Package: reReg (via r-universe)

September 16, 2024

Title Recurrent Event Regression

Version 1.4.6

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 described in Chiou et al. (2023) <doi:10.18637/jss.v105.i05>. 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 (>= 4.2.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

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

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

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

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.1

NeedsCompilation yes

Repository https://stc04003.r-universe.dev

RemoteUrl https://github.com/stc04003/rereg

RemoteRef HEAD

RemoteSha 78f09d8122129e449c30343c67cacd0346ab680d

Contents

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

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 includes tools for visualization of recurrent events and simulation from the regression models.

Author(s)

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

Authors:

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

References

Chiou, S.H., Xu, G., Yan, J. and Huang, C.-Y. (2023). Regression Modeling for Recurrent Events Possibly with an Informative Terminal Event Using R Package reReg. *Journal of Statistical Software*, **105**(5): 1–34.

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. (2002). Marginal Regression Models for Recurrent and Terminal Events. *Statistica Sinica*: 663–688.

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.

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.

Xu, G., Chiou, S.H., Yan, J., Marr, K., and Huang, C.-Y. (2019). Generalized Scale-Change Models for Recurrent Event Processes under Informative Censoring. *Statistica Sinica*, **30**: 1773–1795.

See Also

Useful links:

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

`basebind` *Function used to combine baseline functions in one plot*

Description

Combine different plots into one.

Usage

```
basebind(..., legend.title, legend.labels, control = list())
```

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.
`control`	a list of control parameters.

Examples

```
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))
```

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

```
## 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

Nelson, W. B. (1995) Confidence Limits for Recurrence Data-Applied to Cost or Number of Product Repairs. *Technometrics*, **37**(2): 147–157.

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 <- simDat
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)



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

`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.

Usage

```
## 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.

See Also

`reReg`

Examples

```
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

```
plotEvents(
  formula,
  data,
  result = c("increasing", "decreasing", "asis"),
  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)
```

`plotEvents.control` *Plot options for plotEvents*

Description

This function provides the plotting options for the `plotEvents()` function.

Usage

```
plotEvents.control(
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  terminal.name = NULL,
```

```
  recurrent.name = NULL,
  recurrent.type = NULL,
  legend.position = NULL,
  base_size = 12,
  cex = NULL,
  alpha = 0.7,
  width = NULL,
  bar.color = NULL,
  recurrent.color = NULL,
  recurrent.shape = NULL,
  recurrent.stroke = NULL,
  terminal.color = NULL,
  terminal.shape = NULL,
  terminal.stroke = NULL,
  not.terminal.color = NULL,
  not.terminal.shape = NULL
)
```

Arguments

xlab	a character string indicating the label for the x axis. The default value is "Time".
ylab	a character string indicating the label for the y axis. The default value is "Subject".
main	a character string indicating the title of the plot.
terminal.name	a character string indicating the label for the terminal event displayed in the legend. The default value is "Terminal event".
recurrent.name	a character string indicating the label for the recurrent event displayed in the legend. The default value is "Recurrent events".
recurrent.type	a factor indicating the labels for the different recurrent event types. This option is only available when there are more than one types of recurrent events. The default value is "Recurrent events 1", "Recurrent events 2", ...
legend.position	a character string specifies the position of the legend. The available options are "none", "left", "right", "bottom", "top", "bottomright", "bottomleft","topleft", "topright", or a two-element numeric vector specifies the coordinate of the legend. The legend is placed inside of the plotting region when `legend.position` is specified as one of "bottomright", "bottomleft","topleft", "topright". Otherwise, the argument is passed to the `ggplot` theme environment.
base_size	a numerical value to specify the base font size, given in pts. This argument is passed to the `ggplot` theme environment. The default value is 12.
cex	a numerical value specifies the size of the points.
alpha	a numerical value specifies the transparency of the points.
width	a numerical value specifies the width of the event plot. By `ggplot` default, set to 90% of the resolution of the data.
bar.color	a numerical value or a character string specifies color for lines. Default to gray.

`recurrent.color`
: a numerical value or a character string specifies color for recurrent events. Default to green.

`recurrent.shape`
: a numerical value or a character string specifies shape for recurrent events. Default to circle.

`recurrent.stroke`
: a numerical value or a character string specifies stroke for recurrent events. Default to circle.

`terminal.color`
: a numerical value or a character string specifies color for terminal events. Default to red.

`terminal.shape`
: a numerical value or a character string specifies shape for terminal events. Default to triangle.

`terminal.stroke`
: a numerical value or a character string specifies stroke for terminal events. Default to triangle.

`not.terminal.color`
: a numerical value or a character string specifies color for non-terminal events. Non-terminal events are not plotted at default.

`not.terminal.shape`
: a numerical value or a character string specifies shape for terminal events. Non-terminal events are not plotted at default.

See Also

`plotEvents`

`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.

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)
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 rate function
plotRate(fit)
plotRate(fit, smooth = TRUE)
```

