Package: dynsurv (via r-universe)

September 6, 2024

Title Dynamic Models for Survival Data

Version 0.4-7

Description Time-varying coefficient models for interval censored and
right censored survival data including 1) Bayesian Cox model
with time-independent, time-varying or dynamic coefficients for
right censored and interval censored data studied by Sinha et
al. (1999) <doi:10.1111/j.0006-341X.1999.00585.x> and Wang et
al. (2013) <doi:10.1007/s10985-013-9246-8>, 2) Spline based
time-varying coefficient Cox model for right censored data
proposed by Perperoglou et al. (2006)
<doi:10.1016/j.cmpb.2005.11.006>, and 3) Transformation model
with time-varying coefficients for right censored data using
estimating equations proposed by Peng and Huang (2007)
<doi:10.1093/biomet/asm058>.

Depends R (>= 3.0.2)

Imports data.table (>= 1.9.8), ggplot2, grDevices, grid, nleqslv,
splines2, stats, survival

LinkingTo BH (>= 1.54.0-2)

License GPL (>= 3)

LazyLoad Yes

LazyData Yes

NeedsCompilation yes

Encoding UTF-8

URL https://github.com/wenjie2wang/dynsurv

BugReports https://github.com/wenjie2wang/dynsurv/issues

RoxygenNote 7.3.2

Repository https://wenjie2wang.r-universe.dev

RemoteUrl https://github.com/wenjie2wang/dynsurv

RemoteRef HEAD

RemoteSha eae04ffca8eda947f73a56b60936d203472baccf

Contents

---

bayesCox   *Fit Bayesian Cox Model for Interval Censored Survival Data*

Description

Fit Bayesian Cox model with time-independent, time-varying or dynamic covariate coefficient. The fit is done within a Gibbs sampling framework. The reversible jump algorithm is employed for the dynamic coefficient model. The baseline hazards are allowed to be either time-varying or dynamic.

