Package: surtvep (via r-universe)

October 11, 2024

Title Cox Non-Proportional Hazards Model with Time-Varying Coefficients

Version 1.0.0

URL https://github.com/UM-KevinHe/surtvep, https://um-kevinhe.github.io/surtvep/

BugReports https://github.com/UM-KevinHe/surtvep/issues

Description Fit Cox non-proportional hazards models with time-varying coefficients. Both unpenalized procedures (Newton and proximal Newton) and penalized procedures (P-splines and smoothing splines) are included using B-spline basis functions for estimating time-varying coefficients. For penalized procedures, cross validations, mAIC, TIC or GIC are implemented to select tuning parameters. Utilities for carrying out post-estimation visualization, summarization, point-wise confidence interval and hypothesis testing are also provided. For more information, see Wu et al. (2022) <doi:10.1007/s10985-021-09544-2> and Luo et al. (2023) <doi:10.1177/09622802231181471>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, splines, ggplot2, ggpubr, stats, tibble, rlang

Depends R (>= 4.0)

Suggests knitr, rmarkdown

NeedsCompilation yes

Author Lingfeng Luo [aut, cre], Wenbo Wu [aut], Kevin He [aut]

Maintainer Lingfeng Luo <lfluo@umich.edu>

Repository CRAN

Date/Publication 2023-10-17 10:00:02 UTC

Contents

`baseline` *calculating baseline hazard and baseline cumulative hazard using the result from a* `coxtv` *or* `coxtp` *object*

Description

The baseline estimation is the baseline hazard at each observed failure time when holding all the covariates to be zero.

Usage

```
baseline(fit)
```

Arguments

`fit`	model from `coxtv` or `coxtp`.

Value

A list with three components:

`time`	the unique observed failure times.
`hazard`	the baseline hazard corresponding to each unique failure time point.
`cumulHaz`	the cumulative baseline hazard corresponding to each unique failure time point.

Examples

```
data(ExampleData)
z <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event

fit   <- coxtv(event = event, z = z, time = time)
base.est <- baseline(fit)
```

`confint.coxtp` *get confidence intervals of time-varying coefficients from a fitted object*

Description

Get confidence intervals of time-varying coefficients from a fitted `coxtv` or `coxtp` object.

Usage

```
## S3 method for class 'coxtp'
confint(object, parm, level = 0.95, time, ...)

## S3 method for class 'coxtv'
confint(object, parm, level = 0.95, time, ...)
```

Arguments

`object`	fitted `"coxtv"` model.
`parm`	the names of parameters.
`level`	the confidence level. The default value is 0.95.
`time`	the time points for which the confidence intervals to be estimated. The default value is the unique observed event times in the dataset fitting the time-varying effects model.
`...`	other parameters to function

Value

A list where each element corresponds to one of the parameters specified in `parm`. Each element in the list is a matrix, with rows corresponding to the specified `time` points and three columns representing the estimated values of the parameter, and the lower and upper bounds of the confidence interval at the specified confidence `level`. The length of the list is determined by the number of parameters in `parm`, and each matrix has rows equal to the number of specified `time` points.

A list where each element corresponds to one of the parameters specified in `parm`. Each element in the list is a matrix, with rows corresponding to the specified `time` points and three columns representing the estimated values of the parameter, and the lower and upper bounds of the confidence interval at the specified confidence `level`. The length of the list is determined by the number of parameters in `parm`, and each matrix has rows equal to the number of specified `time` points.

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtp(event = event, z = z, time = time)
IC  <- IC(fit)
confint(IC$model.mAIC)


data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtv(event = event, z = z, time = time)
confint <- confint(fit)
```

`coxtp` *fit a Cox non-proportional hazards model with P-spline or Smoothing-spline, with penalization tuning parameter chosen by information criteria or cross-validation*

Description

Fit a Cox non-proportional hazards model via penalized maximum likelihood.

Usage

```
coxtp(
  event,
  z,
  time,
  strata = NULL,
  penalty = "Smooth-spline",
  nsplines = 8,
  lambda = c(0.1, 1, 10),
  degree = 3L,
  knots = NULL,
  ties = "Breslow",
  tol = 1e-06,
  iter.max = 20L,
  method = "ProxN",
  gamma = 1e+08,
  btr = "dynamic",
  tau = 0.5,
  stop = "ratch",
  parallel = FALSE,
```

