Package ‘reReg’

April 15, 2021

Title Recurrent Event Regression

Version 1.4.0

Description
A comprehensive collection of practical and easy-to-use tools for regression analysis of recurrent events, with or without the presence of a (possibly) informative terminal event. The modeling framework is based on a joint frailty scale-change model, that includes models described in Wang et al. (2001) <doi:10.1198/016214501753209031>, Huang and Wang (2004) <doi:10.1198/016214504000001033>, Xu et al. (2017) <doi:10.1080/01621459.2016.1173557>, and Xu et al. (2019) <doi:10.5705/SS.202018.0224> as special cases. The implemented estimating procedure does not require any parametric assumption on the frailty distribution. The package also allows the users to specify different model forms for both the recurrent event process and the terminal event.

Depends R (>= 3.5.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/stc04003/reReg

BugReports https://github.com/stc04003/reReg/issues

Imports BB, nleqslv, SQUAREM, survival, directlabels, ggplot2, MASS, methods, reda (>= 0.5.0), scam, Rcpp, rootSolve

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

NeedsCompilation yes

Author Sy Han (Steven) Chiou [aut, cre], Chiung-Yu Huang [aut]

Maintainer Sy Han (Steven) Chiou <schiou@utdallas.edu>

Repository CRAN

Date/Publication 2021-04-14 22:10:02 UTC
Description

The package offers a comprehensive collection of practical and easy-to-use tools for analyzing recurrent event data, with or without the presence of a (possibly) correlated terminal event. The modeling framework is based on a joint frailty scale-change model, that encompasses many existing models, including the popular Cox-type models, as special cases and accommodates informative censoring through a subject-specific frailty. The implemented estimating procedure does not require any parametric assumption on the frailty distribution. The package allows the users to specify different model forms for both the recurrent event process and the terminal event. The package also included are visualization of recurrent events and simulation from the regression models.

Author(s)

Maintainer: Sy Han (Steven) Chiou <schiou@utdallas.edu>

Authors:

- Chiung-Yu Huang <ChiungYu.Huang@ucsf.edu>
References


See Also

Useful links:

- [https://github.com/stc04003/reReg](https://github.com/stc04003/reReg)
- Report bugs at [https://github.com/stc04003/reReg/issues](https://github.com/stc04003/reReg/issues)

---

**basebind**

*Function used to combine baseline functions in one plot*

**Description**

Combine different plots into one.

**Usage**

```r
basebind(..., legend.title, legend.labels)
```

**Arguments**

- `...` ggplot objects created by plotting `reReg` objects.
- `legend.title` an optional character string to specify the legend title.
- `legend.labels` an optional character string to specify the legend labels.
Examples

```r
data(simDat)
fm <- Recur(t.stop, id, event, status) ~ x1 + x2
fit1 <- reReg(fm, subset = x1 == 0, data = simDat, B = 200)
fit2 <- reReg(fm, subset = x1 == 1, data = simDat, B = 200)
basebind(plot(fit1), plot(fit2))
```

mcf

The `mcf` function is imported from `reda`.

Description

An S4 class generic function that returns the mean cumulative function (mcf) estimates. The function is imported from the `reda` package; see `?reda::mcf` for more details.

Examples

```r
data(simDat)
m <- mcf(Recur(t.start %to% t.stop, id, event, status) ~ x1, data = simDat)
plot(m, conf.int = TRUE)
```

mcf.formula-class

The `mcf.formula` class is imported from `reda`.

Description

The class `mcf.formula` represents a mcf formula. See `reda` for details.

plot.Recur

Produce Event Plot or Mean Cumulative Function Plot

Description

Plot the event plot or the mean cumulative function (MCF) from an Recur object.