Usage

```
bayesCox(
  formula,
  data,
  grid = NULL,
  out = NULL,
  model = c("TimeIndep", "TimeVarying", "Dynamic"),
  base.prior = list(),
  coef.prior = list(),
  gibbs = list(),
  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 survival object as returned by the function Surv with type = "interval2". help(Surv) for details.
data	A data.frame in which to interpret the variables named in the formula.
grid	Vector of pre-specified time grid points for model fitting. It will be automatically set up from data if it is left unspecified in the function call. By default, it consists of all the unique finite endpoints (rounded to two significant numbers) of the censoring intervals after time zero. The grid specified in the function call determines the location of possible jumps. It should be sorted increasingly and cover all the finite non-zero endpoints of the censoring intervals. Inappropriate grid specified will be taken care by the function internally.
out	An optional character string specifying the name of Markov chain Monte Carlo (MCMC) samples output file. By default, the MCMC samples will be output to a temporary directory set by tempdir and saved in the returned bayesCox object after burning and thinning. If out is specified, the MCMC samples will be preserved in the specified text file.
model	Model type to fit. Available options are "TimeIndep", "TimeVarying", and "Dynamic". Partial matching on the name is allowed.
base.prior	List of options for prior of baseline lambda. Use list(type = "Gamma", shape = 0.1, rate = 0.1) for all models; list(type = "Const", value = 1) for Dynamic model when intercept = TRUE.
coef.prior	List of options for prior of coefficient beta. Use list(type = "Normal", mean = 0, sd = 1) for TimeIndep model; list(type = "AR1", sd = 1) for TimeVarying and Dynamic models; list(type = "HAR1", shape = 2, scale = 1) for TimeVarying and Dynamic models.
gibbs	List of options for Gibbs sampler.
control	List of general control options.
...	Other arguments that are for futher extension.

Details

To use default hyper parameters in the specification of either base.prior or coef.prior, one only has to supply the name of the prior, e.g., list(type = "Gamma"), list(type = "HAR1").

The gibbs argument is a list of components:

iter: Number of iterations, default 3000;

burn: Number of burning, default 500;

thin: Number of thinning, default 1;

verbose: A logical value, default TRUE. If TRUE, print the iteration;

nReport: Print frequency, default 100.

The control argument is a list of components:

intercept: A logical value, default `FALSE`. If `TRUE`, the model will estimate the intercept, which is the log of baseline hazards. If `TRUE`, please remember to turn off the direct estimation of baseline hazards, i.e., `base.prior = list(type = "Const")`

a0: Multiplier for initial variance in time-varying or dynamic models, default 100;

eps0: Size of auxiliary uniform latent variable in dynamic model, default 1.

For users interested in extracting MCMC sampling information from the output files, the detail of the output files is presented as follows: Let k denote the number of time points (excluding time zero) specified in grid, ck equal 1 for model with time-invariant coefficients; ck equal k otherwise, and p denote the number of covariates. Then the each sample saved in each row consists of the following possible parts.

Part 1: The first k numbers represent the jump size of baseline hazard function at each time grid. If we take the column mean of the first k columns of the output file, we will get the same numbers with `obj$est$lambda`, where `obj` is the `bayesCox` object returned by the function.

Part 2: The sequence from $(k+1) to (k+ck*p)$ represent the coefficients of covariates at the time grid. The first k numbers in the sequence are the coefficients for the first covariate at the time grid; The second k numbers' sub-sequence are the coefficients for the second covariate and so on.

Part 3: The sequence from $(k+ck*p+1)$ to $(k+ck*p+p)$ represents the sampled latent variance of coefficients.

Part 4: The sequence from $(k + ck * p + p + 1)$ to $(k + 2 * ck * p + p)$ represents the indicator of whether there is a jump of the covariate coefficients at the time grid. Similar with Part 2, the first k numbers' sub-sequence is for the first covariate, the second k numbers' sub-sequence is for the second covariate, and so on.

For the model with time-independent coefficients, the output file only has Part 1 and Part 2 in each row; For time-varying coefficient model, the output file has Part 1, 2, and 3; The output file for the dynamic model has all the four parts. Note that the dynamic baseline hazard will be taken as one covariate. So p needs being replaced with $(p+1)$ for model with dynamic baseline hazard rate. No function in the package actually needs the Part 1 from the output file now; The Part 2 is used by function `coef` and `survCurve`; The Part 3 is needed by function `nu`; Function `jump` extracts the Part 4.

Value

An object of S3 class `bayesCox` representing the fit.

References

X. Wang, M.-H. Chen, and J. Yan (2013). Bayesian dynamic regression models for interval censored survival data with application to children dental health. Lifetime data analysis, 19(3), 297–316.

X. Wang, X. Sinha, J. Yan, and M.-H. Chen (2014). Bayesian inference of interval-censored survival data. In: D. Chen, J. Sun, and K. Peace, Interval-censored time-to-event data: Methods and applications, 167–195.

X. Wang, M.-H. Chen, and J. Yan (2011). Bayesian dynamic regression models for interval censored survival data. Technical Report 13, Department of Statistics, University of Connecticut.

D. Sinha, M.-H. Chen, and S.K. Ghosh (1999). Bayesian analysis and model selection for interval-censored survival data. *Biometrics* 55(2), 585–590.

See Also

coef.bayesCox, jump.bayesCox, nu.bayesCox, plotCoef, plotJumpTrace, plotNu, survCurve, survDiff, and plotSurv.

Examples

```
## Not run:

library(dynsurv)
set.seed(1216)

## breast cancer data
data(bcos)
mydata <- bcos
myformula <- Surv(left, right, type = "interval2") ~ trt

### Fit time-independent coefficient model
fit0 <- bayesCox(myformula, mydata, model = "TimeIndep",
                 base.prior = list(type = "Gamma", shape = 0.1, rate = 0.1),
                 coef.prior = list(type = "Normal", mean = 0, sd = 1),
                 gibbs = list(iter = 100, burn = 20, thin = 1, verbose = FALSE))

## plot coefficient estimates
plotCoef(coef(fit0, level = 0.9))

## Plot the estimated survival function for given new data
newDat <- data.frame(trt = c("Rad", "RadChem"))
row.names(newDat) <- unique(newDat$trt)
plotSurv(survCurve(fit0, newDat), conf.int = TRUE)