```
  threads = 2L,
  fixedstep = FALSE
)
```

Arguments

event	failure event response variable of length nobs, where nobs denotes the number of observations. It should be a vector containing 0 or 1.
z	input covariate matrix, with nobs rows and nvars columns; each row is an observation.
time	observed event times, which should be a vector with non-negative values.
strata	a vector of indicators for stratification. Default = NULL (i.e. no stratification group in the data), an unstratified model is implemented.
penalty	a character string specifying the spline term for the penalized Newton method. This term is added to the log-partial likelihood, and the penalized log-partial likelihood serves as the new objective function to control the smoothness of the time-varying coefficients. Default is P-spline. Three options are P-spline, Smooth-spline and NULL. If NULL, the method will be the same as coxtv (unpenalized time-varying effects models) and lambda (defined below) will be set as 0. P-spline stands for Penalized B-spline. It combines the B-spline basis with a discrete quadratic penalty on the difference of basis coefficients between adjacent knots. When lambda goes to infinity, the time-varying effects are reduced to be constant. Smooth-spline refers to the Smoothing-spline, the derivative-based penalties combined with B-splines. See degree for different choices. When degree=3, we use the cubic B-spline penalizing the second-order derivative, which reduces the time-varying effect to a linear term when lambda goes to infinity. When degree=2, we use the quadratic B-spline penalizing first-order derivative, which reduces the time-varying effect to a constant when lambda goes to infinity. See Wood (2017) for details. If P-spline or Smooth-spline, then lambda is initialized as a sequence (0.1, 1, 10). Users can modify lambda. See details in lambda.
nsplines	number of basis functions in the splines to span the time-varying effects. The default value is 8. We use the R function splines::bs to generate the B-splines.
lambda	a user-specified lambda sequence as the penalization coefficients in front of the spline term specified by penalty. This is the tuning parameter for penalization. The function IC can be used to select the best tuning parameter based on the information criteria. Alternatively, cross-validation can be used via the cv.coxtp function. When lambda is 0, Newton method without penalization is fitted.
degree	degree of the piecewise polynomial for generating the B-spline basis functions—default is 3 for cubic splines. degree = 2 results in the quadratic B-spline basis functions. If the penalty is P-spline or NULL, degree's default value is 3. If the penalty is Smooth-spline, degree's default value is 2.

knots — the internal knot locations (breakpoints) that define the B-splines. The number of the internal knots should be nsplines-degree-1. If NULL, the locations of knots are chosen as quantiles of distinct failure time points. This choice leads to more stable results in most cases. Users can specify the internal knot locations by themselves.

ties — a character string specifying the method for tie handling. If there are no tied events, the methods are equivalent. By default "Breslow" uses the Breslow approximation, which can be faster when many ties are present. If ties = "none", no approximation will be used to handle ties.

tol — tolerance used for stopping the algorithm. See details in stop below. The default value is 1e-6.

iter.max — maximum iteration number if the stopping criterion specified by stop is not satisfied. The default value is 20.

method — a character string specifying whether to use Newton method or proximal Newton method. If Newton then Hessian is used, while the default method "ProxN" implements the proximal Newton which can be faster and more stable when there exists ill-conditioned second-order information of the log-partial likelihood. See details in Wu et al. (2022).

gamma — parameter for proximal Newton method "ProxN". The default value is 1e8.

btr — a character string specifying the backtracking line-search approach. "dynamic" is a typical way to perform backtracking line-search. See details in Convex Optimization by Boyd and Vandenberghe (2004). "static" limits Newton's increment and can achieve more stable results in some extreme cases, such as ill-conditioned second-order information of the log-partial likelihood, which usually occurs when some predictors are categorical with low frequency for some categories. Users should be careful with static, as this may lead to underfitting.

tau — a positive scalar used to control the step size inside the backtracking line-search. The default value is 0.5.

