# Package 'reReg'

May 30, 2018

**Title** Recurrent Event Regression

**Version** 1.1.4

**Description** A collection of regression models for recurrent event process and failure time. 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 (>= 3.4.0)

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Imports** BB (>= 2014.10.1), nleqslv (>= 3.3.1), SQUAREM (>= 2017.10.1), survival (>= 2.41.3), plyr (>= 1.8.4), ggplot2 (>= 2.2.1), purrr (>= 0.2.4), dplyr (>= 0.7.4), tidyr (>= 0.8.0), MASS (>= 7.3.49), methods

**RoxygenNote** 6.0.1

**NeedsCompilation** yes

**Author** Sy Han (Steven) Chiou [aut, cre]

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

**Repository** CRAN

**Date/Publication** 2018-05-30 17:51:38 UTC

## R topics documented:

---

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

---

## Description

Recurrent event data arise frequently in various fields such as biomedical sciences, public health, engineering, and social sciences. In many instances, the observation of the recurrent event process can be stopped by the occurrence of a correlated failure event, such as death or treatment failure. As such, we implement a collection of regression models for recurrent event process and failure time.

## Author(s)

**Maintainer**: Sy Han (Steven) Chiou <schiou@utdallas.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 D.Y. Lin (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.

---

`plot.reReg` *Plotting Estimated Baseline Cumulative Rate Functions*

---

## Description

Plot the estimated baseline cumulative rate function and harzard function if available.

## Usage

```
## S3 method for class 'reReg'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | an object of class reReg, usually returned by the reReg function. |
| ... | for future methods. |

### See Also

reReg

### Examples

```
data(readmission)
fit <- reReg(reSurv(t.stop, event, death, id) ~ sex + chemo,
             data = subset(readmission, id < 50),
             method = "am.XCHWY", se = "resampling", B = 20)
plot(fit)
```

---

## plot.reSurv *Produce Event Plots*

---

### Description

Plot the event plot for an reSurv object.

### Usage

```
## S3 method for class 'reSurv'
plot(x, data, order = TRUE, return.grob = FALSE,
  control = list(), ...)
```

### Arguments

| | |
|---|---|
| x | an object of class reSurv, usually returned by the reSurv function. |
| data | an optional data frame in which to interpret the variables occurring in the "formula". |
| order | an optional logical value. If "TRUE", the event plot is sorted by the terminal times; the default value is TRUE. |
| return.grob | an optional logical value. If "TRUE", a ggplot2 plot grob will be returned. |
| control | a list of control parameters. |
| ... | for future developments. |

## 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".

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

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

**recurrent.name** customizable label for recurrent event, default value is "Recurrent event plot".

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

**alpha** controls the transparency of points.

## See Also

`reSurv`

## Examples

```
data(readmission)
reObj <- with(subset(readmission, id <= 10), reSurv(t.stop, event, death, id))
## Default labels
plot(reObj)
plot(reObj, order = FALSE)
## User specified labels
plot(reObj, control = list(xlab = "User xlab", ylab = "User ylab", title = "User title"))

## With multiple hypothetical event types
set.seed(1)
reObj2 <- with(readmission, reSurv(t.stop, event * sample(1:3, 861, TRUE), death, id))
plot(reObj2)

## With multiple hypothetical event types
set.seed(1)
reObj2 <- with(readmission, reSurv(t.stop, event * sample(1:3, 861, TRUE), death, id))
plot(reObj2)
```

---

# plotEvents *Produce Event Plots*

---

## Description

Plot the event plot for an `reSurv` object, with more flexible options.

## Usage

```
plotEvents(formula, data, order = TRUE, return.grob = FALSE,
  control = list(), ...)
```

## Arguments

| | |
|---|---|
| formula | a formula object, with the response on the left of a "~" operator, and the terms on the right. The response must be a recurrent event survival object as returned by function reSurv. |
| data | an optional data frame in which to interpret the variables occurring in the "formula". |
| order | an optional logical value. If "TRUE", the event plot is sorted by the terminal times; the default value is TRUE. |
| return.grob | an optional logical value. If "TRUE", a ggplot2 plot grob will be returned. |
| control | a list of control parameters. |
| ... | for future developments. |

## 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".

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

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

**recurrent.name** customizable label for recurrent event, default value is "Recurrent event plot".

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

**alpha** controls the transparency of points.

## See Also

reSurv, plot.reSurv

## Examples

```
data(readmission)
plotEvents(reSurv(t.stop, event, death, id) ~ 1, data = readmission)

## Separate plots by gender
plotEvents(reSurv(t.stop, event, death, id) ~ sex, data = readmission)

## Separate plots by gender and chemo type
plotEvents(reSurv(t.stop, event, death, id) ~ sex + chemo, data = readmission)