### Fit time-varying coefficient model
fit1 <- bayesCox(myformula, mydata, model = "TimeVary",
                 base.prior = list(type = "Gamma", shape = 0.1, rate = 0.1),
                 coef.prior = list(type = "AR1", sd = 1),
                 gibbs = list(iter = 100, burn = 20, thin = 1,
                              verbose = TRUE, nReport = 20))

plotCoef(coef(fit1))
plotSurv(survCurve(fit1), conf.int = TRUE)

### Fit dynamic coefficient model with time-varying baseline hazards
fit2 <- bayesCox(myformula, mydata, model = "Dynamic",
                 base.prior = list(type = "Gamma", shape = 0.1, rate = 0.1),
                 coef.prior = list(type = "HAR1", shape = 2, scale = 1),
                 gibbs = list(iter = 100, burn = 20, thin = 1,
                              verbose = TRUE, nReport = 20))

plotCoef(coef(fit2))
```

```
    plotJumpTrace(jump(fit2))
    plotJumpHist(jump(fit2))
    plotNu(nu(fit2))
    plotSurv(survCurve(fit2), conf.int = TRUE)

    ## Plot the coefficient estimates from three models together
    plotCoef(rbind(coef(fit0), coef(fit1), coef(fit2)))

    ### Fit dynamic coefficient model with dynamic hazards (in log scales)
    fit3 <- bayesCox(myformula, mydata, model = "Dynamic",
                     base.prior = list(type = "Const"),
                     coef.prior = list(type = "HAR1", shape = 2, scale = 1),
                     gibbs = list(iter = 100, burn = 20, thin = 1,
                                  verbose = TRUE, nReport = 20),
                     control = list(intercept = TRUE))

    plotCoef(coef(fit3))
    plotJumpTrace(jump(fit3))
    plotJumpHist(jump(fit3))
    plotNu(nu(fit3))
    plotSurv(survCurve(fit3), conf.int = TRUE)

    ## Plot the estimated survival function and the difference
    plotSurv(survCurve(fit3, newdata = newDat, type = "survival"),
             legendName = "Treatment", conf.int = TRUE)
    plotSurv(survDiff(fit3, newdata = newDat, type = "survival"),
             legendName = "Treatment", conf.int = TRUE, smooth = TRUE)

    ## extract MCMC samples
    mcmc_list <- bayesCoxMcmc(fit3, part = "coef")
    posterior_coef <- mcmc_list$coef
    ## posterior probabilities of hazard ratio of RadChem (vs. Rad)
    ## greater than 1 at time 10
    posterior_coef[covariate == "trtRadChem" & time == 10, mean(exp(coef) > 1)]


    ## End(Not run)
```

---

bayesCoxMcmc   *Get the MCMC Samples from* bayesCox

Description

Returns the MCMC samples produced by bayesCox into data frames.

Usage

```
bayesCoxMcmc(object, parts = c("h0", "coef"), ...)
```

Arguments

`object`	A bayesCox object
`parts`	A character vector specifying the parts to be exacted from the MCMC output text file produced by bayesCox. One or more following options can be specified: `"h0"` for baseline hazard function, `"coef"` for covariate coefficients, `"nu"` for sampled latent variance of coefficients, `"jump"` for indicators of jumps, and `"all"` for all of the above. The default value is `c("h0", "beta")`.
`...`	Other arguments that are not used now.

---

bcos *Breast Cancer Data*

Description

The breast cancer data from Finkelstein (1985) has been analyzed extensively for illustrating new methods in modeling interval censored data. The objective of the study was to compare the time to cosmetic deterioration between two groups: 46 patients receiving radiotherapy only and 48 patients receiving radiotherapy plus chemotherapy. Because the detection of deterioration required a clinic visit, the 56 women who experience deterioration were interval censored, and the other38 women who did not were right censored.

Format

bcos is a data frame with 94 observations and 3 columns.

left: left censoring time;

right: right censoring time;

trt: treatment (Rad = radiotherapy only, RadChem = radiotherapy plus chemotherapy).

bcos.grid is a numeric vector of grid time points.

References

D.M. Finkelstein, and R.A. Wolfe (1985). A semiparametric model for regression analysis of interval-censored failure time data. *Biometrics* 41: 731–740.

Examples

data(bcos)

`coef.bayesCox`   *Extract Coefficients from Bayesian Cox Model*

Description

Extract coefficient values from bayesCox fitting results, and summarize the posterior mean, posterior 2.5% and 97.5% quantiles into a data frame.

Usage