Usage

```r
## S3 method for class 'Recur'
plot(
x,
mcf = FALSE,
event.result = c("increasing", "decreasing", "asis"),
event.calendarTime = FALSE,
mcf.adjustRiskset = TRUE,
mcf.conf.int = FALSE,
control = list(),
...)
```
Arguments

x an object of class `Recur` returned by the `Recur()` function. See ?Recur for creating Recur objects.

mcf an optional logical value indicating whether the mean cumulative function (MCF) will be plotted instead of the event plot. When `mcf = TRUE`, the `mcf` is internally called. See `mcf` for details.

event.result an optional character string that is passed to the `plotEvents()` function as the result argument. See `plotEvents`. This argument is used to specify whether the event plot is sorted by the subjects' terminal time. The available options are:

- increasing sort the terminal time from in ascending order (default). This places longer terminal times on top.
- decreasing sort the terminal time from in descending order. This places shorter terminal times on top.
- none present the event plots as is, without sorting by the terminal times.

event.calendarTime an optional logical value indicating whether to plot in calendar time. When `event.calendarTime = FALSE` (default), the event plot will have patient time on the x-axis.

mcf.adjustRiskset an optional logical value that is passed to the `mcf()` function as the `adjustRiskset` argument. This argument indicates whether risk set size will be adjusted. If `mcf.adjustRiskset = TRUE`, subjects leave the risk set after terminal times as in the Nelson-Aalen estimator. If `mcf.adjustRiskset = FALSE`, subjects remain in the risk set after terminal time.

mcf.conf.int an optional logical value that is passed to the `mcf()` function as the `conf.int` argument. See `mcf` for details.

control a list of control parameters. See Details.

... additional graphical parameters to be passed to methods.

Details

The argument `control` consists of options with argument defaults to a list with the following values:

- **xlab** customizable x-label, default value is "Time".
- **ylab** customizable y-label, default value is "Subject" for event plot and "Cumulative mean" for MCF plot.
- **main** customizable title, the default value is "Recurrent event plot" when `mcf = FALSE` and "Sample cumulative mean function plot" when `mcf = TRUE`.
- **terminal.name** customizable label for terminal event, the default value is "Terminal event".
- **recurrent.name** customizable legend title for recurrent event, the default value is "Recurrent events".
- **recurrent.types** customizable label for recurrent event type, the default value is NULL.
- **alpha** between 0 and 1, controls the transparency of points.

The xlab, ylab and main parameters can be specified outside of the control list.
Value

A ggplot object.

References


See Also

Recur, plotEvents, mcf

Examples

data(simDat)
reObj <- with(simDat, Recur(t.start %to% t.stop, id, event, status))

## Event plots:
plot(reObj)
plot(reObj, event.result = "decreasing")

## With (hypothetical) multiple event types
simDat$event2 <- with(simDat, ifelse(t.stop > 10 & event > 0, 2, event))
reObj2 <- with(simDat, Recur(t.start %to% t.stop, id, event2, status))
plot(reObj2)

## With (hypothetical) calendar times
simDat2$t.start <- as.Date(simDat2$t.start + simDat2$x2 * 5, origin = "20-01-01")
simDat2$t.stop <- as.Date(simDat2$t.stop + simDat2$x2 * 5, origin = "20-01-01")
reObj3 <- with(simDat2, Recur(t.start %to% t.stop, id, event, status))
plot(reObj3, event.calendarTime = TRUE)

## MCF plots
plot(reObj, mcf = TRUE)
plot(reObj, mcf = TRUE, mcf.adjustRiskset = FALSE)
### Usage

```r
## S3 method for class 'reReg'
plot(
  x,
  baseline = c("both", "rate", "hazard"),
  smooth = FALSE,
  newdata = NULL,
  frailty = NULL,
  showName = FALSE,
  control = list(),
  ...
)
```

### Arguments

- **x**: an object of class `reReg`, returned by the `reReg` function.
- **baseline**: a character string specifying which baseline function to plot.
  - `baseline = "both"` plot both the baseline cumulative rate and the baseline cumulative hazard function (if applicable) in separate panels within the same display (default).
  - `baseline = "rate"` plot the baseline cumulative rate function.
  - `baseline = "hazard"` plot the baseline cumulative hazard function.
- **smooth**: an optional logical value indicating whether to add a smooth curve obtained from a monotone increasing P-splines implemented in package `scam`.
- **newdata**: an optional data frame contains variables to include in the calculation of the cumulative rate function. If omitted, the baseline rate function will be plotted.
- **frailty**: an optional vector to specify the shared frailty for `newdata`. If `newdata` is given and `frailty` is not specified, the
- **showName**: an optional logical value indicating whether to label the curves when `newdata` is specified.
- **control**: a list of control parameters. See **Details**.
  - additional graphical parameters to be passed to methods.