## With multiple hypothetical event types
plotEvents(reSurv(t.stop, event * sample(1:3, 861, TRUE), death, id) ~
  sex + chemo, data = readmission)
```

plotHaz *Plotting the baseline hazard function*

## Description

Plot the baseline hazard function after fitting reReg.

## Usage

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

## Arguments

| | |
|---|---|
| x | an object of class reReg, usually returned by the reReg function. |
| control | a list of control parameters. |
| ... | for future developments. |

## 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".

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

## See Also

reReg

## Examples

```
## readmission data
data(readmission)
set.seed(123)
fit <- reReg(reSurv(t.stop, event, death, id) ~ sex + chemo,
             data = subset(readmission, id < 50),
             method = "am.XCHWY", se = "resampling", B = 20)
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative hazard function
plotHaz(fit)
## Plot with user-specified labels
plotHaz(fit, control = list(xlab = "User xlab", ylab = "User ylab", title = "User title"))
```

plotRate *Plotting the baseline rate function*

## Description

Plot the baseline rate function after fitting reReg.

## Usage

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

## Arguments

| | |
|---|---|
| x | an object of class reReg, usually returned by the reReg function. |
| control | a list of control parameters. |
| ... | for future developments. |

## 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".

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

## See Also

reReg

## Examples

```
## readmission data
data(readmission)
set.seed(123)
fit <- reReg(reSurv(t.stop, event, death, id) ~ sex + chemo,
                  data = subset(readmission, id < 50),
                  method = "am.XCHWY", se = "resampling", B = 20)
## Plot both the baseline cumulative rate and hazard function
plot(fit)
## Plot baseline cumulative rate function
plotRate(fit)
## Plot with user-specified labels
plotRate(fit, control = list(xlab = "User xlab", ylab = "User ylab", title = "User title"))
```

`readmission` *Rehospitalization Colorectal Cancer*

## Description

This contains rehospitalization times after surgery in patients diagnosed with colorectal cancer.

## Usage

```
data(readmission)
```

## Format

A `data.frame` contains the following columns:

**id** identificator of each subject (repeated for each recurrence).

**enum** observation number within patient.

**t.start** start of interval (0 or previous recurrence time).

**t.stop** recurrence or censoring time.

**time** interocurrence or censoring time.

**event** rehospitalization status. All event are 1 for each subject excepting last one that it is 0.

**chemo** Treatment (chemotherapy) indicator.

**sex** Gender: Male or Female.

**dukes** Tumour stage under Dukes's classification: A-B, C, or D.

**charlson** Comorbidity Charlson's index, modelled as a time dependent covariate and classified into three groups: Index 0, Index 1-2, and Index >= 3.

**death** death indicator: Dead = 1; alive = 0.

## Source

Gonzalez, J.R., Fernandez, E., Moreno, V., Ribes, J., Peris, M., Navarro, M., Cambray, M. and Borras, JM (2005). Sex differences in hospital readmission among colorectal cancer patients. *Journal of Epidemiology and Community Health*, **59**, 6, 506-511.

reReg *Fits Semiparametric Regression Models for Recurrent Events and Failure Time*

## Description

Fits a survival 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, an accelerated rate model, or a generalized scale-change model. When a joint model is fitted (e.g., `method = "cox.HW"` or `method = "am.XCHWY"`), the hazard function of the terminal event is either in a Cox model or an accelerated failure time model.

## Usage

