Package ‘frailtySurv’

August 28, 2018

Title General Semiparametric Shared Frailty Model
Priority optional
Type Package
Version 1.3.5
Date 2018-08-27
Maintainer Vinnie Monaco <contact@vmonaco.com>
Description Simulates and fits semiparametric shared frailty models under a wide range of frailty distributions using a consistent and asymptotically-normal estimator. Currently supports: gamma, power variance function, log-normal, and inverse Gaussian frailty models.
License LGPL-2
URL https://github.com/vmonaco/frailtySurv/
BugReports https://github.com/vmonaco/frailtySurv/issues
Depends R (>= 3.0.0), survival
Imports stats, nleqslv, reshape2, ggplot2, numDeriv
Suggests knitr, parallel, gridExtra
NeedsCompilation yes
LinkingTo Rcpp
LazyData Yes
LazyLoad Yes
ByteCompile Yes
Repository CRAN
RoxygenNote 5.0.1
Author Vinnie Monaco [aut, cre], Malka Gorfine [aut], Li Hsu [aut]
Date/Publication 2018-08-28 21:24:33 UTC
Description

The Diabetic Retinopathy Study (DRS) was performed to determine whether the onset of blindness in 197 high-risk diabetic patients could be delayed by laser treatment. The treatment was administered to one randomly-selected eye in each patient, leaving the other eye untreated. Thus, there are 394 observations, which are clustered by patient since the level of risk will tend to vary between patients. A failure occurred when visual acuity dropped to below 5/200. All patients had a visual acuity of at least 20/100 at the beginning of the study.

Usage

data("drs")

Format

A data frame with 394 rows and 8 columns. There are two rows for each subject, one row for each eye:

- **subject_id**: unique identifier for each subject
- **eye**: subject’s eye, where 1=right and 2=left
- **time**: the observed follow-up time
- **status**: outcome at the end of the observation period, where 1=blindness and 0 indicates censorship
- **treated**: a binary covariate, where 1=treated or 0=untreated
- **age_at_onset**: age (in years) at the onset of diabetes
- **laser_type**: type of laser used for treatment, where 1=xenon, 2=argon
- **diabetes_type**: type of diabetes, where 1=juvenile (age at dx < 20) and 2=adult
fitfrail

Source

http://www.mayo.edu/research/documents/diabeteshtml/doc-10027460

Examples

## Not run:
data(drs)

# Clustered by subject
fit.drs <- fitfrail(Surv(time, status) ~ treated + cluster(subject_id),
                   drs, frailty="gamma")

fit.drs

# Variance estimates
vcov(fit.drs)

# Plot the estimated cumulative baseline hazard
plot(fit.drs, type="cumhaz")

## End(Not run)

fitfrail Fit a shared frailty model

Description

Fit an extended Cox proportional hazards model with unobserved shared frailty variate and unspecified baseline hazard function, using a semiparametric estimation technique. See Gorfine et al.~(2006) and Zucker et al.~(2008) for details.

Usage

fitfrail(formula, dat, control, frailty, weights = NULL, se = FALSE, ...)

Arguments

formula a formula object, where the lhs is the response as a Surv object and rhs contain the terms, including a cluster term for the cluster identifier
dat data.frame that provides context for the formula
control control parameters in the form of a fitfrail.control object
frailty string name of the shared frailty distribution
weights vector of cluster weights
se logical value, whether the standard errors of the regression coefficient and frailty distribution parameter estimates should be calculated. These are obtained using the vcov.fitfrail function
... additional arguments will be passed to fitfrail.control
Value

A fitfrail object representing the shared frailty model.

- **beta**: the estimated regression coefficients
- **theta**: the estimated frailty distribution parameters
- **Lambda**: a data.frame with the estimated baseline hazard at each failure time
- **Lambda**: a data.frame with the estimated baseline hazard at all observed times
- **Lambda.fun**: a function of time that returns the estimated baseline
- **loglik**: the log-likelihood
- **iter**: the number of iterations performed
- **trace**: the parameter trace during estimation

Convergence

The initial values of the regression coefficients are provided by coxph. Convergence is reached when either the relative reduction or absolute reduction in loglikelihood or score equations (depending on the fitmethod used) are below a threshold. If the maxit iterations are performed before convergence, then the algorithm terminates with a warning.

