Package 'reReg'

December 1, 2019

Title Recurrent Event Regression

Version 1.2.1

Description A collection of regression models for recurrent event process and failure time data. Available methods include these from Xu et al. (2017) <doi:10.1080/01621459.2016.1173557>, Lin et al. (2000) <doi:10.1111/1467-9868.00259>, Wang et al. (2001) <doi:10.1198/016214501753209031>, Ghosh and Lin (2003) <doi:10.1111/j.0006-341X.2003.00102.x>, and Huang and Wang (2004) <doi:10.1198/016214504000001033>.

Depends R (>= 3.5.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

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

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

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

RoxygenNote 6.1.1

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 2019-12-01 21:30:02 UTC

R topics documented:

`reReg-package` *reReg: Recurrent Event Regression*

Description

The package provides easy access to fit regression models to recurrent event data. The available implementations allow users to explore recurrent data through event plot and the cumulative sample mean function plot, fit semiparametric regression models under different assumptions, and simulate recurrent event data.

Author(s)

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

Authors:

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

References

Xu, G., Chiou, S.H., Huang, C.-Y., Wang, M.-C. and Yan, J. (2017). Joint Scale-change Models for Recurrent Events and Failure Time. *Journal of the American Statistical Association*, **112**(518): 796–805.

Lin, D., Wei, L., Yang, I. and Ying, Z. (2000). Semiparametric Regression for the Mean and Rate Functions of Recurrent Events. *Journal of the Royal Statistical Society: Series B (Methodological)*, **62**: 711–730.

Wang, M.-C., Qin, J., and Chiang, C.-T. (2001). Analyzing Recurrent Event Data with Informative Censoring. *Journal of the American Statistical Association*, **96**(455): 1057–1065.

Ghosh, D. and Lin, D.Y. (2003). Semiparametric Analysis of Recurrent Events Data in the Presence of Dependent Censoring. *Biometrics*, **59**: 877–885.

Huang, C.-Y. and Wang, M.-C. (2004). Joint Modeling and Estimation for Recurrent Event Processes and Failure Time Data. *Journal of the American Statistical Association*, **99**(468): 1153–1165.

See Also

Useful links:

- http://github.com/stc04003/reReg
- Report bugs at http://github.com/stc04003/reReg/issues

---

plot.Recur *Produce Event Plot or Cumulative Sample Mean Function Plot*

Description

Plot whether the event plot or the cumulative sample mean (CSM) function for an Recur object.

Usage

```
## S3 method for class 'Recur'
plot(x, CSM = FALSE, event.result = c("increasing",
  "decreasing", "asis"), csm.adjrisk = TRUE, csm.smooth = FALSE,
  control = list(), ...)
```

Arguments

x	an object of class Recur returned by the Recur() function.
CSM	an optional logical value indicating whether the cumulative sample mean (CSM) function will be plotted instead of the event plot (default).
event.result	an optional character string that is passed to the plotEvents() function as result, e.g., 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 increasing order (default). This places longer terminal times on top. decreasing sort the terminal time from in decreasing order (default). This places shorter terminal times on top. asis present the as is, without sorting.
csm.adjrisk	an optional logical value that is passed to the plotCSM() function as adjrisk, e.g., see plotCSM. This argument indicates whether risk set will be adjusted, e.g., if TRUE, subjects leave the risk set after terminal times.
csm.smooth	an optional logical value that is passed to the plotCSM() function as smooth, e.g., see plotCSM. This argument indicates whether to add a smooth curve obtained from a monotone increasing P-splines implemented in package scam.
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 CSM plot.

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

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

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

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

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

The `xlab`, `ylab` and `main` parameters can also be passed down without specifying a `control` list. See **Examples**.

Value

A `ggplot` object.

See Also

`Recur`

Examples

```
data(simDat)
reObj <- with(simDat, Recur(Time, id, event, status))

## Event plots:
plot(reObj)
plot(reObj, event.result = "decreasing")
plot(reObj, event.result = "asis")
plot(reObj, control = list(xlab = "User xlab", ylab = "User ylab", main = "User title"))

## CSM plots
plot(reObj, CSM = TRUE)
plot(reObj, CSM = TRUE, csm.adjrisk = FALSE)
plot(reObj, CSM = TRUE, csm.smooth = TRUE)
plot(reObj, CSM = TRUE, control = list(xlab = "User xlab", ylab = "User ylab", main = "User title"))

## With (hypothetical) multiple event types
set.seed(1)
reObj2 <- with(simDat, Recur(Time, id, event * sample(1:3, nrow(simDat), TRUE), status))
plot(reObj2)
plot(reObj2, event.result = "decreasing")
plot(reObj2, event.result = "asis")

plot(reObj2, CSM = TRUE)
```

```
plot(reObj2, CSM = TRUE, csm.adjrisk = FALSE)
plot(reObj2, CSM = TRUE, csm.smooth = TRUE)
```

`plot.reReg` *Plot the Baseline Cumulative Rate Function and the Baseline Cumulative Hazard Function*

Description

Plot the baseline cumulative rate function and the baseline cumulative hazard function (if applicable) for an `reReg` object. The 95% confidence intervals on the baseline cumulative rate function and the baseline cumulative hazard function are provided when the `reReg` object is fitted with standard error estimation.