---

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

```
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. The available functional form for the rate function and the hazard function include a Cox-type model, an accelerated mean model, an accelerated rate model, or a generalized scale-change model, and can be specified via "cox", "am", "ar", or "gsc", respectively. The rate function and hazard function separated by "|". 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. See reReg.control for default values.

Details

Model specification:

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 recurrent event rate function, $\lambda(t)$, and the terminal event hazard function, $h(t)$, in the form of

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

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 (α, η) and (β, θ) 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 the 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). Additionally, an improved estimation of the proportional rate model (Huang and Huang 2022) can be called by model = "cox.HH" with additional control options to specify the underlying procedure. See online vignette for a detailed discussion of the implemented regression models.

Variance estimation:

The available methods for variance estimation are:

boot performs nonparametric bootstrap.

sand performs the efficient resampling-based variance estimation.

Improving proportional rate model: A common semiparametric regression model for recurrent event process under the noninformative censoring assumption is the Cox-type proportional rate model (available in `reReg()` via `model = "cox.LWYY"`). However, the construction of the pseudo-partial score function ignores the dependency among recurrent events and thus could be inefficient. To improve upon this popular method, Huang and Huang (2022) proposed to combine a system of weighted pseudo-partial score equations via the generalized method of moments (GMM) and empirical likelihood (EL) estimation. The proposed GMM and EL procedures are available in `reReg` via `model = "cox.HH"` with additional control specifications. See online vignette for an illustration of this feature.

Control options:

The `control` list consists of the following parameters:

tol absolute error tolerance.

init a list contains initial guesses used for root search.

solver the equation solver used for root search. The available options are `BB::BBsolve`, `BB::dfsane`, `BB::BBoptim`, `optimx::optimr`, `dfoptim::hjk`, `dfoptim::mads`, `optim`, and `nleqslv::nleqslv`.

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 an 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.

cppl A character string indicating either to improve the proportional rate model via the generalized method of moments (`cppl = "GMM"`) or empirical likelihood estimation (`cppl = "EL"`). This option is only used when `model = "cox.HH"`.

cppl.wfun A list of (up to two) weight functions to be combined with the weighted pseudo-partial likelihood scores. Available options are `"Gehan"` and `"cumbase"`, which correspond to the Gehan's weight and the cumulative baseline hazard function, respectively. Alternatively, the weight functions can be specified with function formulas. This option is only used when `model = "cox.HH"`.

trace A logical variable denoting whether some of the intermediate results of iterations should be displayed to the user. Default is `FALSE`.

References

Chiou, S.H., Xu, G., Yan, J. and Huang, C.-Y. (2023). Regression Modeling for Recurrent Events Possibly with an Informative Terminal Event Using R Package reReg. *Journal of Statistical Software*, **105**(5): 1–34.

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. (2002). Marginal Regression Models for Recurrent and Terminal Events. *Statistica Sinica*: 663–688.

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.

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.

Xu, G., Chiou, S.H., Yan, J., Marr, K., and Huang, C.-Y. (2019). Generalized Scale-Change Models for Recurrent Event Processes under Informative Censoring. *Statistica Sinica*, **30**: 1773–1795.

Huang, M.-Y. and Huang, C.-Y. (2022). Improved semiparametric estimation of the proportional rate model with recurrent event data. *Biometrics*, **79** 3: 1686–1700.

See Also

Recur, simGSC

Examples

```
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"))
```

---

reReg.control *Package options for reReg*

Description

This function provides the fitting options for the `reReg()` function.

Usage

```
reReg.control(
  eqType = c("logrank", "gehan", "gehan_s"),
  solver = c("BB::dfsane", "BB::BBsolve", "BB::BBoptim", "optimx::optimr",
    "dfoptim::hjk", "dfoptim::mads", "optim", "nleqslv::nleqslv"),
  tol = 1e-07,
  cppl = NULL,
  cppl.wfun = list(NULL, NULL),
  init = list(alpha = 0, beta = 0, eta = 0, theta = 0),
  boot.parallel = FALSE,
  boot.parCl = NULL,
  maxit1 = 100,
  maxit2 = 10,
  trace = FALSE,
  numAdj = 1e-07
)
```

Arguments

eqType	a character string indicating whether the log-rank type estimating equation or the Gehan-type estimating equation (when available) will be used.
solver	a character string specifying the equation solver to be used for root search.
tol	a positive numerical value specifying the absolute error tolerance in root search.
cppl	a character string indicating either to improve the proportional rate model via the generalized method of moments (`cppl = "GMM"`) or empirical likelihood estimation (`cppl = "EL"`). This option is only used when `model = "cox.HH"`.
cppl.wfun	a list of (up to two) weight functions to be combined with the weighted pseudo-partial likelihood scores. Available options are `"Gehan"` and `"cumbase"`, which correspond to the Gehan's weight and the cumulative baseline hazard function, respectively. Alternatively, the weight functions can be specified with function formulas. This option is only used when `model = "cox.HH"`.
init	a list contains the initial guesses used for root search.
boot.parallel	an logical value indicating whether parallel computation will be applied when `se = "boot"` is specified in `reReg()`.
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.
maxit1, maxit2	max number of iteration used when `model = "cox.HH"`.
trace	a logical variable denoting whether some of the intermediate results of iterations should be displayed to the user. Default is `FALSE`.
numAdj	a positive numerical value specifying the small constant used in heuristic adjustment of the borrow strength method.

See Also

reReg

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 Δ is the terminal event indicator and $H(t)$ is the cumulative hazard function at t.

Usage

```
## 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.

reSurv *Create an* reSurv *Object*

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 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 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

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

```
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 α, β, η, and θ 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

$$\Lambda_0(t) = 2\log(1+t).$$

`Haz0` 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:

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

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 (α, η) and (β, θ) 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 $\alpha = \eta = (0, 0)^\top$, and $\beta = \theta = (1, 1)^\top$. 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 exponential distribution with mean $\tau X_{i1} + Z^2\tau(1 - X_{i1})$. 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 `Lam0 = function(x) 2 * log(1 + x)` and `Haz0 = function(x) log(1 + x) / 5`. Overall, the default specifications generate the recurrent events and the terminal events from the model:

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

See online vignette for more examples.

See Also

reReg

Examples

```
set.seed(123)
simGSC(100, summary = TRUE)
```

Index