Author(s)

The estimation method was developed by Malka Gorfine, Li Hsu, and David Zucker; implemented by John V. Monaco.

References


See Also

- vcov.fitfrail, genfrail, simfrail, survfit, coxph

Examples

```r
## Not run:
#
## Generate synthetic survival data with regression coefficients
## beta = c(log(2),log(3)) and theta = 2, where the shared frailty
## values from a gamma distribution with expectation 1 and variance theta.
##
dat <- genfrail(N=300, K=2, beta=c(log(2),log(3)),
```
fitfrail.control

Control parameters for fitfrail

Description

This function creates a list of control parameters needed by fitfrail.

Usage

fitfrail.control(fitmethod = "loglik",
abstol = 0, reltol = 1e-6, maxit = 100,
int.abstol = 0, int.reltol = 1, int.maxit = 1000,
init.beta="coxph", init.theta=NULL, verbose = FALSE)
Arguments

fitmethod string indicating which fit method should be used: "loglik" or "score". If "loglik", then the loglikelihood is maximized directly using optim (L-BFGS-B algorithm). If "score", then the system of normalized score equations is solved using nleqslv (Newton algorithm).

abstol numeric absolute tolerance for convergence. If fitmethod is "loglik", convergence is reached when the absolute reduction in loglikelihood is less than abstol. If fitmethod is "score", convergence is reached when the absolute value of each normalized score equation is less then abstol.

reltol numeric relative tolerance for convergence. If fitmethod is "loglik", convergence is reached when the relative reduction in loglikelihood is less than reltol. If fitmethod is "score", convergence is reached when the relative reduction of each estimated parameter is less than reltol.

maxit integer, maximum number of iterations to perform

int.abstol numeric absolute tolerance for convergence of the numeric integration. Only applicable for frailty distributions that require numerical integration. If 0, then ignore. Default is 0.

int.reltol numeric relative tolerance for convergence of the numeric integration. Only applicable for frailty distributions that require numerical integration. If 0, then ignore. Default is 1.

int.maxit integer, maximum number of numeric integration function evaluations. Only applicable for frailty distributions that require numerical integration. If 0, then no limit. Default is 100.

init.beta initial regression coefficients parameter estimates, can be numeric, string, or NULL. If NULL, then a zero vector is used. If "coxph" is passed, then regression coefficients are initialized to the estimates given by coxph (with or without frailty, depending on the init.theta parameter, see below). If numeric, then the initial regression coefficients are specified explicitly.

init.theta initial frailty distribution parameter estimates, can be numeric, string, or NULL. If NULL, then a sensible starting value is chosen for the specified frailty distribution. If "coxph", then hat.theta is initialized such that the initial frailty distribution has the same rank correlation as the estimated gamma frailty distribution. This is achieved by assuming gamma frailty and estimating theta using coxph. This estimate is then transferred to the appropriate frailty distribution through Kendall’s tau. This method only works well for weak cluster dependence, i.e., small Kendall’s tau. If numeric, then the initial frailty distribution parameters are specified explicitly.

verbose logical value, whether to print a trace of the parameter estimation.

Value

A list of control parameters.

Author(s)

John V. Monaco, Malka Gorfine, Li Hsu
Description

frailtySurv provides a suite of functions for generating clustered survival data, fitting multivariate shared frailty models under a wide range of frailty distributions, and visualizing the output. The semi-parametric estimators have better asymptotic properties than most existing implementations, including consistent and asymptotically-normal estimators. Moreover, this is the first package that implements semi-parametric estimators with inverse Gaussian and PVF frailty models.

The frailtySurv package provides functions

- genfrail for data generation
- fitfrail for model fitting
- simfrail for survival simulation

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>frailtySurv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.2.0</td>
</tr>
<tr>
<td>Date:</td>
<td>August 2015</td>
</tr>
<tr>
<td>License:</td>
<td>LGPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Author(s)

John V. Monaco, Malka Gorfine, and Li Hsu.