### Details

The argument `control` consists of options with argument defaults to a list with the following values:

- **xlab**: customizable x-label, default value is "Time".
- **ylab**: customizable y-label, default value is empty.
- **main**: customizable title, default value are "Baseline cumulative rate and hazard function" when `baseline = "both"`, "Baseline cumulative rate function" when `baseline = "rate"`, and "Baseline cumulative hazard function" when `baseline = "hazard".

### Value

A `ggplot` object.
plotEvents

See Also
reReg

Examples

```r
data(simDat)
fm <- Recur(t.start %to% t.stop, id, event, status) ~ x1 + x2

fit <- reReg(fm, data = simDat, B = 0)
plot(fit)
plot(fit, xlab = "Time (days)", smooth = TRUE)

## Predicted cumulative rate and hazard given covariates
newdata <- expand.grid(x1 = 0:1, x2 = mean(simDat$x2))
plot(fit, newdata = newdata, showName = TRUE)
```

---

plotEvents

**Produce Event Plots**

Description

Plot the event plot for an Recur object. The usage of the function is similar to that of plot.Recur() but with more flexible options.

Usage

```r
plotEvents(
  formula,
  data,
  result = c("increasing", "decreasing", "none"),
  calendarTime = FALSE,
  control = list(),
  ...
)
```

Arguments

- **formula** a formula object, with the response on the left of a "~" operator, and the predictors on the right. The response must be a recurrent event survival object as returned by function Recur().
- **data** an optional data frame in which to interpret the variables occurring in the "formula".
- **result** an optional character string specifying whether the event plot is sorted by the subjects' terminal time. The available options are
  - increasing sort the terminal time from in ascending order (default). This places longer terminal times on top.
  - decreasing sort the terminal time from in descending order. This places shorter terminal times on top.
none present the event plots as is, without sorting by the terminal times.

calendarTime an optional logical value indicating whether to plot in calendar time. When calendarTime = FALSE (default), the event plot will have patient time on the x-axis.

control a list of control parameters. See Details.

... graphical parameters to be passed to methods. These include xlab, ylab, main, and more. See Details.

Details

The argument control consists of options with argument defaults to a list with the following values:

xlab customizable x-label, default value is "Time".

ylab customizable y-label, default value is "Subject" for event plot and "Cumulative mean" for MCF plot.

main customizable title, the default value is "Recurrent event plot" when mcf = FALSE and "Sample cumulative mean function plot" when mcf = TRUE.

terminal.name customizable label for terminal event, the default value is "Terminal event".

recurrent.name customizable legend title for recurrent event, the default value is "Recurrent events".

recurrent.types customizable label for recurrent event type, the default value is NULL.

alpha between 0 and 1, controls the transparency of points.

The xlab, ylab and main parameters can be specified outside of the control list.

Value

A ggplot object.

See Also

Recur, plot.Recur

Examples

data(simDat)
plotEvents(Recur(t.start %to% t.stop, id, event, status) ~ 1, data = simDat,
           xlab = "Time in days", ylab = "Subjects arranged by terminal time")

## Separate plots by x1
plotEvents(Recur(t.start %to% t.stop, id, event, status) ~ x1, data = simDat)

## For multiple recurrent events
simDat$x3 <- ifelse(simDat$x2 < 0, "x2 < 0", "x2 > 0")
simDat$event <- simDat$event * sample(1:3, nrow(simDat), TRUE)
plotEvents(Recur(t.start %to% t.stop, id, event, status) ~ x1 + x3, data = simDat)
plotHaz

Plot the Baseline Cumulative Hazard Function for the Terminal Time

Description

Plot the baseline cumulative hazard function for an reReg object. The 95% confidence interval on the baseline cumulative rate function

Usage

plotHaz(
  x,
  newdata = NULL,
  frailty = NULL,
  showName = FALSE,
  type = c("unrestricted", "bounded", "scaled"),
  smooth = FALSE,
  control = list(),
  ...
)

Arguments

x an object of class reReg, returned by the reReg function.
newdata an optional data frame contains variables to include in the calculation of the cumulative rate function. If omitted, the baseline rate function will be plotted.
frailty an optional vector to specify the shared frailty for newdata. If newdata is given and frailty is not specified, the
showName an optional logical value indicating whether to label the curves when newdata is specified.
type a character string specifying the type of rate function to be plotted. Options are "unrestricted", "scaled", "bounded". See Details.
smooth an optional logical value indicating whether to add a smooth curve obtained from a monotone increasing P-splines implemented in package scam.
control a list of control parameters.
... graphical parameters to be passed to methods. These include xlab, ylab, main, and more. See Details.

