Package 'frailtyEM'

October 30, 2026

Type Package

Title Fitting Frailty Models with the EM Algorithm

Version 1.0.2

URL https://github.com/tbalan/frailtyEM

BugReports https://github.com/tbalan/frailtyEM/issues

Description Contains functions for fitting shared frailty models with a semi-parametric baseline hazard with the Expectation-Maximization algorithm. Supported data formats include clustered failures with left truncation and recurrent events in gap-time or Andersen-Gill format. Several frailty distributions, such as the the gamma, positive stable and the Power Variance Family are supported.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.3.0), survival

Imports Rcpp (>= 0.12.8), magrittr, msm, ggplot2, expint, tibble, Matrix, numDeriv

LinkingTo Rcpp

Suggests dplyr, plotly, gridExtra, egg

Collate 'RcppExports.R' 'autoplot.emfrail.R' 'ca_test_cph.R' 'ca_test_fit.R' 'em_fit.R' 'emfrail_aux.R' 'emfrail.R' 'emfrail_arguments.R' 'emfrail_methods.R' 'fast_Estep.R' 'frailtyEM.R' 'plot.emfrail.R' 'predict.emfrail.R' 'print.emfrail.R' 'print.emfrail_summary.R' 'summary.emfrail.R'

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Theodor Adrian Balan [aut, cre], Hein Putter [aut]

Maintainer Theodor Adrian Balan <hello@tbalan.com>

Repository CRAN

Date/Publication 2026-10-30 08:10:02 UTC

Contents

autoplot 2
autoplot.emfrail 2
ca_test 4
emfrail 5
emfrail_control 10
emfrail_dist 12
emfrail_pll 13
logLik.emfrail 15
plot.emfrail 16
predict.emfrail 18
residuals.emfrail 20
summary.emfrail 21

Index **24**

`autoplot` *Generic autoplot function*

Description

The following is imported and then re-exported to avoid conflicts with `ggplot2`

`autoplot.emfrail` *Plots for emfrail objects using* `ggplot2`

Description

Plots for emfrail objects using `ggplot2`

Usage

```
## S3 method for class 'emfrail'
autoplot(
  object,
  type = c("hist", "hr", "pred", "frail"),
  newdata = NULL,
  lp = NULL,
  strata = NULL,
  quantity = "cumhaz",
  type_pred = c("conditional", "marginal"),
  conf_int = "adjusted",
  conf_level = 0.95,
  individual = FALSE,
  ...
)
```

Arguments

object	emfrail object, typically result of emfrail()
type	One (or more) of hist for a histogram of the estimated frailty values, hr for a plot of the conditional and marginal hazard ratio between two cases, pred for the predicted conditional and marginal cumulative hazard or survival for one case, frail for a caterpillar plot of the ordered frailty estimates with confidence intervals, where available.
newdata	A data.frame with values of the covariates. For type == "hr" the hazard ratio between the first two rows of newdata is calculated. For type == "pred" the prediction for the first row of newdata is calculated.
lp	A numeric vector of values of the linear predictor, each corresponding to a case. For type == "hr" the hazard ratio between the first two values of lp is calculated. For type == "pred" the prediction for the first value of lp is calculated.
strata	The name of the strata (if applicable) for which the prediction should be made.
quantity	One of c("cumhaz", "survival") for type == "pred"; see quantity in predict.emfrail
type_pred	One of c("conditional", "marginal") for type == "pred"; see type in predict.emfrail
conf_int	One of c("regular", "adjusted") for type == "pred"; see conf_int in predict.emfrail
conf_level	The width of the confidence interval for type == "pred"; see conf_level in predict.emfrail
individual	Logical, for type == "pred" to be used for drawing a curve when the rows of newdata refer to the same individual; see individual in predict.emfrail
...	Further arguments to be passed on to 'ggplot' (ignored)

Value

A list of ggplot2 objects corresponding to the required plots, or one ggplot2 if only one plot is selected

For the catterpillar plot, in the case of the gamma frailty model, the vertical lines represent the 0.025 and 0.975 quantiles of the posterior gamma distribution. For other distributions, this quantity is not easy to calculate (at least not in closed form) and only the frailty estimates are shown.

Note

It's normal for autoplot to give a warning of the type Warning: Ignoring unknown aesthetics: id . This is because, in ggplot2 terms, the id aesthetic is not recognized. This is correct, and for any practical purpose this will not make a difference (you can safely ignore the warnings). However, this makes it easier to create an interactive plot out of the resulting object.

See Also

predict.emfrail, summary.emfrail, plot.emfrail.