References


See Also

genfrail, fitfrail, simfrail

generation of survival data

Description

Generate clustered survival data from a shared frailty model, with hazard function given by

\[ S(t) = \exp\left[ -\Lambda_0(t) \omega_i \exp(\beta Z_{ij}) \right] \]

where \( \Lambda_0 \) is the cumulative baseline hazard, \( \omega_i \) is the frailty value of cluster \( i \), \( \beta \) is the regression coefficient vector, and \( Z_{ij} \) is the covariate vector for individual \( i \) in cluster \( j \).

The baseline hazard can be specified by the inverse cumulative baseline hazard, cumulative baseline hazard, or simply the baseline hazard. Frailty values can be sampled from gamma, power variance function (PVF), log-normal, inverse Gaussian, and positive stable distributions.

Usage

```r
genfrail(N = 300, K = 2, K.param = c(2, 0), beta = c(log(2)),
frailty = "gamma", theta = c(2),
covar.distr = "normal", covar.param = c(0, 1), covar.matrix = NULL,
censor.distr = "normal", censor.param = c(130, 15),
censor.rate = NULL, censor.time = NULL,
lambda_0 = NULL, Lambda_0 = NULL, Lambda_0.inv = NULL,
round.base = NULL, control, ...)
```

Arguments

- **N**: integer; number of clusters
- **K**: integer, string, or vector; If an integer, the number of members in each cluster. If a string, the name of the distribution to sample the cluster sizes from. This can be one of: "poisson", "pareto", or "uniform". The K.param argument specifies the distribution parameters. If a vector, must be of length N and contains the integer size of each cluster.
- **K.param**: vector of the cluster size distribution parameters if K is a string. If "poisson", the vector should contain the rate and truncated value (see rtpois). If "pareto", the exponent, lower, and upper bounds (see rtzeta). If "uniform", the lower (noninclusive) and upper (inclusive) bounds.
- **beta**: vector of regression coefficients.
- **frailty**: string name of the frailty distribution. Can be one of: "gamma", "pvf", "lognormal", "invgauss", "posstab", or "none". See dgamma_r, dpvf_r, dlognormal_r, dinvgauss_r, posstab_r for the respective density functions. (Also see the *_c for C implementations of the respective density functions.)
theta vector the frailty distribution parameters
covar.distr string distribution to sample covariates from. Can be one of: "normal", "uniform", "zero"
covar.param vector covariate distribution parameters.
covar.matrix matrix with dimensions \( c(N K, \text{length} (\text{beta})) \) that contains the desired covariates. If not NULL, this overrides covar.distr and covar.param.
censor.distr string censoring distribution to use. Followup times are sampled from the censoring distribution to simulate non-informative right censorship. The censoring distribution can be one of: "normal", "lognormal", "uniform", "none".
censor.param vector of censoring distribution parameters. For normal and lognormal censorship, this should be \( c(\mu, \sigma) \) where \( \mu \) is the mean and \( \sigma \) is the standard deviation (Note: this is still the mean and standard deviation for lognormal). For uniform censorship, the vector \( c(\text{lower}, \text{upper}) \) should specify the lower and upper bounds.
censor.rate numeric value between 0 and 1 to specify the empirical censoring rate. The mean specified in the censor.param parameter is adjusted to achieve a desired censoring rate if censor.rate is given. Note that the standard deviation (the second parameter in censor.param) must still be specified so that the problem is identifiable. For uniform censorship, the interval given by \( c(\text{lower}, \text{upper}) \) is adjusted to achieve the desired censorship, while keeping the variance fixed (i.e., upper - lower does not change).
censor.time vector of right-censorship times. This must have length \( N*K \) and specifies the right-censoring times of each observation. Note that this overrides all other censor.* params and cannot be used with variable cluster sizes.
lambda_0 function baseline hazard. Only one of lambda_0, Lambda_0, and Lambda_0_inv need to be specified. Passing the baseline hazard (lambda_0) is the most computationally expensive since this requires numerical integration inside a root-finding algorithm.
Lambda_0 function cumulative baseline hazard. This overrides lambda_0.
Lambda_0_inv function inverse cumulative baseline hazard. This overrides both lambda_0 and Lambda_0.
round.base numeric if specified, round the followup times to the nearest round.base
ccontrol control parameters in the form of a genfrail.control object
... additional arguments will be passed to genfrail.control

Value
A data.frame with row-observations is returned.

family the cluster
rep the member within each cluster
time observed followup time
status failure indicator
Z1... covariates, where there are length(beta) Z columns
Author(s)
John V. Monaco, Malka Gorine, and Li Hsu.