```
## S3 method for class 'bayesCox'
coef(object, ...)
```

Arguments

object	An object returned by function bayesCox.
...	Optional arguments. Currently, the only applicable arguemnt is `level` for the credible level. The default value is 0.95.

Value

A data.frame with 6 columns (`"Low"`, `"Mid"`, `"High"`, `"Time"`, `"Cov"`, `"Model"`), where `"Low"` and `"High"` are the posterior 2.5% and 97.5% quantiles as default; `"Mid"` is the posterior mean; `"Cov"` and `"Model"` contain character values of the covariates and model types.

See Also

bayesCox, and plotCoef.

Examples

```
## See the examples in bayesCox.
```

`coef.splineCox`   *Extract Coefficients from Spline Base Cox Model*

Description

Extract coefficient values from splineCox fitting results, and summarize the point estimate and 95% confidence band into a data frame.

Usage

```
## S3 method for class 'splineCox'
coef(object, ...)
```

Arguments

object	An object returned by function splineCox.
...	Optional arguments. Currently, the only applicable arguemnt is level for the credible level. The default value is 0.95.

Value

A data.frame with 6 columns ("Low", "Mid", "High", "Time", "Cov", "Model"), where "Mid" is the point estimates; "Low" and "High" are the point estimates plus and minus 1.96 times standard deviations (under default level); "Cov" and "Model" contain character values of the covariates and model type.

Note

It essentially expand the break points, and then call function coxph in package survival

See Also

[splineCox](#), and [plotCoef](#).

Examples

```
## See the examples in splineCox.
```

---

coef.tvTran	*Extract Coefficients from Time-varying Transformation Model*

Description

Extract coefficient values from tvTran fitting results, and summarize the point estimate and 95% credible band into a data frame.

Usage

```
## S3 method for class 'tvTran'
coef(object, ...)
```

Arguments

object	An object returned by function tvTran.
...	Optional arguments. Currently, the only applicable arguemnt is level for the credible level. The default value is 0.95.

Value

A data.frame with 6 columns (`"Low"`, `"Mid"`, `"High"`, `"Time"`, `"Cov"`, `"Model"`), where `"Mid"` is the point estimates; `"Low"` and `"High"` are the 2.5% and 97.5% quantiles estimates from resampling method as default; `"Cov"` and `"Model"` contain character values of the covariates and model type.

See Also

`tvTran`, and `plotCoef`.

Examples

```
## See the examples in tvTran.
```

---

   is.bayesCox   *Is* bayesCox *Class Object*

Description

Checks if the input object is of Class bayesCox.

Usage

```
is.bayesCox(object)
```

Arguments

object   An R object.

---

   jump   *Generic function for jump information*

Description

Generic function for jump information

Usage

```
jump(object, ...)

## S3 method for class 'bayesCox'
jump(object, ...)
```

Arguments

object   An object returned by function bayesCox.

...   Other arguments.

Value

A data.frame with 3 columns ("Count", "Iter", "Cov"), where "Count" is the number of coefficient pieces (jumps) for each iteration; Iter is the iteration number; Cov contains the character values of the covariates.

Methods (by class)

- jump(bayesCox): Extract Jump Information from Bayesian Dyanmic Model

  Extract number of coefficient pieces from bayesCox fitting results, and summarize them into a data frame. It is only applicable when model="Dynamic" is specified.

See Also

bayesCox, plotJumpTrace, and plotJumpHist.

Examples

```
## See the examples in bayesCox
```

---

nu   *Generic function for the latent variance of coefficients*

Description

Generic function for the latent variance of coefficients

Usage

```
nu(object, ...)