Usage

```
## S3 method for class 'reReg'
plot(x, baseline = c("both", "rate", "hazard"),
  smooth = 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 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`.
`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 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.

See Also

reReg

Examples

```
data(simDat)
fm <- Recur(Time, id, event, status) ~ x1 + x2

fit <- reReg(fm, data = simDat, method = "cox.HW")
plot(fit)
plot(fit, baseline = "rate")
plot(fit, baseline = "rate", xlab = "Time (days)")
plot(fit, baseline = "rate", smooth = TRUE)
```

---

plotCSM *Produce Cumulative Sample Mean Function Plots*

Description

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

Usage

```
plotCSM(formula, data, onePanel = FALSE, adjrisk = TRUE,
  smooth = 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 returned by the Recur() function.
data	an optional data frame in which to interpret the variables occurring in the "formula".
onePanel	an optional logical value indicating whether the cumulative sample means (CSM) will be plotted in the same panel. This is only useful when there are multiple recurrent event types or in the presence of (discrete) covariates.
adjrisk	an optional logical value indicating whether risk set will be adjusted, e.g., if TRUE, subjects leave the risk set after terminal times. 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. This feature only works for data with one recurrent event type.

control	a list of control parameters.
...	graphical parameters to be passed to methods. These include xlab, ylab, main, and more. See **Details**.

Details

When adjrisk = TRUE, the plotCSM is equivalent to the Nelson-Aalen estimator for the intensity function of the recurrent event process. When adjrisk = FALSE, the plotCSM does not adjust for the risk set and assumes all subjects remain at risk after the last observed recurrent event. This is known as the survivor rate function. 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 "Cumulative mean".

main customizable title, default value is "Sample cumulative mean function plot".

The xlab, ylab and main parameters can also be passed down without specifying a control list.

Value

A ggplot object.

See Also

Recur, plot.Recur

Examples

```
data(simDat)
plotCSM(Recur(Time, id, event, status) ~ 1, data = simDat)
plotCSM(Recur(Time, id, event, status) ~ x1, data = simDat)
plotCSM(Recur(Time, id, event, status) ~ x1, data = simDat, onePanel = 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

```
plotEvents(formula, data, result = c("increasing", "decreasing", "asis"),
  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 increasing order (default). This places longer terminal times on top.
	decreasing sort the terminal time from in decreasing order (default). This places shorter terminal times on top.
	asis present the as is, without sorting.
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 "Subject".

main customizable title, default value is "Recurrent event plot".

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

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

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

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

The xlab, ylab and main parameters can also be passed down without specifying a control list. See **Examples**.

Value

A ggplot object.

See Also

Recur, plot.Recur

Examples

```
data(simDat)
plotEvents(Recur(Time, id, event, status) ~ 1, data = simDat)
plotEvents(Recur(Time, id, event, status) ~ 1, data = simDat,
           xlab = "Time in days", ylab = "Subjects arranged by terminal time")
```

```
## Separate plots by x1
plotEvents(Recur(Time, id, event, status) ~ x1, data = simDat)

## For multiple recurrent events
simDat$x3 <- ifelse(simDat$x2 < 0, "x2 < 0", "x2 > 0")
plotEvents(Recur(Time, id, event, status) ~ x1 + x3, data = simDat)
simDat$event <- simDat$event * sample(1:3, nrow(simDat), TRUE)
plotEvents(Recur(Time, id,  event, status) ~ x1, data = simDat)
plotEvents(Recur(Time, 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 hazard function is provided when the `reReg` object is fitted with standard error estimation.

Usage

```
plotHaz(x, smooth = FALSE, control = list(), ...)
```

Arguments

`x`	an object of class `reReg`, returned by the `reReg` function.
`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. See **Examples**.

Value

A `ggplot` object.

See Also

reReg plot.reReg

Examples

