Package 'dynsurv'

December 24, 2016

Title Dynamic Models for Survival Data

Version 0.3-4

Date 2016-12-24

Description Functions to fit time-varying coefficient models for interval censored and right censored survival data. Three major approaches are implemented: 1) Bayesian Cox model with time-independent, time-varying or dynamic coefficients for right censored and interval censored data; 2) Spline based time-varying coefficient Cox model for right censored data; 3) Transformation model with time-varying coefficients for right censored data using estimating equations.

Depends R (>= 3.0.2), survival

Imports ggplot2, grDevices, grid, nleqslv, plyr, reshape, stats, utils

LinkingTo BH (>= 1.54.0-2)

License GPL (>= 3)

LazyLoad Yes

LazyData Yes

NeedsCompilation yes

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

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

RoxygenNote 5.0.1

Author Wenjie Wang [cre, aut],
Ming-Hui Chen [aut],
Xiaojing Wang [aut],
Jun Yan [aut]

Maintainer Wenjie Wang <`wenjie.2.wang@uconn.edu`>

R topics documented:

`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 = "mcmc.txt", 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 `Surv` function.
`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 of the censoring intervals after time zero. The `grid` specified in the function call must be sorted, and covers all the finite non-zero endpoints of the censoring intervals.
`out`	Name of Markov chain Monte Carlo (MCMC) samples output file. Each row contains one MCMC sample information. The file is needed for those functions further summarizing estimation results in this package.
`model`	Model type to fit.
`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.

Details

For application, it is recommended that users preprocess the data by rounding down (up) the left (right) endpoints of the censoring intervals to at most three significant digits and leave the `grid` unspecified in the function call to reduce probably unnecessary computational burden. It also helps avoid possible errors caused by mispecifed `grid`. For example, the left endpoints can be rounded down in the nearest 0.1 unit of time by `left <- floor(left * 10) / 10` and the right endpoints can similarly be rounded up by `right <- ceiling(right * 10) / 10`.

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.

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:
############################################################################
### Attach one of the following two data sets
############################################################################

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

## tooth data
## data(tooth) ## load tooth and tooth.grid
## mydata <- tooth
## mygrid <- tooth.grid
## myformula <- Surv(left, rightInf, type = "interval2") ~ dmf + sex

############################################################################
### Fit Bayesian Cox models
############################################################################

## Fit time-independent coefficient model
fit0 <- bayesCox(myformula, mydata, out = "tiCox.txt",
                 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 = TRUE, nReport = 5))
plotCoef(coef(fit0, level = 0.9))

## Fit time-varying coefficient model
fit1 <- bayesCox(myformula, mydata, out = "tvCox.txt",
                 model = "TimeVarying",
                 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 = 5))
plotCoef(coef(fit1))

## Fit dynamic coefficient model with time-varying baseline hazards
fit2 <- bayesCox(myformula, mydata, out = "dynCox1.txt",
                 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 = 5))
plotCoef(coef(fit2))
plotJumpTrace(jump(fit2))
plotJumpHist(jump(fit2))
plotNu(nu(fit2))

## 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, out = "dynCox2.txt",
                 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=5),
                 control = list(intercept = TRUE))
plotCoef(coef(fit3))
plotJumpTrace(jump(fit3))
plotJumpHist(jump(fit3))
plotNu(nu(fit3))

## Plot the estimated survival function and hazard function
newDat <- bcos[c(1L, 47L), ]
row.names(newDat) <- c("Rad", "RadChem")
plotSurv(survCurve(fit3, newdata = newDat, type = "survival"),
         legendName = "Treatment", conf.int = TRUE)
plotSurv(survDiff(fit3, newdata = newDat, type = "cumhaz"),
         legendName = "Treatment", conf.int = TRUE, smooth = TRUE)

## End(Not run)
```

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.
```

dynsurv — *dynsurv: Time-varying coefficient models for interval censored data.*

Description

Functions to fit time-varying coefficient models for interval censored and right censored survival data. Three major approaches are implemented:

1. Bayesian Cox model with time-independent, time-varying or dynamic coefficients for right censored and interval censored data;
2. Spline based time-varying coefficient Cox model for right censored data;
3. Transformation model with time-varying coefficients for right censored data using estimating equations.

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)

- `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)

- `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(` 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 difference*

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 coefficient 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

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

`survCurve` returns estimated suvival function or cumulative function from posterior sample. Note that the function is currently 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

```
survCurve(object, newdata, type = c("survival", "cumhaz"),
          level = 0.95, centered = FALSE, cache = 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`.
`cache`	A logical value. If `TRUE`, the cache RData file will be generated in the working directory to improve the performance of function `survCurve` and `survDiff`. This option would be quite helpful if the number of MCMC sample is large.
`...`	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 suvival 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, cache = FALSE, ...)
```

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.
`cache`	A logical value. If `TRUE`, the cache RData file will be generated in the working directory to improve the performance of function `survCurve` and `survDiff`. This option would be quite helpful if the number of MCMC sample is large.
`...`	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".

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 `http://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

L. Peng, and Y. Huang (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