## S3 method for class 'bayesCox'
nu(object, ...)
```

Arguments

object	An object returned by function bayesCox.
...	Other arguments.

Value

A data.frame with 4 columns ("Iter", "Model", "Cov","Value"), where Iter is the iteration number; Model and Cov contain the character values of the model type and covariates.

Methods (by class)

- nu(bayesCox): Extract Latent Variance from Bayesian Cox Model

  Extract latent variance of coefficients from bayesCox fitting results, and summarize them into a data frame. It is applicable when model="TimeVarying" or model="Dynamic", and coef.prior=list(type="HAR1").

  For details, see section on prior model in Wang (2013) and Wang (2014). The latent variance of coefficients in prior model was denoted as omega in Wang (2013).

References

X. Wang, M.-H. Chen, and J. Yan (2013). Bayesian dynamic regression models for interval censored survival data with application to children dental health. Lifetime data analysis, 19(3), 297–316.

X. Wang, X. Sinha, J. Yan, and M.-H. Chen (2014). Bayesian inference of interval-censored survival data. In: D. Chen, J. Sun, and K. Peace, Interval-censored time-to-event data: Methods and applications, 167–195.

See Also

bayesCox, and plotNu.

Examples

```
## See the examples in bayesCox.
```

---

plotCoef   *Plot Coefficient Function*

Description

Plot coefficient values formatted in a data frame returned by function coef.

Usage

```
plotCoef(object, smooth = FALSE, ...)
```

Arguments

object	A data.frame returned by function coef.
smooth	A logical value, default FALSE. If TRUE, plot the coefficients as smooth lines; otherwise, plot the coefficients as piece-wise constant step functions.
...	Other arguments.

Details

To plot estimated coefficient functions from different models together, one can first combine the data frames returned by `coef`, and then call `plotCoef`, for example, `plotCoef(rbind(coef(fit1),coef(fit2)))`.

To specify the time range of the plot, one can either utilize the ggplot functionality, say `plotCoef(coef(fit))` `+ xlim(2, 10)`; or manipulate the data frame first before calling `plotCoef`, e.g., `plotCoef(subset(coef(fit),` `Time > 2 & Time < 10))`.

Value

A ggplot object.

See Also

`coef.bayesCox`, `coef.splineCox`, and `coef.tvTran`.

Examples

```
## See the examples in bayesCox, splineCox, and tvTran.
```

---

plotJump   *Plot Jump Information in Bayesian Dynamic Model*

Description

`plotJumpTrace` plots the MCMC history of the number of pieces. `plotJumpHist` plots the histogram of the number of pieces. The input data frame is returned by function `jump`.

Usage

```
plotJumpTrace(object, ...)

plotJumpHist(object, ...)
```

Arguments

object   A data.frame returned by function `jump`.
...   Other arguments.

Value

A ggplot object.

See Also

`jump.bayesCox`.

Examples

```
## See the examples in bayesCox
```

---

plotNu   *Plot Latent Variance in Bayesian Cox Model*

Description

Plot the latent variance nu when the hierarchical AR(1) process prior is used for the bayesCox
model. It is applicable when model="TimeVarying" or model="Dynamic", and coef.prior=list(type="HAR1").
The input data frame is returned by function nu.

Usage

```
plotNu(object, ...)
```

Arguments

object	A data.frame returned by the function nu.
...	Other arguments.

Value

A ggplot object.

See Also

[nu.bayesCox](#).

Examples

```
## See the examples in bayesCox
```

---

plotSurv   *Plot Survival Curves (or Cumulative Hazard Function) and their dif-
                                ference*

Description

Plot the survival curves (or cumulative hazard) and their difference for objects returned by function
survCurve or survDiff. By using ggplot2 plotting system, the plots generated are able to be
further customized properly.

Usage

```
plotSurv(
  object,
  legendName = "",
  conf.int = FALSE,
  smooth = FALSE,
  lty,
  col,
  ...
)
```

Arguments

`object`	An object returned by function `survCurve` or `survDiff`.
`legendName`	An optional name for the figure legend.
`conf.int`	A logical value indicating whether to plot the credible interval(s).
`smooth`	A logical value, default `FALSE`. If `TRUE`, plot the coefficients as smooth lines; otherwise, plot the coefficients as piece-wise constant step functions.
`lty`	An optional numeric vector indicating line types specified to different groups: 0 = blank, 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash.
`col`	An optional character or numeric vector indicating line colors specified to different groups.
`...`	Other arguments for future usage.

Value

A `ggplot` object.

See Also

[bayesCox](#), [survCurve](#), and [survDiff](#).

Examples

```
## See the examples in bayesCox.
```

---

splineCox   *Fit Spline Based Cox Model for Right Censored Survival Data*

Description

Rearrange the rignt censored survival data in a counting process style. Model the time-varying co-efficient function using B-splines. The fit is done by introducing pseudo time-dependent covariates and then calling function coxph in **survival** package.

Usage

```
splineCox(formula, data, 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 survival object as returned by the Surv function.
data	A data.frame in which to interpret the variables named in the formula.
control	List of control options.