```
data(simDat)
fm <- Recur(Time, id, event, status) ~ x1 + x2

fit <- reReg(fm, data = simDat, method = "cox.HW")
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative hazard function
plotHaz(fit)
plotHaz(fit, smooth = TRUE)
## Plot with user-specified labels
plotHaz(fit, xlab = "User xlab", ylab = "User ylab", main = "User title")
plotHaz(fit, control = list(xlab = "User xlab", ylab = "User ylab", main = "User title"))
```

---

plotRate *Plotting the Baseline Cumulative Rate Function for the Recurrent Event Process*

Description

Plot the baseline rate function for an reReg object. The 95% confidence interval on the baseline cumulative rate function is provided when the reReg object is fitted with standard error estimation.

Usage

```
plotRate(x, smooth = FALSE, control = list(), ...)
```

Arguments

x	an object of class reReg, usually returned by the reReg function.
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 rate function".

These arguments can also be passed down without specifying a control list. See **Examples**.

Value

A ggplot object.

See Also

reReg plot.reReg

Examples

```
data(simDat)
fm <- Recur(Time, id, event, status) ~ x1 + x2

fit <- reReg(fm, data = simDat, method = "cox.HW")
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative rate function
plotRate(fit)
plotRate(fit, smooth = TRUE)
## Plot with user-specified labels
plotRate(fit, xlab = "User xlab", ylab = "User ylab", main = "User title")
plotRate(fit, control = list(xlab = "User xlab", ylab = "User ylab", main = "User title"))
```

Recur *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))
```

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 semiparametric regression model for the recurrent event data. The rate function of the underlying process for the recurrent event process can be specified as a Cox-type model, an accelerated mean model, or a generalized scale-change model. See details for model specifications.

Usage

```
reReg(formula, data, B = 200, method = c("cox.LWYY", "cox.GL",
  "cox.HW", "am.GL", "am.XCHWY", "sc.XCYH"), se = c("NULL", "bootstrap",
  "resampling"), contrasts = NULL, 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".
B	a numeric value specifies the number of resampling for variance estimation. When B = 0, variance estimation will not be performed.
method	a character string specifying the underlying model. See **Details**.
se	a character string specifying the method for standard error estimation. See **Details**.
contrasts	an optional list.
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 τ. We formulate the rate function, $\lambda(t)$, for the recurrent event process and the hazard function, $h(t)$, for the censoring time under the following model specifications:

Cox-type model:

$$\lambda(t) = Z\lambda_0(t)e^{X^\top \alpha}, h(t) = Zh_0(t)e^{X^\top \beta},$$

Accelerated mean model:

$$\lambda(t) = Z\lambda_0(te^{X^\top \alpha})e^{X^\top \alpha}, h(t) = Zh_0(te^{X^\top \beta})e^{X^\top \beta},$$

Scale-change model:

$$\lambda(t) = Z\lambda_0(te^{X^\top \alpha})e^{X^\top \beta},$$

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 α, Z is an unobserved shared frailty variable, and β are unknown p-dimensional regression parameters.

The `reReg` function fits models with the following available methods:

`method = "cox.LWYY"` assumes the Cox-type model with `Z = 1` and requires independent censoring. The returned result is equivalent to that from `coxph`. See reference Lin et al. (2000).

`method = "cox.HW"` assumes the Cox-type model with unspecified `Z`, thus accommodate informative censoring. See the references See reference Wang, Qin and Chiang (2001) and Huang and Wang (2004).

`method = "am.GL"` assumes the accelerated mean model with `Z = 1` and requires independent censoring. See the reference Ghosh and Lin (2003).

`method = "am.XCHWY"` assumes the accelerated mean model with unspecified `Z`, thus accommodate informative censoring. See the reference Xu et al. (2017).

`method = "sc.XCYH"` assumes the generalized scale-change model, and includes the methods `"cox.HW"` and `"am.XCHWY"` as special cases. Informative censoring is accounted for through the unspecified frailty variable `Z`. The methods also provide a hypothesis test of these submodels.

The available methods for variance estimation are:

`NULL` variance estimation will not be performed. This is equivalent to setting `B = 0`.

`"resampling"` performs the efficient resampling-based sandwich estimator that works with methods `"cox.HW"`, `"am.XCHWY"` and `"sc.XCYH"`.

`"bootstrap"` works with all fitting methods.

The `control` list consists of the following parameters:

`tol` absolute error tolerance.

`a0, b0` 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`.

`parallel` an logical value indicating whether parallel computation will be applied when `se = "bootstrap"` is called.

`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

Xu, G., Chiou, S.H., Huang, C.-Y., Wang, M.-C. and Yan, J. (2017). Joint Scale-change Models for Recurrent Events and Failure Time. *Journal of the American Statistical Association*, **112**(518): 796–805.

Lin, D., Wei, L., Yang, I. and Ying, Z. (2000). Semiparametric Regression for the Mean and Rate Functions of Recurrent Events. *Journal of the Royal Statistical Society: Series B (Methodological)*, **62**: 711–730.

Wang, M.-C., Qin, J., and Chiang, C.-T. (2001). Analyzing Recurrent Event Data with Informative Censoring. *Journal of the American Statistical Association*, **96**(455): 1057–1065.

Ghosh, D. and Lin, D.Y. (2003). Semiparametric Analysis of Recurrent Events Data in the Presence of Dependent Censoring. *Biometrics*, **59**: 877–885.

Huang, C.-Y. and Wang, M.-C. (2004). Joint Modeling and Estimation for Recurrent Event Processes and Failure Time Data. *Journal of the American Statistical Association*, **99**(468): 1153–1165.

See Also

Recur, simSC

Examples