Examples

```
mod_rec <- emfrail(Surv(start, stop, status) ~ treatment + number + cluster(id), bladder1,
control = emfrail_control(ca_test = FALSE, lik_ci = FALSE))

# Histogram of the estimated frailties
autoplot(mod_rec, type = "hist")

# Ordered estimated frailties (with confidence intervals, for gamma distribution)
autoplot(mod_rec, type = "frail")

# hazard ratio between placebo and pyridoxine
newdata1 <- data.frame(treatment = c("placebo", "pyridoxine"),
                       number = c(1, 3))

autoplot(mod_rec, type = "hr", newdata = newdata1)

# predicted cumulative hazard for placebo, and number = 1
autoplot(mod_rec, type = "pred", newdata = newdata1[1,])

# predicted survival for placebo, and number = 1
autoplot(mod_rec, type = "pred", quantity = "survival", newdata = newdata1[1,])

# predicted survival for an individual that switches from
# placebo to pyridoxine at time = 15
## Not run:
newdata2 <- data.frame(treatment = c("placebo", "pyridoxine"),
                       number = c(1, 3),
                       tstart = c(0, 15),
                       tstop = c(15, Inf))

autoplot(mod_rec, type = "pred", quantity = "survival", newdata = newdata2, individual = TRUE)

## End(Not run)
```

---

ca_test *Commenges-Andersen test for heterogeneity*

Description

Commenges-Andersen test for heterogeneity

Usage

```
ca_test(object, id = NULL)
```

Arguments

object	A coxph object with a cluster() statement in the right-hand side of the formula.
id	Optionally, a vector determining the grouping to be tested. See details.

Details

The Cox model with a `+cluster()` statement has the same point estimates as the one without that statmenet. The only difference is in the adjusted standard errors. In some cases, a model with `+cluster()` statments can't be fitted. For example, when there are no covariates. In that case, a vector may be passed on in the `cluster` argument.

Value

A named vector containing the test statistic, variance, and p-value

References

Commenges, D. and Andersen, P.K., 1995. Score test of homogeneity for survival data. Lifetime Data Analysis, 1(2), pp.145-156.

Examples

```
mcox1 <- coxph(Surv(time, status) ~ rx + sex + cluster(litter),
rats, model = TRUE, x = TRUE)
ca_test(mcox1)

mcox2 <- coxph(Surv(time, status) ~ 1, rats, x = TRUE)
ca_test(mcox2, rats$litter)
```

`emfrail` *Fitting semi-parametric shared frailty models with the EM algorithm*

Description

Fitting semi-parametric shared frailty models with the EM algorithm

Usage

```
emfrail(
  formula,
  data,
  distribution = emfrail_dist(),
  control = emfrail_control(),
  model = FALSE,
  model.matrix = FALSE,
  ...
)
```

Arguments

formula	A formula that contains on the left hand side an object of the type Surv and on the right hand side a +cluster(id) statement. Two special statments may also be used: +strata() for specifying a grouping column that will represent different strata and +terminal()
data	A data.frame in which the formula argument can be evaluated
distribution	An object as created by emfrail_dist
control	An object as created by emfrail_control
model	Logical. Should the model frame be returned?
model.matrix	Logical. Should the model matrix be returned?
...	Other arguments, currently used to warn about deprecated argument names

Details

The emfrail function fits shared frailty models for processes which have intensity

$$\lambda(t) = z\lambda_0(t)\exp(\beta'\mathbf{x})$$

with a non-parametric (Breslow) baseline intensity $\lambda_0(t)$. The outcome (left hand side of the formula) must be a Surv object.

If the object is Surv(tstop, status) then the usual failure time data is represented. Gap-times between recurrent events are represented in the same way. If the left hand side of the formula is created as Surv(tstart, tstop, status), this may represent a number of things: (a) recurrent events episodes in calendar time where a recurrent event episode starts at tstart and ends at tstop (b) failure time data with time-dependent covariates where tstop is the time of a change in covariates or censoring (status = 0) or an event time (status = 1) or (c) clustered failure time with left truncation, where tstart is the individual's left truncation time. Unlike regular Cox models, a major distinction is that in case (c) the distribution of the frailty must be considered conditional on survival up to the left truncation time.

The +cluster() statement specified the column that determines the grouping (the observations that share the same frailty). The +strata() statement specifies a column that determines different strata, for which different baseline hazards are calculated. The +terminal specifies a column that contains an indicator for dependent censoring, and then performs a score test

The distribution argument must be generated by a call to emfrail_dist. This determines the frailty distribution, which may be one of gamma, positive stable or PVF (power-variance-function), and the starting value for the maximum likelihood estimation. The PVF family also includes a tuning parameter that differentiates between inverse Gaussian and compound Poisson distributions. Note that, with univariate data (at most one event per individual, no clusters), only distributions with finite expectation are identifiable. This means that the positive stable distribution should have a maximum likelihood on the edge of the parameter space ($theta = +\inf$, corresponding to a Cox model for independent observations).

The control argument must be generated by a call to emfrail_control. Several parameters may be adjusted that control the precision of the convergenge criteria or supress the calculation of different quantities.