Details

The control argument is a list of components:

df: degree of freedom for the B-splines, default 5;

knots: interior knots point, default NULL. If NULL, the knots will be automatically choosen;

boundary: lower and upper boundaries for the spline function, default NULL. If NULL, the minimun and maximun finite event time or censoring time will be specified.

Value

An object of S3 class splineCox representing the fit.

Note

This function is essentially a wrapper function of coxph for the expanded data set. It does not implements the algorithm disscussed in the reference paper. These authors implemented their algorithm into a tvcox package, which is more efficient for larger data set, but may not be stable compared to coxph.

References

Perperoglou, A., le Cessie, S., & van Houwelingen, H. C. (2006). A fast routine for fitting Cox models with time varying effects of the covariates. Computer Methods and Programs in Biomedicine, 81(2), 154–161.

See Also

[coef.splineCox](), [plotCoef]().

Examples

```
## Not run:
## Attach the veteran data from the survival package
mydata <- survival::veteran
mydata$celltype <- relevel(mydata$celltype, ref = "large")
myformula <- Surv(time, status) ~ karno + celltype
```

```
## Fit the time-varying transformation model
fit <- splineCox(myformula, mydata, control = list(df = 5))

## Plot the time-varying coefficient function between two time points
plotCoef(subset(coef(fit), Time > 15 & Time < 175), smooth = TRUE)

## End(Not run)
```

---

survCurve   *Estimated Survival Function or Cumulative Hazard Function*

Description

Estimated survival function or cumulative hazard function from posterior sample for an object returned by function bayesCox.

Usage

```
survCurve(
  object,
  newdata,
  type = c("survival", "cumhaz"),
  level = 0.95,
  centered = FALSE,
  ...
)
```

Arguments

object	An object returned by function bayesCox.
newdata	An optional data frame used to generate a design matrix.
type	An optional character value indicating the type of function to compute. The possible values are "survival" and "cumhaz". The former means the estimated survival function; the latter represents the estimated cumulative hazard function for the given newdata.
level	A numerical value between 0 and 1 indicating the level of cradible band.
centered	A logical value. If TRUE, the mean function for the given newdata will be computed. The default is FALSE.
...	Other arguments for further usage.

Details

The estimated survival curve is a step function representing the posterior mean survival proportion at the given time grid from the posterior sample. The credible interval for the survival curve is constructed based on the quantiles of all the survival curves from posterior sample at given credible level. More details were available in Section posterior computation of Wang (2016).

Value

A data frame with column: "Low", "Mid", "High", "Time", "Design", and "type", and attribute, "surv" valued as "survCurve".

References

Wang, W., Chen, M. H., Chiou, S. H., Lai, H. C., Wang, X., Yan, J., & Zhang, Z. (2016). Onset of persistent pseudomonas aeruginosa infection in children with cystic fibrosis with interval censored data. *BMC Medical Research Methodology*, 16(1), 122.

See Also

bayesCox, survDiff, and plotSurv.

Examples

```
## See the examples in bayesCox.
```

---

survDiff	*Estimated Difference Between Survival or Cumulative Hazard Functions*

Description

survDiff returns estimated survival function or cumulative function from posterior estimates. Note that currently, the function is only applicable to the Bayesian dynamic Cox model with dynamic hazard, where the control argument is specified to be control = list(intercept = TRUE) in function bayesCox.

Usage

```
survDiff(object, newdata, type = c("survival", "cumhaz"), level = 0.95, ...)
```

Arguments

object	An object returned by function bayesCox.
newdata	An optional data frame used to generate a design matrix. Note that it must lead to a design matrix with two different design.
type	An optional character value indicating the type of function to compute. The possible values are "survival" and "cumhaz". The former means the estimated survival function; the latter represents the estimated cumulative hazard function for the given newdata.
level	A numerical value between 0 and 1 indicating the level of cradible band.
...	Other arguments for further usage.