```
fm <- Recur(Time, id, event, status) ~ x1 + x2

## Accelerated Mean Model
set.seed(1)
dat <- simSC(80, c(-1, 1), c(-1, 1), type = "am")
(fit <- reReg(Recur(Time, id, event, status) ~ x1 + x2,
              data = dat, method = "am.XCHWY", se = "resampling", B = 20))
summary(fit)

## Generalized Scale-Change Model
set.seed(1)
dat <- simSC(100, c(-1, 1), c(-1, 1), type = "sc")
(fit <- reReg(Recur(Time, id, event, status) ~ x1 + x2,
              data = dat, method = "sc.XCYH", se = "resampling", B = 20))
summary(fit)
```

---

reSurv *Create an* reSurv *Object*

Description

Create a recurrent event survival object, used as a response variable in reReg. This function is being 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 Verson 1.1.7
with(dat, reSurv(Time, id, event, status))
## Use Recur() instead
with(dat, Recur(Time, id, event, status))

## End(Not run)
```

---

simDat *Simulated dataset for demonstration*

Description

A simulated data frame with variables

id subjects identification

Time observation 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 274 rows and 6 variables.

Details

The sample dataset is generated by `set.seed(1); simSC(50,c(-1,1),c(-1,1))`. See `simSC` for instruction on simulating recurrent event data from scale-change models.

`simSC` *Function to generate simulated data*

Description

The function `simSC` generates simulated recurrent event data from either a Cox-type model, an accelerated mean model, or a scale-change model. The censoring time could be either independent (given covariates) or informative. The simulated data is used for illustration.

Usage

```
simSC(n, a, b, indCen = TRUE, type = c("cox", "am", "sc"), tau = 60,
  zVar = 0.25, summary = FALSE)
```

Arguments

`n`	number of observation.
`a`	a numeric vector of parameter of length 2.
`b`	a numeric vector of parameter of length 2.
`indCen`	a logical value indicating whether the censoring assumption is imposed. When `indCen = TRUE`, we set $Z = 1$. Otherwise, Z is generated from a unit-mean gamma distribution See **Details**.
`type`	a character string specifying the underlying model. See **Details**
`tau`	a numeric value specifying the maximum observation time.
`zVar`	a numeric variable specifying the variance of Z. This is only needed when $indCen$ is TRUE. The default value is 0.25.
`summary`	a logical value indicating whether a brief data summary will be printed.

Details

The function `simSC` generates simulated recurrent event data under different scenarios based on the following assumptions. See **Details** in `reReg` for a more complete model assumptions.

`type = "cox"` generates recurrent event data from a Cox-type model with

$$\lambda(t) = Z\lambda_0(t)e^{X^\top a}, h(t) = Zh_0(t)e^{X^\top b}.$$

`type = "am"` generates recurrent event data from an accelerated mean model with

$$\lambda(t) = Z\lambda_0(te^{X^\top a})e^{X^\top a}, h(t) = Zh_0(te^{X^\top b})e^{X^\top b}.$$

`type = "sc"` generates recurrent event data from a generalized scale-change model with

$$\lambda(t) = Z\lambda_0(te^{X^\top a})e^{X^\top b}, h(t) = Zh_0(te^{X^\top a})e^{X^\top b}.$$

Let D be the informative failure time with the above hazard function. An non-informative failure time, C, is generated separately from an exponential distribution with mean 80. The observed follow-up time is then taken to be $min(D, C, \tau)$. We further assume

$$\lambda_0(t) = \frac{2}{1+t}, h_0(t) = \frac{1}{8(1+t)}.$$

Two covariates are considered; `x1` follows a Bernoulli distribution with probability 0.5 and `x2` follows a standard normal distribution.

See Also

`reReg`

Examples

```
set.seed(123)
simSC(200, c(-1, 1), c(-1, 1), summary = TRUE)
```

Index