Value

An object of class `emfrail` that contains the following fields:

`coefficients`	A named vector of the estimated regression coefficients
`hazard`	The breslow estimate of the baseline hazard at each event time point, in chronological order
`var`	The variance-covariance matrix corresponding to the coefficients and hazard, assuming θ constant
`var_adj`	The variance-covariance matrx corresponding to the coefficients and hazard, adjusted for the estimation of theta
`logtheta`	The logarithm of the point estimate of θ. For the gamma and PVF family of distributions, this is the inverse of the estimated frailty variance.
`var_logtheta`	The variance of the estimated logarithm of θ
`ci_logtheta`	The likelihood-based 95% confidence interval for the logarithm of θ
`frail`	The posterior (empirical Bayes) estimates of the frailty for each cluster
`residuals`	A list with two elements, cluster which is a vector that the sum of the cumulative hazards from each cluster for a frailty value of 1, and individual, which is a vector that contains the cumulative hazard corresponding to each row of the data, multiplied by the corresponding frailty estimate
`tev`	The time points of the events in the data set, this is the same length as hazard
`nevents_id`	The number of events for each cluster
`loglik`	A vector of length two with the log-likelihood of the starting Cox model and the maximized log-likelihood
`ca_test`	The results of the Commenges-Andersen test for heterogeneity
`cens_test`	The results of the test for dependence between a recurrent event and a terminal event, if the `+terminal()` statement is specified and the frailty distribution is gamma
`zph`	The result of `cox.zph` called on a model with the estimated log-frailties as offset
`formula`, `distribution`, `control`	The original arguments
`nobs`, `fitted`	Number of observations and fitted values (i.e. $z\exp(\beta^T x)$)
`mf`	The `model.frame`, if `model = TRUE`
`mm`	The `model.matrix`, if `model.matrix = TRUE`

Note

Several options in the `control` arguemnt shorten the running time for `emfrail` significantly. These are disabling the adjustemnt of the standard errors (`se_adj = FALSE`), disabling the likelihood-based confidence intervals (`lik_ci = FALSE`) or disabling the score test for heterogeneity (`ca_test = FALSE`).

The algorithm is detailed in the package vignette. For the gamma frailty, the results should be identical with those from `coxph` with `ties = "breslow"`.

Author(s)

Theodor Balan <hello@tbalan.com>

References

Balan TA, Putter H (2019) "frailtyEM: An R Package for Estimating Semiparametric Shared Frailty Models", *Journal of Statistical Software* **90**(7) 1-29. doi:10.18637/jss.v090.i07

See Also

`plot.emfrail` and `autoplot.emfrail` for plot functions directly available, `emfrail_pll` for calculating $\widehat{L}(\theta)$ at specific values of θ, `summary.emfrail` for transforming the `emfrail` object into a more human-readable format and for visualizing the frailty (empirical Bayes) estimates, `predict.emfrail` for calculating and visalizing conditional and marginal survival and cumulative hazard curves. `residuals.emfrail` for extracting martingale residuals and `logLik.emfrail` for extracting the log-likelihood of the fitted model.

Examples

```
m_gamma <- emfrail(formula = Surv(time, status) ~  rx + sex + cluster(litter),
                   data =  rats)

# Inverse Gaussian distribution
m_ig <- emfrail(formula = Surv(time, status) ~  rx + sex + cluster(litter),
                data =  rats,
                distribution = emfrail_dist(dist = "pvf"))

# for the PVF distribution with m = 0.75
m_pvf <- emfrail(formula = Surv(time, status) ~  rx + sex + cluster(litter),
                 data =  rats,
                 distribution = emfrail_dist(dist = "pvf", pvfm = 0.75))

# for the positive stable distribution
m_ps <- emfrail(formula = Surv(time, status) ~  rx + sex + cluster(litter),
                data =  rats,
                distribution = emfrail_dist(dist = "stable"))
## Not run:
# Compare marginal log-likelihoods
models <- list(m_gamma, m_ig, m_pvf, m_ps)

models
logliks <- lapply(models, logLik)

names(logliks) <- lapply(models,
                         function(x) with(x$distribution,
                                          ifelse(dist == "pvf",
                                                 paste(dist, "/", pvfm),
                                                 dist))
)

logliks
```

```
## End(Not run)

# Stratified analysis
## Not run:
  m_strat <- emfrail(formula = Surv(time, status) ~  rx + strata(sex) + cluster(litter),
                     data =  rats)

## End(Not run)


# Test for conditional proportional hazards (log-frailty as offset)
## Not run:
m_gamma <- emfrail(formula = Surv(time, status) ~  rx + sex + cluster(litter),
  data =  rats, control = emfrail_control(zph = TRUE))
par(mfrow = c(1,2))
plot(m_gamma$zph)

## End(Not run)

# Draw the profile log-likelihood
## Not run:
  fr_var <- seq(from = 0.01, to = 1.4, length.out = 20)

  # For gamma the variance is 1/theta (see parametrizations)
  pll_gamma <- emfrail_pll(formula = Surv(time, status) ~  rx + sex + cluster(litter),
                           data =  rats,
                           values = 1/fr_var )
  plot(fr_var, pll_gamma,
       type = "l",
       xlab = "Frailty variance",
       ylab = "Profile log-likelihood")


  # Recurrent events
  mod_rec <- emfrail(Surv(start, stop, status) ~ treatment + cluster(id), bladder1)
  # The warnings appear from the Surv object, they also appear in coxph.

  plot(mod_rec, type = "hist")

## End(Not run)

# Left truncation
## Not run:
  # We simulate some data with truncation times
  set.seed(2018)
  nclus <- 300
  nind <- 5
  x <- sample(c(0,1), nind * nclus, TRUE)
  u <- rep(rgamma(nclus,1,1), each = 3)

  stime <- rexp(nind * nclus, rate = u * exp(0.5 * x))
```

```
status <- ifelse(stime > 5, 0, 1)
stime[status == 0] <- 5

# truncate uniform between 0 and 2
ltime <- runif(nind * nclus, min = 0, max = 2)

d <- data.frame(id = rep(1:nclus, each = nind),
                x = x,
                stime = stime,
                u = u,
                ltime = ltime,
                status = status)
d_left <- d[d$stime > d$ltime,]

mod <- emfrail(Surv(stime, status)~ x + cluster(id), d)
# This model ignores the left truncation, 0.378 frailty variance:
mod_1 <- emfrail(Surv(stime, status)~ x + cluster(id), d_left)

# This model takes left truncation into account,
# but it considers the distribution of the frailty unconditional on the truncation
mod_2 <- emfrail(Surv(ltime, stime, status)~ x + cluster(id), d_left)

# This is identical with:
mod_cox <- coxph(Surv(ltime, stime, status)~ x + frailty(id), data = d_left)


# The correct thing is to consider the distribution of the frailty given the truncation
mod_3 <- emfrail(Surv(ltime, stime, status)~ x + cluster(id), d_left,
                 distribution = emfrail_dist(left_truncation = TRUE))

summary(mod_1)
summary(mod_2)
summary(mod_3)

## End(Not run)
```