```
reReg(formula, data, B = 200, method = c("cox.LWYY", "cox.HW", "am.GL",
  "am.XCHWY", "sc.XCYH"), se = c("NULL", "bootstrap", "resampling"),
  plot.ci = FALSE, contrasts = NULL, control = list())
```

## Arguments

| | |
|---|---|
| `formula` | a formula object, with the response on the left of a "~" operator, and the terms on the right. The response must be a recurrent event survival object as returned by function `reSurv`. |
| `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, only the point estimates will be displayed. |
| `method` | a character string specifying the underlying model. See `Details`. |
| `se` | a character string specifying the method for standard error estimation. See `Details`. |
| `plot.ci` | a logical value indicating whether the 95 and/or the estimated cumulative hazard function should be computed. Default is "FALSE". |
| `contrasts` | an optional list. |
| `control` | a list of control parameters. |

## Details

The underlying models and assumptions are different for different methods. The available methods are:

`method == "cox.LWYY"` models the rate function for the recurrent event through a Cox-type model. This function returns results equivalent to that of `coxph`. See reference Lin et al. (2000).

`method == "cox.HW"` jointly model the recurrent events and failure time. This method assumes a Cox-type model for both the intensity function of the recurrent event process and the hazard function of the failure time. Informative censoring is accounted for via a shared frailty variable. See the references See reference Wang, Qin and Chiang(2001) and Huang and Wang (2004).

`method == "am.GL"` jointly model the recurrent events and failure time. This method assumes an accelerated mean model for both the recurrent event process and the failure time. This method uses artificial censoring to allow for an unspecified association between the two types of outcomes. Informative censoring is not allowed. See the reference Ghosh and Lin (2003).

`method == "am.XCHWY"` jointly model the recurrent events and failure time. This method assumes an accelerated mean model for both the recurrent event process and the failure time. Informative censoring is accounted for via a shared frailty variable. See the reference Xu et al. (2017).

`method == "sc.XCYH"` models the rate function for the recurrent events via a generalized scale-change model that includes Cox-type models, accelerated mean models, and accelerated rate models as special cases. The methods also provide a hypothesis test of these submodels. Informative censoring is accounted for via a shared frailty variable.

For available method for standard errors estimation are:

**NULL** means do not calculate standard errors.

`se == "resampling"` performs the efficient resampling-based sandwich estimator that works with `method == "am.XCHWY"` and `method == "sc.XCYH"`.

`"bootstrap"` works with all fitting methods.

## 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 D.Y. Lin (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

`reSurv`

## Examples

```
## readmission data
data(readmission)
set.seed(123)
## Acceralted Mean Model
(fit <- reReg(reSurv(t.stop, event, death, id) ~ sex + chemo,
              data = subset(readmission, id < 50),
              method = "am.XCHWY", se = "resampling", B = 20))
```

```
summary(fit)

## Generalized Scale-Change Model
set.seed(123)
(fit <- reReg(reSurv(t.stop, event, death, id) ~ sex + chemo,
              data = subset(readmission, id < 50),
              method = "sc.XCYH", se = "resampling", B = 20))
summary(fit)

## Not run:
## simulation data
simDat <- function(n, a, b, latent = FALSE) {
    ## setting rate function
    Lam.f <- function(t, z, x, w) .5 * z * exp(-x + w) * log(1 + t * exp(x))
    Lam.f0 <- function(t) .5 * log(1 + t)
    invLam.f  <- function(t, z, x, w) (exp((2 / z) * exp(x - w) * t )- 1) / exp(x)
    ## setting hazard funciton
    ## Haz.f0 <- function(t) .5 * log(1 + t) # assume constant hazard for now
    dat <- NULL
    for (id in 1:n) {
        z <- ifelse(latent, rgamma(1, 4, 4), 1)
        x1 <- rnorm(1)
        x2 <- rnorm(1)
        x <- c(x1, x2)
        cen <- rexp(1, z * exp(x %*% b) / 60) ## this gives constant hazard of 1/60
        y <- min(cen, 60)
        D <- 1 * (cen == y)
        tmpt <- NULL
        while(sum(tmpt) < Lam.f(y, z, c(x %*% a), c(x %*% b))) {
            tmpt <- c(tmpt, rexp(1))
        }
        m <- length(tmpt) - 1
        if (m > 0) {
            tt <- invLam.f(cumsum(tmpt[1:m]), z, c(x %*% a), c(x %*% b))
            dat <- rbind(dat, cbind(ID = id, Time = c(tt[order(tt)], y),
                                    event = c(rep(1, m), 0), status = c(rep(0, m), D),
                                    Z = z, M = m, X1 = x1, X2 = x2))
        } else {
            dat <- rbind(dat, cbind(ID = id, Time = y, event = 0, status = D,
                                    Z = z, M = m, X1 = x1, X2 = x2))
        }
    }
    return(data.frame(dat))
}
set.seed(2017)
dat <- simDat(200, c(0, 0), c(0, 0), TRUE) ## generate data under informative censoring
fm <- reSurv(Time, event, status, ID) ~ X1 + X2
fit.HW <- reReg(fm, data = dat, method = "cox.HW", B = 5)

## End(Not run)
```

reSurv *Create an* reSurv *Object*

## Description

Create a recurrent event survival object, used as a response variable in reReg model formula.

## Usage

```
reSurv(time1, time2, event, status, id)

is.reSurv(x)
```

## 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. |
| event | a binary vector used as the recurrent event indicator. |
| status | a binary vector used as the status indicator for the terminal event. |
| id | observation subject's id |
| x | an reSurv object. |

## Examples

```
data(readmission)
with(readmission, reSurv(t.stop, event, death, id))
with(readmission, reSurv(t.start, t.stop, event, death, id))
```

# Index