See Also
fitfrail

Examples

# Generate the same dataset 3 different ways

# Using the baseline hazard (least efficient)
set.seed(1234)
dat.1 <- genfrail(N = 300, K = 2,
   beta = c(log(2),log(3)),
   frailty = "gamma", theta = 2,
   lambda_0=function(t, tau=4.6, C=0.01) (tau*(C*t)*tau)/t)

# Using the cumulative baseline hazard
set.seed(1234)
dat.2 <- genfrail(N = 300, K = 2,
   beta = c(log(2),log(3)),
   frailty = "gamma", theta = 2,
   Lambda_0 = function(t, tau=4.6, C=0.01) (C*t)*tau)

# Using the inverse cumulative baseline hazard (most efficient)
set.seed(1234)
dat.3 <- genfrail(N = 300, K = 2,
   beta = c(log(2),log(3)),
   frailty = "gamma", theta = 2,
   Lambda_0_inv=function(t, tau=4.6, C=0.01) (t^(1/tau))/C)

# Generate data with PVF frailty, truncated Poisson cluster sizes, normal
# covariates, and 0.35 censorship from a lognormal distribution
set.seed(1234)
dat.4 <- genfrail(N = 100, K = "poisson", K.param=c(5, 1),
   beta = c(log(2),log(3)),
   frailty = "pvf", theta = 0.3,
   covar.distr = "lognormal",
   censor.rate = 0.35) # Use the default baseline hazard

# Cluster sizes have size >= 2, summarized by summary(dat.4)

# An oscillating baseline hazard
set.seed(1234)
dat.5 <- genfrail(lambda_0=function(t, tau=4.6, C=0.01, A=2, f=0.1)
   A*sin(f*pi*t) * (tau*(C*t)*tau)/t)

# Uniform censorship with 0.25 censoring rate
set.seed(1234)
dat.6 <- genfrail(N = 300, K = 2,
   beta = c(log(2),log(3)),
   frailty = "gamma", theta = 2,
   censor.distr = "uniform",
   censor.rate = 0.25)
**Description**

This function creates a list of control parameters needed by `genfrail`.

**Usage**

```r
genfrail.control(censor.reltol = 1e-4,
censor.subdivisions = 1000L,
crowther.reltol = 1e-4,
crowther.subdivisions = 1000L)
```

**Arguments**

- `censor.reltol`: numeric relative tolerance for convergence of the censorship numerical integration. Default is 0.001.
- `censor.subdivisions`: integer, maximum number of censorship numerical integration subdivisions. Default is 1000.
- `crowther.reltol`: numeric relative tolerance for convergence of the numerical integration in Crowther's formula. Default is 0.001.
- `crowther.subdivisions`: integer, maximum number of numerical integration subdivisions in Crowther's formula. Default is 1000.

**Value**

A list of control parameters.

**Author(s)**

John V. Monaco, Malka Gorfine, Li Hsu

**See Also**

`genfrail`
Hard drive failure dataset

Description
This dataset contains the observed follow-up times and SMART statistics of 52k unique hard drives. Daily snapshots of a large backup storage provider over 2 years were made publicly available. On each day, the Self-Monitoring, Analysis, and Reporting Technology (SMART) statistics of operational drives are recorded. When a hard drive is no longer operational, it is marked as a failure and removed from the subsequent daily snapshots. New hard drives are also continuously added to the population. In total, there are over 52k unique hard drives over approximately 2 years and 2885 (5.5%) failures.

Usage
data("hdfail")

Format
A data frame with 52422 observations on the following 8 variables.

- serial    unique serial number of the hard drive
- model     hard drive model
- time      the observed follow up time
- status    failure indicator
- temp      temperature in Celsius
- rsc       binary covariate, where 1 indicates sectors that encountered read, write, or verification errors
- rer       binary covariate, where 1 indicates a non-zero rate of errors that occur in hardware when reading from data from disk.
- psc       binary covariate, where 1 indicates there were sectors waiting to be remapped due to an unrecoverable error.