`emfrail_control` *Control parameters for emfrail*

Description

Control parameters for emfrail

Usage

```
emfrail_control(
  opt_fit = TRUE,
  se = TRUE,
  se_adj = TRUE,
  ca_test = TRUE,
```

```
  lik_ci = TRUE,
  lik_interval = exp(c(-3, 20)),
  lik_interval_stable = exp(c(0, 20)),
  nlm_control = list(stepmax = 1),
  zph = FALSE,
  zph_transform = "km",
 em_control = list(eps = 1e-04, maxit = Inf, fast_fit = TRUE, verbose = FALSE, upper_tol
    = exp(10), lik_tol = 1)
)
```

Arguments

opt_fit	Logical. Whether the outer optimization should be carried out. If FALSE, then the frailty parameter is treated as fixed and the emfrail function returns only log-likelihood. See details.
se	Logical. Whether to calculate the variance / covariance matrix.
se_adj	Logical. Whether to calculate the adjusted variance / covariance matrix (needs se == TRUE)
ca_test	Logical. Should the Commenges-Andersen test be calculated?
lik_ci	Logical. Should likelihood-based confidence interval be calculated for the frailty parameter?
lik_interval	The edges, on the scale of θ, of the parameter space in which to search for likelihood-based confidence interval
lik_interval_stable	(for dist = "stable") The edges, on the scale of θ, of the parameter space in which to search for likelihood-based confidence interval
nlm_control	A list of named arguments to be sent to nlm for the outer optimization.
zph	Logical. Should the cox.zph test be performed at the maximum likelihood estimate?
zph_transform	One of "km", "rank", "identity" or a function of one argument to be pased on to cox.zph.
em_control	A list of parameters for the inner optimization. See details.

Details

The nlm_control argument should not overalp with hessian, f or p.

The em_control argument should be a list with the following items:

eps A criterion for convergence of the EM algorithm (difference between two consecutive values of the log-likelihood)

maxit The maximum number of iterations between the E step and the M step

fast_fit Logical, whether the closed form formulas should be used for the E step when available

verbose Logical, whether details of the optimization should be printed

upper_tol An upper bound for θ; after this treshold, the algorithm returns the limiting log-likelihood of the no-frailty model. That is because the no-frailty scenario corresponds to a $\theta = \infty$, which could lead to some numerical issues

`lik_tol` For values higher than this, the algorithm returns a warning when the log-likelihood decreases between EM steps. Technically, this should not happen, but if the parameter θ is somewhere really far from the maximum, numerical problems might lead in very small likelihood decreases.

The `fast_fit` option make a difference when the distribution is gamma (with or without left truncation) or inverse Gaussian, i.e. pvf with m = -1/2 (without left truncation). For all the other scenarios, the fast_fit option will automatically be changed to FALSE. When the number of events in a cluster / individual is not very small, the cases for which fast fitting is available will show an improvement in performance.

The starting value of the outer optimization may be set in the `distribution` argument.

Value

An object of the type `emfrail_control`.