Details

The argument control consists of options with argument defaults to a list with the following values:

xlab customizable x-label, default value is "Time".
ylab customizable y-label, default value is empty.
main customizable title, default value is "Baseline cumulative hazard function".

These arguments can also be passed down without specifying a control list.
plotRate

Value

A ggplot object.

See Also

reReg plot.reReg

Examples

data(simDat)
fm <- Recur(t.start %to% t.stop, id, event, status) ~ x1 + x2
fit <- reReg(fm, data = simDat, model = "cox|cox", B = 0)
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative hazard function
plotHaz(fit)
plotHaz(fit, smooth = TRUE)

plotRate

Plotting the Baseline Cumulative Rate Function for the Recurrent Event Process

Description

Plot the baseline cumulative rate function for an reReg object.

Usage

plotRate(
  x,
  newdata = NULL,
  frailty = NULL,
  showName = FALSE,
  type = c("unrestricted", "bounded", "scaled"),
  smooth = FALSE,
  control = list(),
  ...
)

Arguments

x an object of class reReg, usually returned by the reReg function.
newdata an optional data frame contains variables to include in the calculation of the cumulative rate function. If omitted, the baseline rate function will be plotted.
frailty an optional vector to specify the shared frailty for newdata. If newdata is given and frailty is not specified, the
showName an optional logical value indicating whether to label the curves when newdata is specified.

type a character string specifying the type of rate function to be plotted. Options are "unrestricted", "scaled", "bounded". See Details.

smooth an optional logical value indicating whether to add a smooth curve obtained from a monotone increasing P-splines implemented in package scam.

control a list of control parameters.

... graphical parameters to be passed to methods. These include xlab, ylab, main, and more. See Details.

Details

The plotRate() plots the estimated baseline cumulative rate function depending on the identifiability assumption. When type = "unrestricted" (default), the baseline cumulative rate function is plotted under the assumption $E(Z) = 1$. When type = "scaled", the baseline cumulative rate function is plotted under the assumption $\Lambda(\min(Y^{*}, \tau)) = 1$. When type = "bounded", the baseline cumulative rate function is plotted under the assumption $\Lambda(\tau) = 1$. See ?reReg for the specification of the notations and underlying models.

The argument control consists of options with argument defaults to a list with the following values:

xlab customizable x-label, default value is "Time".

ylab customizable y-label, default value is empty.

main customizable title, default value is "Baseline cumulative rate function".

These arguments can also be specified outside of the control list.

Value

A ggplot object.

See Also

reReg plot.reReg

Examples

data(simDat)
fм <- Recur(t.start %to% t.stop, id, event, status) ~ x1 + x2
fit <- reReg(fm, data = simDat, model = "cox|cox", B = 0)
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative rate function
plotRate(fit)
plotRate(fit, smooth = TRUE)
The Recur function is imported from reda.

**Description**

Create a recurrent event survival object, used as a response variable in reReg. This function is replacing the original reSurv() in version 1.1.6. See ?reda::Recur for more details.

**See Also**

%2%

**Examples**

Recur(2:6, id = c(1, 1, 1, 2, 2))
Recur(2:6, id = c(1, 1, 1, 2, 2))
Recur(1:5 %2% 2:6, id = c(1, 1, 1, 2, 2))

The Recur class is imported from reda.

**Description**

The class Recur is an S4 that represents a formula response for recurrent event data model. See reda for details.

**Recur-pipe**

The %to% function is imported from reda

**Description**

This pipe operator specifies the time segments or recurrent episodes by endpoints. See reda for more details.

**Examples**

Recur(2:6, id = c(1, 1, 1, 2, 2))
Recur(2:6, id = c(1, 1, 1, 2, 2))
Recur(1:5 %2% 2:6, id = c(1, 1, 1, 2, 2))
**reReg**

*Fits Semiparametric Regression Models for Recurrent Event Data*

**Description**

Fits a general (joint) semiparametric regression model for the recurrent event data, where the rate function of the underlying recurrent event process and the hazard function of the terminal event can be specified as a Cox-type model, an accelerated mean model, an accelerated rate model, or a generalized scale-change model. See details for model specifications.