Source

Examples
```r
## Not run:
data(hdfail)

# Select only Western Digital hard drives
dat <- subset(hdfail, grepl("WDC", model))

fit.hd <- fitfrail(Surv(time, status) ~ temp + rer + rsc
                   + psc + cluster(model),
```
Dat frailty="gamma", fitmethod="score")

fit.hd

## End(Not run)

---

**plot.fitfrail** *Plot method for fitfrail objects*

**Description**

Plot the cumulative baseline hazard estimates or the parameter trace from model estimation.

**Usage**

```r
## S3 method for class 'fitfrail'
plot(x, type = c("cumhaz", "trace"), ...)
```

**Arguments**

- `x` a fitfrail object
- `type` string, the type of plot. Can be either "cumhaz" to plot the mean estimated cumulative hazard or "trace" to plot the parameter and log-likelihood trace.
- `...` extra arguments include:
  - `CI` for type="cumhaz", numeric confidence interval between 0 and 1. If CI=0, no confidence interval is displayed. Otherwise, the bootstrapped confidence interval is calculated and displayed.
  - `end` for type="cumhaz", numeric x-axis limit (plot up to time end)
  - `show.loglik` for type="trace", logical whether to show the log-likelihood trace.

**Value**

The plot object.

**Author(s)**

John. V Monaco, Malka Gorfine, Li Hsu

**See Also**

fitfrail
Examples

```r
## Not run:
data(drs)
fit.drs <- fitfrail(Surv(time, status) ~ treated + cluster(subject_id),
                    drs, frailty="gamma")

# Plot the parameter and log-likelihood trace
plot(fit.drs, type="trace")

# This may take a while to run.
# Use parameter B to specify the number of repetitions in the weighted bootstrap
plot(fit.drs, type="cumhaz", CI=0.95)

## End(Not run)
```

---

**plot.simfrail**

*Plot method for simfrail objects*

**Description**

Plot the estimated parameter residuals or the mean estimated cumulative baseline hazard.

**Usage**

```r
## S3 method for class 'simfrail'
plot(x, type = c("residuals", "cumhaz"), ...)
```

**Arguments**

- `x`  
  a `fitfrail` object

- `type`  
  string, the type of plot. Can be either "residuals" or "cumhaz". If `type="residuals"`, a boxplot of the estimated parameter residuals is created. If `type="cumhaz"`, the mean estimated and true cumulative baseline hazard are plotted.

- `...`  
  extra arguments include:
  - `CI` for `type="cumhaz"`, the confidence interval for the empirical cumulative baseline hazard.
  - `n.Lambda` for `type="residuals"`, the number of time points to show the cumulative baseline hazard residuals for.

**Value**

The plot object.

**Author(s)**

John. V Monaco, Malka Gorfine, Li Hsu
simcoxph

Simulate survival data and fit models

Description

Generates simulated clustered survival data by repeatedly generating data, using a shared frailty model, and fitting the models. Respective arguments are passed to genfrail and coxph, and the resulting parameter estimates are aggregated and summarized.

This function is similar to simfrail, except models are fitted using the coxph.

Usage

simcoxph(reps, genfrail.args, coxph.args, Lambda.times, cores = 0)

Arguments

reps number of times to repeat the simulation
genfrail.args list of arguments to pass to genfrail
coxph.args list of arguments to pass to coxph
Lambda.times vector of time points to obtain baseline hazard estimates at
cores integer; if > 0, the number of cores to use; if < 0, the number of cores not to use; if 0, use all available cores
Value

A simcoxph object that is essentially a data.frame of the resulting parameter estimates. Each row is a single run, and columns are as follows.

seed  the seed used for the run
runtime  the time it took to fit the model
N  number of clusters
mean.K  average cluster size
cens  empirical censorship
beta  true regression coefficients
hat.beta  estimated regression coefficients
se.beta  standard error of each regression coefficient
theta  true frailty distribution parameters
hat.theta  estimated frailty distribution parameters
se.theta  standard error of each frailty distribution parameter (NA since coxph does not currently provide this.)
Lambda  true cumulative baseline hazard at each Lambda.times point
hat.Lambda  estimated cumulative baseline hazard at each Lambda.times point
se.Lambda  standard error at each Lambda.times point (NA since coxph does not currently provide this)