See Also

`emfrail`, `emfrail_dist`, `emfrail_pll`

Examples

```
emfrail_control()
emfrail_control(em_control = list(eps = 1e-7))
```

`emfrail_dist` *Distribution parameters for emfrail*

Description

Distribution parameters for emfrail

Usage

```
emfrail_dist(
  dist = "gamma",
  theta = 2,
  pvfm = -1/2,
  left_truncation = FALSE,
  basehaz = "breslow"
)
```

Arguments

`dist`	One of 'gamma', 'stable' or 'pvf'.
`theta`	A starting value for the 'outer' maximization with respect to the frailty parameter θ. Must be >0.
`pvfm`	Only relevant if `dist = 'pvf'` is used. It determines which PVF distribution should be used. Must be larger than -1 and not equal to 0.
`left_truncation`	Logical. Whether the data set represents left truncated survival times.
`basehaz`	A character string which determines how the baseline hazard is calculated. The default is "breslow", but other possible options are "weibull", "exponential" "gaussian", "logistic", "lognormal" or "loglogistic".

Details

The `theta` argument must be positive. In the case of gamma or PVF, this is the inverse of the frailty variance, i.e. the larger the `theta` is, the closer the model is to a Cox model. When `dist = "pvf"` and `pvfm = -0.5`, the inverse Gaussian distribution is obtained. For the positive stable distribution, the γ parameter of the Laplace transform is $\theta/(1+\theta)$, with the $alpha$ parameter fixed to 1.

Value

An object of the type `emfrail_dist`, which is mostly used to denote the supported frailty distributions in a consistent way.

See Also

`emfrail`, `emfrail_control`

Examples

```
emfrail_dist()
# Compound Poisson distribution:
emfrail_dist(dist = 'pvf', theta = 1.5, pvfm = 0.5)
# Inverse Gaussian distribution:
emfrail_dist(dist = 'pvf')
```

`emfrail_pll` *Profile log-likelihood calculation*

Description

Profile log-likelihood calculation

Usage

```
emfrail_pll(formula, data, distribution = emfrail_dist(), values)
```

Arguments

formula	Same as in emfrail
data	Same as in emfrail
distribution	Same as in emfrail
values	A vector of values on where to calculate the profile likelihood. See details.

Details

This function can be used to calculate the profile log-likelihood for different values of θ. The scale is that of theta as defined in emfrail_dist(). For the gamma and pvf frailty, that is the inverse of the frailty variance.

Value

The profile log-likelihood at the specific value of the frailty parameter

Note

This function is just a simple wrapper for emfrail() with the control argument a call from emfrail_control with the option opt_fit = FALSE. More flexibility can be obtained by calling emfrail with this option, especially for setting other emfrail_control parameters.

Examples

```
fr_var <- seq(from = 0.01, to = 1.4, length.out = 20)
pll_gamma <- emfrail_pll(formula = Surv(time, status) ~  rx + sex + cluster(litter),
 data =  rats,
 values = 1/fr_var )
 plot(fr_var, pll_gamma,
     type = "l",
     xlab = "Frailty variance",
     ylab = "Profile log-likelihood")

# check with coxph;
# attention: theta is the the inverse frailty variance in emfrail,
# but theta is the frailty variance in coxph.

pll_cph <- sapply(fr_var, function(th)
  coxph(data =  rats, formula = Surv(time, status) ~ rx + sex + frailty(litter, theta = th),
        method = "breslow")$history[[1]][[3]])

lines(fr_var, pll_cph, col = 2)

# Same for inverse gaussian
pll_if <- emfrail_pll(Surv(time, status) ~  rx + sex + cluster(litter),
                      rats,
                      distribution = emfrail_dist(dist = "pvf"),
                      values = 1/fr_var )

# Same for pvf with a positive pvfm parameter
```

```
pll_pvf <- emfrail_pll(Surv(time, status) ~  rx + sex + cluster(litter),
                       rats,
                       distribution = emfrail_dist(dist = "pvf", pvfm = 1.5),
                       values = 1/fr_var )

miny <- min(c(pll_gamma, pll_cph, pll_if, pll_pvf))
maxy <- max(c(pll_gamma, pll_cph, pll_if, pll_pvf))

plot(fr_var, pll_gamma,
     type = "l",
     xlab = "Frailty variance",
     ylab = "Profile log-likelihood",
     ylim = c(miny, maxy))
points(fr_var, pll_cph, col = 2)
lines(fr_var, pll_if, col = 3)
lines(fr_var, pll_pvf, col = 4)

legend(legend = c("gamma (emfrail)", "gamma (coxph)", "inverse gaussian", "pvf, m=1.5"),
       col = 1:4,
       lty = 1,
       x = 0,
       y = (maxy + miny)/2)
```

`logLik.emfrail` *Log-likelihood for* `emfrail` *fitted models*

Description

Log-likelihood for `emfrail` fitted models

Usage

```
## S3 method for class 'emfrail'
logLik(object, ...)
```

Arguments

`object`	An `emfrail` object
`...`	Other arguments

Details

The formula for the likelihood can be found in the manual which accompanies the package. Note that a constant is added. If we denote $\bar{n}$ the total number of events and $\bar{n_i}$ the total number of events at time point i, for each time point where events are observed, then this is equal to

$$\bar{n} - \sum_i \bar{n_i} \log \bar{n_i}.$$

This is mostly because of compatibility, i.e. to match the log-likelihood given by the survival package.