**Usage**

```r
reReg(
  formula,
  data,
  subset,
  model = "cox",
  B = 0,
  se = c("boot", "sand"),
  control = list()
)
```

**Arguments**

- `formula`: a formula object, with the response on the left of a "~" operator, and the predictors on the right. The response must be a recurrent event survival object as returned by function `Recur`.
- `data`: an optional data frame in which to interpret the variables occurring in the "formula".
- `subset`: an optional logical vector specifying a subset of observations to be used in the fitting process.
- `model`: a character string specifying the underlying model. See Details.
- `B`: a numeric value specifies the number of bootstraps for variance estimation. When B = 0, variance estimation will not be performed.
- `se`: a character string specifying the method for the variance estimation. See Details.
  - `boot`: nonparametric bootstrap approach
  - `sand`: resampling-based sandwich estimator
- `control`: a list of control parameters.

**Details**

Suppose the recurrent event process and the failure events are observed in the time interval \( t \in [0, \tau] \), for some constant \( \tau \). We formulate the recurrent event rate function, \( \lambda(t) \), and the terminal event hazard function, \( h(t) \), in the form of

\[
\lambda(t) = Z\lambda_0(te^{X^T\alpha})e^{X^T\beta}, \quad h(t) = Zh_0(te^{X^T\eta})e^{X^T\theta},
\]
where $\lambda_0(t)$ is the baseline rate function, $h_0(t)$ is the baseline hazard function, $X$ is a $n \times p$ covariate matrix and $\alpha, Z$ is an unobserved shared frailty variable, and $(\alpha, \eta)$ and $(\beta, \theta)$ correspond to the shape and size parameters, respectively. The model includes several popular semiparametric models as special cases, which can be specified via the `model` argument with the rate function and hazard function separated by "|". For examples, Wang, Qin and Chiang (2001) ($\alpha = \eta = \theta = 0$) can be called with `model = "cox"`; Huang and Wang (2004) ($\alpha = \eta = 0$) can be called with `model = "cox|cox"`; Xu et al. (2017) ($\alpha = \beta$ and $\eta = \theta$) can be called with `model = "am|am"`; Xu et al. (2019) ($\eta = \theta = 0$) can be called with `model = "gsc"`. Users can mix the models depending on the application. For example, `model = "cox|ar"` postulate a Cox proportional model for the recurrent event rate function and an accelerated rate model for the terminal event hazard function ($\alpha = \theta = 0$). If only one model is specified without an "|", it is used for both the rate function and the hazard function. For example, specifying `model = "cox"` is equivalent to `model = "cox|cox"`. Some models that assumes $Z = 1$ and requires independent censoring are also implemented in `reReg`; these includes `model = "cox.LWYY"` for Lin et al. (2000), `model = "cox.GL"` for Ghosh and Lin (2002), and `model = "am.GL"` for Ghosh and Lin (2003).

The available methods for variance estimation are:

- **boot** performs nonparametric bootstrap.
- **sand** performs the efficient resampling-based variance estimation.

The `control` list consists of the following parameters:

- **tol** absolute error tolerance.
- **alpha, beta, eta, theta** initial guesses used for root search.
- **solver** the equation solver used for root search. The available options are `BB::BBsolve`, `BB::dfsane`, `BB::BBoptim`, and `optim`.
- **eqType** a character string indicating whether the log-rank type estimating equation or the Gehan-type estimating equation (when available) will be used.
- **boot.parallel** a logical value indicating whether parallel computation will be applied when `se = "boot"` is called.
- **boot.parCl** an integer value specifying the number of CPU cores to be used when `parallel = TRUE`. The default value is half the CPU cores on the current host.