Author(s)

John. V Monaco, Malka Gorfine, Li Hsu

See Also

coxph, genfrail, simfrail

Examples

## Not run:
sim <- simcoxph(reps=100,
genfrail.args=alist(
  N=50, K=2,
  beta=c(log(2),log(3)),
  frailty="gamma", theta=2,
  Lambda_0 = function(t, tau=4.6, C=0.01) (C+t)*tau),
coxph.args=alist(
  formula=Surv(time, status) ~ Z1 + Z2 + cluster(family),
  frailty="gamma"),
Lambda.times=1:120, cores = 0)

# Summarize the results
summary(sim)
# Plots and Residuals

```r
# Plot the residuals
plot(sim, "residuals")

## End(Not run)
```

---

### simfrail

**Simulate survival data and fit models**

#### Description

Generates simulated clustered survival data by repeatedly generating data, using a shared frailty model, and fitting the models. Respective arguments are passed to genfrail and fitfrail, and the resulting parameter estimates are aggregated and summarized.

#### Usage

```
simfrail(reps, genfrail.args, fitfrail.args, Lambda.times,
         vcov.args = list(), cores = 0, skip.SE = FALSE)
```

#### Arguments

- `reps`: number of times to repeat the simulation
- `genfrail.args`: list of arguments to pass to `genfrail`
- `fitfrail.args`: list of arguments to pass to `fitfrail`
- `Lambda.times`: vector of time points to obtain baseline hazard estimates at
- `vcov.args`: list of arguments to pass to `vcov.fitfrail` for variance estimates. This is mainly used to specify whether bootstrap or estimated variances should be obtained.
- `cores`: integer; if > 0, the number of cores to use; if < 0, the number of cores not to use; if 0, use all available cores
- `skip.SE`: logical value, whether to skip the standard error estimates (saves time)

#### Value

A `simfrail` object that is essentially a `data.frame` of the resulting parameter estimates. Each row is a single run, and columns are as follows.

- `seed`: the seed used for the run
- `runtime`: the time it took to fit the model
- `N`: number of clusters
- `mean.K`: average cluster size
- `cens`: empirical censorship
- `beta`: true regression coefficients
- `hat.beta`: estimated regression coefficients
- `se.beta`: standard error of each regression coefficient
theta  true frailty distribution parameters
hat.theta  estimated frailty distribution parameters
se.theta  standard error of each frailty distribution parameter
Lambda  true cumulative baseline hazard at each Lambda.times point
hat.Lambda  estimated cumulative baseline hazard at each Lambda.times point
se.Lambda  standard error at each Lambda.times point

Author(s)
John. V Monaco, Malka Gorfine, Li Hsu

See Also
genfrail, fitfrail

Examples

```r
## Not run:
sim <- simfrail(reps=100,
genfrail.args=list(
N=50, K=2,
beta=c(log(2),log(3)),
frailty="gamma", theta=2,
Lambda_0 = function(t, tau=4.6, C=0.01) (C+t)^tau),
fitfrail.args=list(
 formula=Surv(time, status) ~ Z1 + Z2 + cluster(family),
 frailty="gamma"),
Lambda.times=1:120, cores = 0)

# Summarize the results
summary(sim)

# Plot the residuals
plot(sim, "residuals")

## End(Not run)
```

---

**summary.fitfrail**

Summary of the survival curve

**Description**

Returns a data.frame summarizing the survival curve of the fitted model. If specified, this function uses a weighted bootstrap procedure to calculate SE of the survival curve estimates. Subsequent calls with the same arguments will use the cached SE and avoid performing the weighted bootstrap again.
**Usage**

```r
## S3 method for class 'fitfrail'
summary(object, type = "survival", Lambda.times = NULL,
         censored = FALSE, se = FALSE, CI = 0.95, ...)
```

**Arguments**

- `object`: a `fitfrail` object
- `type`: string indicating the type of summary: either "survival" for a summary of the survival curve, or "cumhaz" for a summary of the cumulative baseline hazard.
- `Lambda.times`: vector of times where the curve should be evaluated. The resulting `data.frame` will have 1 row for each time. If NULL and `censored=True`, all observed times are used by default. If NULL and `censored=False`, only the failure times are including in the results.
- `censored`: logical value, whether the survival curve should contain the censored times. Ignored if `Lambda.times` is not NULL.
- `se`: logical value, whether the survival SE should be included with the results. If `se=True`, a weighted bootstrap procedure is used to determine estimated survival SE.
- `CI`: numeric, the confidence interval to evaluate upper and lower limits for the survival estimate at each time point
- `...`: extra arguments will be passed to `vcov.fitfrail`

**Value**

A `data.frame` summarizing the survival curve with the following columns.

