor_time < time, event is changed to 0, otherwise not changed

Usage

```
censor_event(censor_time, time, event)

censor_time(censor_time, time)
```

Arguments

`censor_time`	the time to censor
`time`	the time variable where the censor_time is applied
`event`	the variable with the event. It can be initialized in the call with a value for all times.

Details

if censor_time < time, time is changed to censor_time, otherwise no change

Be careful and do not overwrite the time variable with the censor time variable to not lose track of the events

Value

censored time or event

Functions

- `censor_time()`: Censor time

Examples

```
# Typical workflow in a simulation of survival time.
# Simulate time to event (sim_t_event)
# and simulates the time to lost to follow up (tim_t_ltof)
# the simulation time frame is 1, so everything after 1 is censored

require(dplyr)
data.frame(
  sim_t_event = c(0.5,0.6,1,10,20),
  sim_t_ltof = c(2,0.5,2,2,0.8)
 ) |>
 mutate(sevent = censor_event(1,sim_t_event,event=1)) |>
 mutate(stime = censor_time(1,sim_t_event)) |>
 mutate(event = censor_event(sim_t_ltof, stime, sevent)) |>
 mutate(timeto = censor_time(sim_t_ltof, stime))
```

`fx_h_time` *Functions to help in time conversion*

Description

This set of functions helps in the time conversion, taking into account generic times and not specific times. The conversions are based on the assumption that 1 year is 365.25 days and is 12 months. There is no adjustment for leap days or hours or difference of days between months

Usage

```
dtom(x)

mtod(x)

dtoy(x)

ytod(x)

mtoy(x)

ytom(x)
```

Arguments

`x`	the time to convert

Value

the converted time

Functions

- `dtom()`: convert days to months
- `mtod()`: convert months to days
- `dtoy()`: convert days to years
- `ytod()`: convert years to days
- `mtoy()`: convert months to year
- `ytom()`: convert years to months

Examples

```
dtom(365.25)
mtod(12)
dtoy(165.25)
ytod(1)
mtoy(12)
ytom(365.25)
```

`is_single_number` *Confirm is a single number*

Description

Evaluates if the argument is a single number

Usage

```
is_single_number(x)
```

Arguments

`x`	a variable to evaluate

Value

`TRUE` if it is a single number, `FALSE` otherwise

Examples

```
is_single_number(3)  #TRUE
is_single_number(c(3,3,3)) #FALSE
is_single_number(list(a=3)) #FALSE
is_single_number("3") #FALSE
```

nhpp *Generate random recurrent episodes under a non-homogeneous Poisson process*

Description

Generate random draws from the distribution of a recurrent set of survival times under a non-homogeneous Poisson process following Leemis (1987)

Usage

```
nhpphr(SURVIVAL, hr, prevtime)

nhppaft(SURVIVAL, aft, prevtime)
```

Arguments

SURVIVAL	Object of survival class
hr	Vector of hazard ratios
prevtime	Vector of previous survival times
aft	Vector of accelerated failure ratios

Value

Vector of survival times

Functions

- `nhpphr()`: Recurrent episodes under a proportional hazard model
- `nhppaft()`: Recurrent episodes under an accelerated failure time model

Examples

```
s_obj <- s_exponential(fail = 0.4, t = 1)
hr <- c(1,1,0.5,0.5)
time1 <- rsurvhr(s_obj, hr)
time2 <- nhpphr(s_obj,hr, time1)

s_obj <- s_exponential(fail = 0.4, t = 1)
aft <- c(1,1,0.5,0.5)
timea <- rsurvaft(s_obj, aft)
timeb <- nhppaft(s_obj, aft, timea)
```

renewal *Generate random recurrent episodes under a renewal Poisson process*

Description

Generate random draws from the distribution of a recurrent set of survival times under a renewal Poisson process following Leemis (1987)

Usage

```
renewhr(SURVIVAL, hr, prevtime)

renewaft(SURVIVAL, aft, prevtime)
```

Arguments

SURVIVAL	Object of survival class
hr	Vector of hazard ratios
prevtime	Vector of previous survival times
aft	Vector of accelerated failure time ratios

Value

Vector of survival times

Vector of survival times

Functions

- `renewhr()`: Recurrent episodes under a proportional hazard model
- `renewaft()`: Recurrent episodes under an accelerated failure time model