**References**


**See Also**

`Recur`, `simGSC`

**Examples**

```r
data(simDat)

## Nonparametric estimate
plot(reReg(Recur(t.start %to% t.stop, id, event, status) ~ 1, data = simDat, B = 50))

fm <- Recur(t.start %to% t.stop, id, event, status) ~ x1 + x2
## Fit the Cox rate model
summary(reReg(fm, data = simDat, model = "cox", B = 50))
## Fit the joint Cox/Cox model
summary(reReg(fm, data = simDat, model = "cox|cox", B = 50))
## Fit the scale-change rate model
summary(reReg(fm, data = simDat, model = "gsc", B = 50, se = "sand"))
```

---

**residuals.reReg**

*Calculate Residuals for a ‘reReg’ Fit*

**Description**

Calculates residuals for a joint frailty scale-change model fitted by ‘reReg’. Under the recurrent event model, at each observation time, \( t \), the residual is calculated as

\[
\text{observed number of recurrent events at } t - \text{expected number of recurrent events at } t.
\]

The expected number of recurrent events at \( t \) is calculated by the cumulative rate function at \( t \). Under the failure time model, the residual is calculated as

\[
\Delta - H(t),
\]

where \( \Delta \) is the terminal event indicator and \( H(t) \) is the cumulative hazard function at \( t \).

**Usage**

```r
## S3 method for class 'reReg'
residuals(object, model = c("recurrent", "failure"), ...)
```
Arguments

object  an object of class reReg returned by the reReg() function.
model  a character string specifying whether the residuals will be calculated under the recurrent event model or the failure time model.
...  additional parameters for future development.

Description

Create a recurrent event survival object, used as a response variable in reReg. This function is deprecated in Version 1.1.6. A recurrent event object is now being created with Recur(). See '?Recur()' for details.

Usage

reSurv(time1, time2, id, event, status, origin = 0)

Arguments

time1  when "time2" is provided, this vector is treated as the starting time for the gap time between two successive recurrent events. In the absence of "time2", this is the observation time of recurrence on calendar time scale, in which, the time corresponds to the time since entry/inclusion in the study.
time2  an optional vector for ending time for the gap time between two successive recurrent events.
id  subject’s id.
event  a binary vector used as the recurrent event indicator. event = 1 for recurrent times.
status  a binary vector used as the status indicator for the terminal event. status = 0 for censored times.
origin  a numerical vector indicating the time origin of subjects. When origin is a scalar, reSurv assumes all subjects have the same origin. Otherwise, origin needs to be a numerical vector, with length equals to the number of subjects. In this case, each element corresponds to different origins for different subjects. This argument is only needed when "time2" is missing.

Examples

## Not run:
data(simDat)
## being deprecated in Version 1.1.7
with(dat, reSurv(Time, id, event, status))
## Use Recur() instead
simDat

Simulated dataset for demonstration

Description

A simulated data frame with the following variables

- **id**: subjects identification
- **t.start**: start of the interval
- **t.stop**: endpoint of the interval; when time origin is 0 this variable also marks the recurrence or terminal/censoring time
- **status**: terminal event indicator; 1 if a terminal event is recorded
- **event**: recurrent event indicator; 1 if a recurrent event is recorded
- **x1**: baseline covariate generated from a standard uniform distribution
- **x2**: baseline covariate generated from a standard uniform distribution (independent from z1)

Usage

data(simDat)

Format

A data frame with 874 rows and 7 variables.

Details

The sample dataset `simDat` is generated by `set.seed(0); dat <- simGSC(200)`. See `simGSC` for instruction on simulating recurrent event data from scale-change models.
simGSC

Function to generate simulated recurrent event data

Description

The function `simGSC()` generates simulated recurrent event data from either a Cox-type model, an accelerated mean model, an accelerated rate model, or a generalized scale-change model.

Usage

```r
simGSC(
  n,
  summary = FALSE,
  para,
  xmat,
  censoring,
  frailty,
  tau,
  origin,
  Lam0,
  Haz0
)
```

Arguments

- `n` number of observation.
- `summary` a logical value indicating whether a brief data summary will be printed.
- `para` a list of numerical vectors for the regression coefficients in the joint scale-change model. The names of the list elements are `alpha`, `beta`, `eta`, and `theta` correspond to \( \alpha \), \( \beta \), \( \eta \), and \( \theta \) in the joint scale-change model, respectively. See Details for `reReg`.
- `xmat` an optional matrix specifying the design matrix.
- `censoring` a numeric variable specifying the censoring times for each of the `n` observation.
- `frailty` a numeric variable specifying the frailty variable.
- `tau` a numeric value specifying the maximum observation time.
- `origin` a numeric value specifying the time origin.
- `Lam0` is an optional function that specifies the baseline cumulative rate function. When left-unspecified, the recurrent events are generated using the baseline rate function of

\[
\lambda_0(t) = \frac{2}{1 + t},
\]

or equivalently, the cumulative rate function of

\[
\Lambda_0(t) = 2 \log(1 + t).
\]
simGSC

is an optional function that specifies the baseline hazard function. When left-unspecified, the recurrent events are generated using the baseline hazard function

\[ h_0(t) = \frac{1}{5(1 + t)}. \]

or equivalently, the cumulative hazard function of

\[ H_0(t) = \log(1 + t)/5. \]