Details

The estimated difference between survival curves is a step function representing the difference between the posterior mean survival proportion at the given time grid from the posterior sample. Its credible interval is constructed based on the quantiles of all the pair difference between the survival curves from posterior sample at given credible level.

Value

A data frame with column: "Low", "Mid", "High", "Time", "Design", and "type", and attribute, "surv" valued as "survDiff".

References

Wang, W., Chen, M. H., Chiou, S. H., Lai, H. C., Wang, X., Yan, J., & Zhang, Z. (2016). Onset of persistent pseudomonas aeruginosa infection in children with cystic fibrosis with interval censored data. *BMC Medical Research Methodology*, 16(1), 122.

See Also

bayesCox, survCurve, and plotSurv.

Examples

```
## See the examples in bayesCox.
```

---

tooth   *Tooth Data*

Description

The tooth data was from a longitudinal prospective dental study performed in Flanders (Belgium) in 1996 – 2001. Every one of 4,386 randomly sampled children in the cohort was examined annually by one of 16 trained dentists, resulting at most 6 dental observations for each child. The outcome of interest was the time to emergence of permanent tooth 24, which was either interval censored (2,775, 63%) or right censored (1,611, 37%).

Format

tooth is a data frame with 4,386 observations and 7 columns

id: children's id;

left: left censoring time;

right: right censoring time where infinity is coded as 999.

sex: gender of children (0 = boy, 1 = girl);

dmf: status of the primary predecessor of this tooth (0 = sound, 1 = delayed, missing or filled);

rightInf: right censoring time where infinity is coded as `Inf`;

rightNA: right censoring time where infinity is coded as `NA`.

`tooth.grid` is a numeric vector of grid time points.

Source

Adapted from the data set available at <https://grass.upc.edu/en/software/tooth24>.

References

J. Vanobbergen, L. Martens, D. Declerck, and M. Lesaffre (2000). The signal tandmobiel(r) project: a longitudinal intervention oral health promotion study in Flanders (Belgium): baseline and first year results. *European Journal of Paediatric Dentistry* 2, 87.

G. Gomez, M. Calle, R. Oller, and K. Langohr (2009). Tutorial on methods for interval-censored data and their implementation in R. *Statistical Modeling* 9(4), 259.

Examples

```
data(tooth)
```

---

tvTran	*Fit Time-varying Transformation Model for Right Censored Survival Data*

Description

Unlike the time-varying coefficient Cox model, the transformation model fomulates the temporal covariate effects in terms of survival function, i.e.,

$$S(t|X) = g(\beta_0(t)'X),$$

where $g(z) = exp(-exp(z))$. It can be viewed as a functional generalized linear model with response $I(T > t)$, and other transformation function is possible. The time-varying coefficients are solved a set of estimating equations sequentially.

Usage

```
tvTran(formula, data, 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 survival object as returned by the `Surv` function.
data	A data.frame in which to interpret the variables named in the `formula`.
control	List of control options.

Details

Note that because the time-varying coefficient function is connected to the survival function, it has a different interpretation of the time-varying coefficient function in Cox model.

The `control` argument is a list of components:

resample A logical value, default `TRUE`. If `TRUE`, the model will estimate a 95% confidence band by resampling method.

R Number of resamplings, default 30.

Value

An object of S3 class `tvTran` representing the fit.

References

Peng, L. and Huang, Y. (2007). Survival analysis with temporal covariate effects. *Biometrika* 94(3), 719–733.

See Also

`coef.tvTran`, `plotCoef`.

Examples

```
## Not run:
## Attach the veteran data from the survival package
mydata <- survival::veteran
mydata$celltype <- relevel(mydata$celltype, ref = "large")
myformula <- Surv(time, status) ~ karno + celltype

## Fit the time-varying transformation model
fit <- tvTran(myformula, mydata, control = list(resample = TRUE, R = 30))

## Plot the time-varying coefficient function between two time points
plotCoef(subset(coef(fit), Time > 15 & Time < 175))

## End(Not run)
```

Index