Examples

```
s_obj <- s_exponential(fail = 0.4, t = 1)
hr <- c(1,1,0.5,0.5)
time1 <- rsurvhr(s_obj, hr)
time2 <- renewhr(s_obj, hr, time1)

s_obj2 <- s_exponential(fail = 0.4, t = 1)
aft <- c(1,1,0.5,0.5)
timea <- rsurvaft(s_obj2, aft)
timeb <- renewaft(s_obj2, aft, timea)
```

sfx *Functions for SURVIVAL objects*

Description

All the SURVIVAL objects have access to the functions described here

Usage

```
sfx(SURVIVAL, t)

hfx(SURVIVAL, t)

Cum_Hfx(SURVIVAL, t)

invCum_Hfx(SURVIVAL, H)

rsurv(SURVIVAL, n)

rsurvhr(SURVIVAL, hr)

rsurvaft(SURVIVAL, aft)

rsurveh(SURVIVAL, aft, hr)

plot_survival(SURVIVAL, timeto, main)

ggplot_survival_random(SURVIVAL, timeto, subjects, nsim, alpha = 0.1)

compare_survival(SURVIVAL1, SURVIVAL2, timeto, main)

ggplot_survival_hr(SURVIVAL, hr, timeto, subjects, nsim, alpha = 0.1)

ggplot_survival_aft(SURVIVAL, aft, timeto, subjects, nsim, alpha = 0.1)

ggplot_survival_eh(SURVIVAL, aft, hr, timeto, subjects, nsim, alpha = 0.1)
```

Arguments

SURVIVAL	a SURVIVAL object
t	Time
H	cumulative hazard
n	number of observations
hr	hazard ratio
aft	accelerated failure time

`timeto`	plot the distribution up to timeto
`main`	title of the graph
`subjects`	number of subjects per group to simulate in each simulation
`nsim`	number of simulations
`alpha`	alpha value for the graph
`SURVIVAL1`	a SURVIVAL object
`SURVIVAL2`	a SURVIVAL object

Details

`rsurveh()` generates random values under the Extended Hazards model of Chen & Jewell (2001), which combines a proportional-hazards effect and an accelerated-failure-time effect: h*(t) = hr * aft * h0(aft * t). It nests the proportional hazards model (`aft = 1`) and the accelerated failure time model (`hr = 1`) as special cases. Note this differs from the Accelerated Hazards model of Chen & Wang (2000), h*(t) = h0(theta * t), which is the special case `hr = 1 / aft`.

Value

Depending on the function a proportion surviving, hazard, cumulative hazard, inverse of the cumulative hazard, a random draw or a plot

Functions

- `sfx()`: Survival function
- `hfx()`: Hazard function
- `Cum_Hfx()`: Cumulative Hazard function
- `invCum_Hfx()`: Inverse of the Cumulative Hazard function
- `rsurv()`: Generate random values from the distribution
- `rsurvhr()`: Generate random values from the distribution under proportional hazard ratios
- `rsurvaft()`: Generate random values from the distribution under accelerated failure time ratios
- `rsurveh()`: Generate random values from the distribution under the Extended Hazards model (combined proportional hazards and accelerated failure time)
- `plot_survival()`: Plot of the survival functions
- `ggplot_survival_random()`: ggplot of the simulation of survival times
- `compare_survival()`: Compare graphically two survival distributions
- `ggplot_survival_hr()`: ggplot of the simulation of survival times with hazard ratios
- `ggplot_survival_aft()`: ggplot of the simulation of survival times with accelerated failure times
- `ggplot_survival_eh()`: ggplot of the simulation of survival times with the Extended Hazards model

Examples

```
#' # Define a SURVIVAL object
obj <- s_factory(s_weibull, surv = 0.8, t = 2, shape = 1.2)

# Survival, Hazard and Cumulative hazard at time 0.4
sfx(SURVIVAL = obj, t= 0.4)
hfx(SURVIVAL = obj, t = 0.4)
Cum_Hfx(SURVIVAL = obj, t = 0.4)

# Time when the Cumulative hazard is 0.8
invCum_Hfx(SURVIVAL = obj, H = 0.8)

# Draw one random survival time from the distribution
rsurv(SURVIVAL = obj, n = 1)

# Draw one random survival time from the distribution under Proportional
# hazard, Accelerated failure time or the combined Extended Hazards model.
rsurvhr(SURVIVAL = obj, hr = 0.5)
rsurvaft(SURVIVAL = obj, aft = 2)
rsurveh(SURVIVAL = obj, aft = 2, hr = 0.5)

# Plot the survival functions
plot_survival(SURVIVAL = obj, timeto = 2, main = "Example of Weibull distribution" )
```

---

s_exponential — *Factory of SURVIVAL objects with Exponential distributions*

Description

Creates a SURVIVAL object with an Exponential distribution.