Details

The function `simGSC()` generates simulated recurrent event data over the interval \((0, \tau)\) based on the specification of the recurrent process and the terminal events. Specifically, the rate function, \(\lambda(t)\), of the recurrent process can be specified as one of the following model:

\[
\lambda(t) = Z\lambda_0(\text{e}^{X^\top \alpha})\text{e}^{X^\top \beta},
\]

where \(\lambda_0(t)\) is the baseline rate function, \(h_0(t)\) is the baseline hazard function, \(X\) is a \(n\) by \(p\) covariate matrix and \(\alpha, Z\) is an unobserved shared frailty variable, and \((\alpha, \eta)\) and \((\beta, \theta)\) correspond to the shape and size parameters of the rate function and the hazard function, respectively.

Under the default settings, the `simGSC()` function assumes \(p = 2\) and the regression parameters to be \(\alpha = \eta = (0, 0)^T\), and \(\beta = \theta = (1, 1)^T\). When the `xmat` argument is not specified, the `simGSC()` function assumes \(X_i\) is a two-dimensional vector \(X_i = (X_{i1}, X_{i2}), i = 1, \ldots, n\), where \(X_{i1}\) is a Bernoulli variable with rate 0.5 and \(X_{i2}\) is a standard normal variable. With the default `xmat`, the censoring time $C$ is generated from an independent uniform distribution in \([0, 2\tau X_{i1} + 2Z^2\tau(1 - X_{i1})]\). Thus, the censoring distribution is covariate dependent and is informative when \(Z\) is not a constant. When the `frailty` argument is not specified, the frailty variable \(Z\) is generated from a gamma distribution with a unit mean and a variance of 0.25. The default values for `tau` and `origin` are 60 and 0, respectively. When arguments `Lam0` and `Haz0` are left unspecified, the `simGSC()` function uses \(\Lambda_0(t) = 2\log(1 + t)\) and \(H_0(t) = \log(1 + t)/5\), respectively. This is equivalent to setting \(\Lambda_0 = \text{function}(x) 2 \times \log(1 + x)\) and \(H_0 = \text{function}(x) \log(1 + x) / 5\). Overall, the default specifications generate the recurrent events and the terminal events from the model:

\[
\lambda(t) = \frac{2Z}{1 + t\text{e}^{-X_{i1} - X_{i2}}}, h(t) = \frac{Z}{5(1 + t\text{e}^{X_{i1} + X_{i2}})}, t \in [0, 60].
\]

See Also

`reReg`

Examples

```R
set.seed(123)
simGSC(100, summary = TRUE)
```
The `summary.Recur` class is imported from `reda`.

**Description**

The class `summary.Recur` is an S4 that represents the summary of a `Recur` object. See `reda` for details.
## Index

**Plots**

- basebind, 3
- plot.Recur, 4
- plot.reReg, 6
- plotEvents, 8
- plotHaz, 10
- plotRate, 11
- %2% (Recur-pipe), 13
- %to% (Recur-pipe), 13
- %2%, 13
- `_PACKAGE` (reReg-package), 2

- basebind, 3

- is.Recur (Recur), 13

- mcf, 4, 5, 6
- mcf.formula-class, 4

- plot.Recur, 4, 9
- plot.reReg, 6, 11, 12
- plotEvents, 5, 6, 8
- plotHaz, 10
- plotRate, 11

- Recur, 6, 9, 13, 16
- Recur-class, 13
- Recur-pipe, 13
- reReg, 8, 11, 12, 14, 19, 20
- reReg-package, 2
- reReg-packages (reReg-package), 2
- residuals.reReg, 16
- reSurv, 17

- simDat, 18
- simGSC, 16, 18, 19
- summary.Recur-class, 21