- `time`: the time points
- `surv/cumhaz`: survival/cumulative hazard estimate at time t+
- `n.risk`: number of subjects at risk at time t-
- `n.event`: the number of failures that occured from the last time point to time t+
- `std.err`: the SE of the survival estimate
- `lower.ci`: lower bound on the specified confidence interval
- `upper.ci`: upper bound on the specified confidence interval

**Note**

Similar to `summary.survfit` function in the `survival` package.

**See Also**

`fitfrail, vcov.fitfrail`
Examples

```r
## Not run:
dat <- genfrail(N=200, K=2, beta=c(log(2), log(3)),
               frailty="gamma", theta=2,
               censor.rate=0.35,
               Lambda_0=function(t, tau=4.6, C=0.01) (C*t)^tau)

fit <- fitfrail(Surv(time, status) ~ Z1 + Z2 + cluster(family),
               dat, frailty="gamma")

surv <- summary(fitfrail, B=50, se=TRUE, CI=0.95)
head(surv)

## End(Not run)
```

---

**vcov.fitfrail**  
Compute variance/covariance matrix for fitfrail model

---

**Description**

Compute the variance/covariance matrix for fitfrail estimated parameters. This can be performed by an asymptotically-normal and consistent variance estimator or a weighted bootstrap. The resulting covariance matrix is cached in the fitted object and later retrieved if the same arguments to `vcov.fitfrail` are supplied.

**Usage**

```r
## S3 method for class 'fitfrail'
vcov(object, boot=FALSE, B=100, Lambda.times=NULL, cores=0, ...)
```

**Arguments**

- `object`  
a fitfrail object
- `boot`  
logical value, whether to use a weighted bootstrap. If `boot == FALSE`, a consistent estimator is used and the cumulative baseline hazard variance will not be estimated.
- `B`  
number of repetitions in the weighted bootstrap.
- `Lambda.times`  
time points where the variance/covariance should be evaluated. If `Lambda.times == NULL`, then the points where the cumulative baseline hazard increases (where failures occur) are used.
- `cores`  
number of cores to use when computing the covariance matrix in parallel
- `...`  
extra arguments are not used

**Value**

variance/covariance matrix for the fitfrail model parameters
## Examples

```r
## Not run
dat <- genfrail(N=200, K=2, beta=c(log(2), log(3)),
                frailty="gamma", theta=2,
                censor.rate=0.35,
                Lambda_0=function(t, tau=4.6, C=0.01) (C*t)^tau)

fit <- fitfrail(Surv(time, status) ~ Z1 + Z2 + cluster(family),
                dat, frailty="gamma")

# boot=TRUE will give the weighted bootstrap variance estimates
COV <- vcov(fit, boot=FALSE)
COV

## End(Not run)
```
Index

*Topic datasets
  drs, 2
  hdfail, 12
*Topic shared frailty
  genfrail, 8
*Topic survival data
  genfrail, 8
*Topic survival simulation
  simcoxph, 15
  simfrail, 17

coxph, 4, 15, 16

drs, 2

fitfrail, 3, 7, 8, 10, 13, 18, 19, 21
fitfrail.control, 3, 5
frailtySurv, 7
frailtySurv-package (frailtySurv), 7

genfrail, 4, 7, 8, 8, 11, 16, 18
genfrail.control, 9, 11

hdfail, 12

plot.fitfrail, 13
plot.simfrail, 14

simcoxph, 15
simfrail, 4, 7, 8, 15, 16, 17
summary.fitfrail, 18
Surv, 3
survfit, 4

vcov.fitfrail, 4, 19, 20