Usage

```
s_exponential(...)
```

Arguments

`...`	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the exponential distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create an exponential survival object the following options are available:

lambda to specify the canonical parameter of the distribution, or

surv and t for the proportion surviving (no events) at time t, or

fail and t for the proportion failing (events) at time t

lambda = -log(surv)/t

lambda = -log(1-fail)/t

The parameters should be spelled correctly as partial matching is not available

Examples

```
s_exponential(lambda = 3)
s_exponential(surv = 0.4, t = 2)
s_exponential(fail = 0.6, t = 2)
```

---

s_factory *Factory of objects of class SURVIVAL*

Description

Create objects of the class SURVIVAL

Usage

```
s_factory(s_family, ...)
```

Arguments

s_family	a factory for a specific distribution
...	parameters to define the survival distribution

Details

The objects of the class SURVIVAL define different distributions of survival times. Each class has its own set of parameters but once the SURVIVAL object is defined, they have access to the same functions to calculate:

- survival function: sfx(),
- hazard function: hfx(),
- cumulative hazard function: Cum_Hfx()
- the inverse of the cumulative hazard function: invCum_Hfx().
- generate random survival times: rsurv()
- generate random survival times under proportional hazard ratio: rsurvhr().

There are several functions to plot the distributions

- generic S3: `plot.SURVIVAL()`
- `plot_survival()`: to plot the functions
- `ggplot_survival_random()`: to ggplot random draws from the distribution
- `compare_survival()`: to compare the functions of two SURVIVAL objects

Value

a SURVIVAL object

Distributions

The following factories are implemented:

- `s_exponential()`: for Exponential distributions
- `s_weibull()`: for Weibull distributions
- `s_gompertz()`: for Gompertz distributions
- `s_piecewise()`: for Piecewise exponential distributions
- `s_loglogistic()`: for Log-Logistic distributions
- `s_lognormal()`: for Log-Normal distributions

Examples

```
# Define a SURVIVAL object
obj <- s_factory(s_exponential, lambda = 2)

# Survival, Hazard and Cumulative hazard at time 0.4
sfx(SURVIVAL = obj, t= 0.4)
hfx(SURVIVAL = obj, t = 0.4)
Cum_Hfx(SURVIVAL = obj, t = 0.4)

# Time when the Cumulative hazard is 0.8
invCum_Hfx(SURVIVAL = obj, H = 0.8)

# Draw one random survival time from the distribution
rsurv(SURVIVAL = obj, n = 1)

# Draw one random survival time from the distribution, with hazard ratio 0.5
rsurvhr(SURVIVAL = obj, hr = 0.5)

# Plot the survival functions
plot(obj)
```

`s_gompertz` *Factory of SURVIVAL objects with Gompertz distributions*

Description

Creates a SURVIVAL object with a Gompertz distribution.

Usage

```
s_gompertz(...)
```

Arguments

`...`	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the Gompertz distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create a Gompertz survival object the following options are available:

`scale` and `shape` to specify the canonical parameters of the distribution, or

`surv`, `t` and `shape` for the proportion surviving (no events) at time t and shape, or

`fail` and `t` and `shape` for the proportion failing (events) at time t and shape.

scale = -log(surv)·shape/(exp(shape·t))

scale = -log(1-fail)·shape/(exp(shape·t))

The parameters should be spelled correctly as partial matching is not available

Negative shape

A negative `shape` produces a decreasing hazard whose cumulative hazard is bounded above by `scale / abs(shape)` (following Bender, Augustin & Blettner 2005). This makes the distribution improper: a cure fraction of `exp(-scale / abs(shape))` of the subjects never experience the event, so their simulated survival time (`rsurv`, `rsurvhr`, `rsurvaft`, `rsurveh`) and `invCum_Hfx()` are returned as `Inf`.

Examples

```
s_gompertz(scale = 1, shape = 1.5)
s_gompertz(surv = 0.4, t = 2, shape = 1.5)
s_gompertz(fail = 0.6, t = 2, shape = 1.5)
```

s_loglogistic *Factory of SURVIVAL objects with Log Logistic distributions*

Description

Creates a SURVIVAL object with a Log Logistic distribution.