The df attribute of this object is equal to the number of regression coefficents plus 1. In general, the number of degrees of freedom for a frailty model is an unclear concept. For the coxph frailty fits, and in general for the shared frailty models fitted by penalized likelihood, the degrees of freedom is a number that depends on the penalization. However, even in that case, there is no straight forward interpretation or use of this quantity. The decision made here is because this would keep the likelihood ratio test for a covariate effect valid.

Value

An object of class logLik containing the marginal log-likelihood of the fitted model

---

plot.emfrail *Plots for emfrail objects*

Description

Plots for emfrail objects

Usage

```
## S3 method for class 'emfrail'
plot(
  x,
  type = c("hist", "hr", "pred"),
  newdata = NULL,
  lp = NULL,
  strata = NULL,
  quantity = "cumhaz",
  type_pred = c("conditional", "marginal"),
  conf_int = "adjusted",
  conf_level = 0.95,
  individual = FALSE,
  ...
)
```

Arguments

x	emfrail object, typically result of emfrail()
type	One (or more) of hist for a histogram of the estimated frailty values, hr for a plot of the conditional and marginal hazard ratio between two cases and pred for the predicted conditional and marginal cumulative hazard or survival for one case
newdata	A data.frame with values of the covariates. For type == "hr" the hazard ratio between the first two rows of newdata is calculated. For type == "pred" the prediction for the first row of newdata is calculated.

lp	A numeric vector of values of the linear predictor, each corresponding to a case. For type == "hr" the hazard ratio between the first two values of lp is calculated. For type == "pred" the prediction for the first value of lp is calculated.
strata	The name of the strata (if applicable) for which the prediction should be made.
quantity	For type == "pred" the predicted quantity; see quantity in predict.emfrail
type_pred	For type == "pred" the type of predicted quantity; see type in predict.emfrail
conf_int	For type == "pred" the type of confidence intervals; see conf_int in predict.emfrail
conf_level	The width of the confidence interval for type == "pred"; see conf_level in predict.emfrail
individual	For type == "pred" for drawing a curve when the rows of newdata refer to the same individual; see individual in predict.emfrail
...	Further arguments to be passed to the plot function

Value

Nothing

See Also

predict.emfrail, summary.emfrail, autoplot.emfrail.

Examples

```
mod_rec <- emfrail(Surv(start, stop, status) ~ treatment + number + cluster(id), bladder1,
control = emfrail_control(ca_test = FALSE, lik_ci = FALSE))

# Histogram of the estimated frailties
plot(mod_rec, type = "hist")

# hazard ratio between placebo and pyridoxine
newdata1 <- data.frame(treatment = c("placebo", "pyridoxine"),
                       number = c(1, 3))

plot(mod_rec, type = "hr", newdata = newdata1)

# predicted cumulative hazard for placebo, and number = 1
plot(mod_rec, type = "pred", newdata = newdata1[1,])

# predicted survival for placebo, and number = 1
plot(mod_rec, type = "pred", quantity = "survival", newdata = newdata1[1,])

# predicted survival for an individual that switches from
# placebo to pyridoxine at time = 15
newdata2 <- data.frame(treatment = c("placebo", "pyridoxine"),
                       number = c(1, 3),
                       tstart = c(0, 15),
                       tstop = c(15, Inf))

plot(mod_rec, type = "pred", quantity = "survival", newdata = newdata2, individual = TRUE)
```

`predict.emfrail` *Predicted hazard and survival curves from an* `emfrail` *object*

Description

Predicted hazard and survival curves from an `emfrail` object

Usage

```
## S3 method for class 'emfrail'
predict(
  object,
  newdata = NULL,
  lp = NULL,
  strata = NULL,
  quantity = c("cumhaz", "survival"),
  type = c("conditional", "marginal"),
  conf_int = NULL,
  individual = FALSE,
  conf_level = 0.95,
  ...
)
```

Arguments

`object`	An `emfrail` fit object
`newdata`	A data frame with the same variable names as those that appear in the `emfrail` formula, used to calculate the `lp` (optional).
`lp`	A vector of linear predictor values at which to calculate the curves. Default is 0 (baseline).
`strata`	The name of the strata (if applicable) for which the prediction should be made.
`quantity`	Can be `"cumhaz"` and/or `"survival"`. The quantity to be calculated for the values of `lp`.
`type`	Can be `"conditional"` and/or `"marginal"`. The type of the quantity to be calculated.
`conf_int`	Can be `"regular"` and/or `"adjusted"`. The type of confidence interval to be calculated.
`individual`	Logical. Are the observations in `newdata` from the same individual? See details.
`conf_level`	The width of the confidence intervals. By default, 95% confidence intervals are calculated.
`...`	Ignored

Details

The function calculates predicted cumulative hazard and survival curves for given covariate or linear predictor values; for the first, `newdata` must be specified and for the latter `lp` must be specified. Each row of `newdata` or element of `lp` is consiered to be a different subject, and the desired predictions are produced for each of them separately.