Usage

```
s_loglogistic(...)
```

Arguments

`...`	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the log-logistic distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create a Log Logistic survival object the following options are available:

`scale` and `shape` to specify the canonical parameters of the distribution, or

`surv`, `t` and `shape` for the proportion surviving (no events) at time t and the shape parameter, or

`fail`, `t` and `shape` for the proportion failing (events) at time t and the shape parameter, or

`intercept` and `scale` for the parameters returned by `survreg(.., dist = "loglogistic")` models.

The parameters should be spelled correctly as partial matching is not available

Examples

```
s_loglogistic(scale = 2,shape = 2)
s_loglogistic(surv = 0.6, t= 12, shape = 0.5)
s_loglogistic(fail = 0.4, t = 12, shape =0.5)
s_loglogistic(intercept = 0.4, scale = 0.5)
```

s_lognormal *Factory of SURVIVAL objects with Log Normal distributions*

Description

Creates a SURVIVAL object with a Log Normal distribution.

Usage

```
s_lognormal(...)
```

Arguments

...	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the log-normal distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create a Log Normal survival object the following options are available:

`scale` and `shape` to specify the canonical parameters of the distribution, or

`surv`, `t` and `shape` for the proportion surviving (no events) at time t and the shape parameter, or

`fail`, `t` and `shape` for the proportion failing (events) at time t and the shape parameter, or

`intercept` and `scale` for the parameters returned by `survreg(.., dist = "lognormal")` models.

The scale parameter is the median value of the distribution, and the shape is the log standard deviation

The parameters should be spelled correctly as partial matching is not available

Examples

```
s_lognormal(scale = 2,shape = 2)
s_lognormal(surv = 0.6, t= 12, shape = 0.5)
s_lognormal(fail = 0.4, t = 12, shape =0.5)
s_lognormal(intercept = 0.4, scale = 0.5)
```

`s_piecewise` *Factory of SURVIVAL objects with Piecewise Exponential distributions*

Description

Creates a SURVIVAL object with a Piecewise Exponential distribution.

Usage

```
s_piecewise(...)
```

Arguments

`...`	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the piecewise exponential distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create a piecewise exponential survival object the following options are available:

`breaks` and `hazards` to specify the exponential (constant) hazard until each break, or

`surv`, `breaks` and `segments` for the proportion surviving (no events) at the end of the last segment, or

`fail`, `breaks` and `segments` for the proportion failing (events) at the end of the last segment

If `surv` or `fail` parameters are indicated, the `segments` are scaled to hazards in order to match the surviving or failing proportion at the end of the last segment.

Define the last break point as `Inf` to fully define the distribution, otherwise an error will be produced if the function after the last break is requested

The parameters should be spelled correctly as partial matching is not available

Examples

```
s_piecewise(breaks = c(1,2,3,Inf), hazards = c(0.5,0.6,0.5,0.1))
s_piecewise(surv = 0.4, breaks = c(1,2,3,Inf), segments = c(1,2,1,2))
s_piecewise(fail = 0.6, breaks = c(1,2,3,Inf), segments = c(1,2,1,2))
```

s_weibull *Factory of SURVIVAL objects with Weibull distributions*

Description

Creates a SURVIVAL object with a Weibull distribution.

Usage

```
s_weibull(...)
```

Arguments

`...`	Parameters to define the distribution. See the Parameters for details

Value

a SURVIVAL object of the Weibull distribution family. See the documentation of `s_factory` for the methods available for SURVIVAL objects

Parameters

To create a Weibull survival object the following options are available:

`scale` and `shape` to specify the canonical parameters of the distribution, or

`surv`, `t` and `shape` for the proportion surviving (no events) at time t and the shape parameter, or

`fail`, `t` and `shape` for the proportion failing (events) at time t and the shape parameter, or

`intercept` and `scale` for the parameters returned by `survreg(.., dist = "weibull")` models.

scale = -log(surv)/(t^shape)

scale = -log(1-fail)/(t^shape)

The parameters should be spelled correctly as partial matching is not available

Examples

```
s_weibull(scale = 2,shape = 2)
s_weibull(surv = 0.6, t= 12, shape = 0.5)
s_weibull(fail = 0.4, t = 12, shape =0.5)
s_weibull(intercept = 0.4, scale = 0.5)
```

Index