In `newdata` two columns may be specified with the names `tstart` and `tstop`. In this case, each subject is assumed to be at risk only during the times specified by these two values. If the two are not specified, the predicted curves are produced for a subject that is at risk for the whole follow-up time.

A slightly different behaviour is observed if `individual == TRUE`. In this case, all the rows of `newdata` are assumed to come from the same individual, and `tstart` and `tstop` must be specified, and must not overlap. This may be used for describing subjects that are not at risk during certain periods or subjects with time-dependent covariate values.

The two "quantities" that can be returned are named `cumhaz` and `survival`. If we denote each quantity with `q`, then the columns with the marginal estimates are named `q_m`. The confidence intervals contain the name of the quantity (conditional or marginal) followed by `_l` or `_r` for the lower and upper bound. The bounds calculated with the adjusted standard errors have the name of the regular bounds followed by `_a`. For example, the adjusted lower bound for the marginal survival is in the column named `survival_m_l_a`.

The `emfrail` only gives the Breslow estimates of the baseline hazard $\lambda_0(t)$ at the event time points, conditional on the frailty. Let $\lambda(t)$ be the baseline hazard for a linear predictor of interest. The estimated conditional cumulative hazard is then $\Lambda(t) = \sum_{s=0}^{t} \lambda(s)$. The variance of $\Lambda(t)$ can be calculated from the (maybe adjusted) variance-covariance matrix.

The conditional survival is obtained by the usual expression $S(t) = \exp(-\Lambda(t))$. The marginal survival is given by

$$\bar{S}(t) = E\left[\exp(-\Lambda(t))\right] = \mathcal{L}(\Lambda(t)),$$

i.e. the Laplace transform of the frailty distribution calculated in $\Lambda(t)$.

The marginal hazard is obtained as

$$\bar{\Lambda}(t) = -\log \bar{S}(t).$$

The only standard errors that are available from `emfrail` are those for $\lambda_0(t)$. From this, standard errors of $\log \Lambda(t)$ may be calculated. On this scale, the symmetric confidence intervals are built, and then moved to the desired scale.

Value

The return value is a single data frame (if `lp` has length 1, `newdata` has 1 row or `individual == TRUE`) or a list of data frames corresponding to each value of `lp` or each row of `newdata` otherwise. The names of the columns in the returned data frames are as follows: `time` represents the unique event time points from the data set, `lp` is the value of the linear predictor (as specified in the input or as calculated from the lines of `newdata`). By default, for each `lp` a data frame will contain the following columns: `cumhaz`, `survival`, `cumhaz_m`, `survival_m` for the cumulative hazard and survival, conditional and marginal, with corresponding confidence bands. The naming of the columns is explained more in the Details section.

Note

The linear predictor is taken as fixed, so the variability in the estimation of the regression coefficient is not taken into account. Does not support left truncation (at the moment). That is because, if `individual == TRUE` and `tstart` and `tstop` are specified, for the marginal estimates the distribution of the frailty is used to calculate the integral, and not the distribution of the frailty given the truncation.

For performance reasons, consider running with `conf_int = NULL`; the reason is that the `deltamethod` function that is used to calculate the confidence intervals easily becomes slow when there is a large number of time points for the cumulative hazard.

See Also

`plot.emfrail`, `autoplot.emfrail`

Examples

```
kidney$sex <- ifelse(kidney$sex == 1, "male", "female")
m1 <- emfrail(formula = Surv(time, status) ~  sex + age  + cluster(id),
              data =  kidney)

# get all the possible prediction for the value 0 of the linear predictor
predict(m1, lp = 0)

# get the cumulative hazards for two different values of the linear predictor
predict(m1, lp = c(0, 1), quantity = "cumhaz", conf_int = NULL)

# get the cumulative hazards for a female and for a male, both aged 30
newdata1 <- data.frame(sex = c("female", "male"),
                       age = c(30, 30))

predict(m1, newdata = newdata1, quantity = "cumhaz", conf_int = NULL)

# get the cumulative hazards for an individual that changes
# sex from female to male at time 40.
newdata2 <- data.frame(sex = c("female", "male"),
                      age = c(30, 30),
                      tstart = c(0, 40),
                      tstop = c(40, Inf))

predict(m1, newdata = newdata2,
        individual = TRUE,
        quantity = "cumhaz", conf_int = NULL)
```

`residuals.emfrail` *Residuals for frailty models*

Description

Residuals for frailty models

Usage

```
## S3 method for class 'emfrail'
residuals(object, type = "group", ...)
```

Arguments

`object`	An `emfrail` object
`type`	One of `cluster` or `individual`
`...`	Other arguments

Details

For cluster i, individual j and observation row k, we write the cumulative hazard contribution as

$$\Lambda_{ijk} = \exp(\beta^\top \mathbf{x}_{ijk})\Lambda_{0,ijk}$$

where $\Lambda_{0,ijk}$ is the baseline cumulative hazard correspinding to the row (i, j, k).

When `type == "individual"`, the returned residuals are equal to $z_i\Lambda_{ijk}$ where z_i is the estimated frailty in cluster i. When `type == "cluster"`, the returned residuals are equal to $\sum_{j,k}\Lambda_{ijk}$,

Value

A vector corresponding to the Martingale residuals, either for each cluster or for each individual (row of the data).

`summary.emfrail` *Summary for* `emfrail` *objects*

Description

Summary for `emfrail` objects

Usage

```
## S3 method for class 'emfrail'
summary(
  object,
  lik_ci = TRUE,
 print_opts = list(coef = TRUE, dist = TRUE, fit = TRUE, frailty = TRUE, adj_se = TRUE,
    verbose_frailty = TRUE),
  ...
)
```

Arguments

`object`	An object of class `emfrail`
`lik_ci`	Logical. Should the confidence intervals for the frailty parameter be calculated based on the likelihood? If not, they are calculated with the delta method.
`print_opts`	A list with options for printing the summary object. These include `coef`, `dist`, `fit`, `frailty`, `adj_se`, `verbose_frailty`.
`...`	Ignored

Details

Regardless of the fitted model, the following fields will be present in this object: `est_dist` (an object of class `emfrail_distribution`) with the estimated distribution, `loglik` (a named vector with the log-likelihoods of the no-frailty model, the frailty model, the likelihood ratio test statistic and the p-value of the one-sided likelihood ratio test), `theta` (a named vector with the estimated value of the parameter θ, the standard error, and the limits of a 95 is a data frame with the following columns: `id` (cluster identifier), `z` (empirical Bayes frailty estimates), and optional `lower_q` and `upper_q` as the 2.5

For the the PVF or gamma distributions, the field `fr_var` contains a transformation of `theta` to correspond to the frailty variance. The fields `pvf_pars` and `stable_pars` are for quantities that are calculated only when the distribution is PVF or stable. If the model contains covariates, the field `coefmat` contains the corresponding estimates. The p-values are based on the adjusted standard errors, if they have been calculated successfully (i.e. if they appear when prining the summary object). Otherwise, they are based on the regular standard errors.

Value

An object of class `emfrail_summary`, with some more human-readable results from an `emfrail` object.

See Also

`predict.emfrail`, `plot.emfrail`

Examples

```
data("bladder")
mod_gamma <- emfrail(Surv(start, stop, status) ~ treatment + cluster(id), bladder1)
summary(mod_gamma)
summary(mod_gamma, print_opts = list(frailty_verbose = FALSE))

# plot the Empirical Bayes estimates of the frailty
# easy way:
plot(mod_gamma, type = "hist")

# a fancy graph:
sum_mod <- summary(mod_gamma)
library(dplyr)
library(ggplot2)
```

```
# Create a plot just with the points
pl1 <- sum_mod$frail %>%
  arrange(z) %>%
  mutate(x = 1:n()) %>%
  ggplot(aes(x = x, y = z)) +
  geom_point()

# If the quantiles of the posterior distribution are
# known, then error bars can be added:
if(!is.null(sum_mod$frail$lower_q))
  pl1 <- pl1 + geom_errorbar(aes(ymin = lower_q, ymax = upper_q), alpha = 0.5)

pl1

# The plot can be made interactive!
# ggplot2 gives a warning about the "id" aesthetic, just ignore it
pl2 <- sum_mod$frail %>%
  arrange(z) %>%
  mutate(x = 1:n()) %>%
  ggplot(aes(x = x, y = z)) +
  geom_point(aes(id = id))

if(!is.null(sum_mod$z$lower_q))
  pl2 <- pl2 + geom_errorbar(aes(ymin = lower_q, ymax = upper_q, id = id), alpha = 0.5)

library(plotly)
ggplotly(pl2)

# Proportional hazards test
off_z <- log(sum_mod$frail$z)[match(bladder1$id, sum_mod$frail$id)]

zph1 <- cox.zph(coxph(Surv(start, stop, status) ~ treatment + cluster(id), data = bladder1))

# no sign of non-proportionality
zph2 <- cox.zph(coxph(Surv(start, stop, status) ~ treatment + offset(off_z), data = bladder1))

zph2
# the p-values are even larger; the frailty "corrects" for proportionality.
```

Index

autoplot, 2
autoplot.emfrail, 2, *8*, *17*, *20*

ca_test, 4

emfrail, 5, *12, 13*
emfrail_control, *6*, 10, *13*
emfrail_dist, *6*, *12*, 12
emfrail_pll, *8*, *12*, 13

logLik.emfrail, *8*, 15

plot.emfrail, *3*, *8*, 16, *20*, *22*
predict.emfrail, *3*, *8*, *17*, 18, *22*

residuals.emfrail, *8*, 20

summary.emfrail, *3*, *8*, *17*, 21