stop — a character string specifying the stopping rule to determine convergence. "incre" means we stop the algorithm when Newton's increment is less than the tol. See details in Convex Optimization (Chapter 10) by Boyd and Vandenberghe (2004). "relch" means we stop the algorithm when the $(loglik(m) - loglik(m-1))/(loglik(m))$ is less than the tol, where $loglik(m)$ denotes the log-partial likelihood at iteration step m. "ratch" means we stop the algorithm when $(loglik(m) - loglik(m-1))/(loglik(m) - loglik(0))$ is less than the tol. "all" means we stop the algorithm when all the stopping rules ("incre", "relch", "ratch") are met. The default value is ratch. If iter.max is achieved, it overrides any stop rule for algorithm termination.

parallel — if TRUE, then the parallel computation is enabled. The number of threads in use is determined by threads.

threads — an integer indicating the number of threads to be used for parallel computation. The default value is 2. If parallel is false, then the value of threads has no effect.

fixedstep — if TRUE, the algorithm will be forced to run iter.max steps regardless of the stopping criterion specified.

Details

The sequence of models implied by `lambda.spline` is fit by the (proximal) Newton method. The objective function is

$$loglik - P_\lambda,$$

where P_λ is a penalty matrix for `P-spline` or `Smooth-spline`. The λ is the tuning parameter (See details in `lambda`). Users can define the initial sequence. The function `IC` below provides different information criteria to choose the tuning parameter λ. Another function `cv.coxtp` uses the cross-validation to choose the tuning parameter.

Value

A list of objects with S3 class `"coxtp"`. The length is the same as that of `lambda`; each represents the model output with each value of the tuning parameter `lambda`.

`call`	the call that produced this object.
`beta`	the estimated time-varying coefficient for each predictor at each unique time. It is a matrix of dimension `len_unique_t` by `nvars`, where `len_unique_t` is the length of unique observed event `times`.
`bases`	the basis matrix used in model fitting. If `ties="none"`, the dimension of the basis matrix is `nvars` by `nsplines`; if `ties="Breslow"`, the dimension is `len_unique_t` by `nsplines`. The matrix is constructed using the `bs::splines` function.
`ctrl.pts`	estimated coefficient of the basis matrix of dimension `nvars` by `nsplines`. Each row represents a covariate's coefficient on the `nsplines`-dimensional basis functions.
`Hessian`	the Hessian matrix of the log-partial likelihood, of which the dimension is `nsplines * nvars` by `nsplines * nvars`.
`internal.knots`	the internal knot locations (breakpoints) that define the B-splines.
`nobs`	number of observations.
`penalty`	the spline term `penalty` specified by user.
`theta.list`	the history of `ctrl.pts` of length `m` (the length of algorithm iterations), including `ctrl.pts` for each algorithm iteration.
`VarianceMatrix`	the variance matrix of the estimated coefficients of the basis matrix, which is the inverse of the negative Hessian matrix.

References

Boyd, S., and Vandenberghe, L. (2004) Convex optimization. *Cambridge University Press*.

Gray, R. J. (1992) Flexible methods for analyzing survival data using splines, with applications to breast cancer prognosis. *Journal of the American Statistical Association*, **87(420)**: 942-951.

Gray, R. J. (1994) Spline-based tests in survival analysis. *Biometrics*, **50(3)**: 640-652.

Luo, L., He, K., Wu, W., and Taylor, J. M. (2023) Using information criteria to select smoothing parameters when analyzing survival data with time-varying coefficient hazard models. *Statistical*

Methods in Medical Research, **in press**.

Perperoglou, A., le Cessie, S., and 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.

Wu, W., Taylor, J. M., Brouwer, A. F., Luo, L., Kang, J., Jiang, H., and He, K. (2022) Scalable proximal methods for cause-specific hazard modeling with time-varying coefficients. *Lifetime Data Analysis*, **28(2)**: 194-218.

Wood, S. N. (2017) P-splines with derivative based penalties and tensor product smoothing of unevenly distributed data. *Statistics and Computing*, **27(4)**: 985-989.

See Also

IC, cv.coxtp plot, get.tvcoef and baseline.

Examples

```
data(ExampleData)
z <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event

lambda  = c(0,1)
fit   <- coxtp(event = event, z = z, time = time, lambda=lambda)
```

---

coxtv *fit a Cox non-proportional hazards model*

Description

Fit a Cox non-proportional hazards model via maximum likelihood.

Usage

```
coxtv(
  event,
  z,
  time,
  strata = NULL,
  nsplines = 8,
  knots = NULL,
```

```
  degree = 3,
  ties = "Breslow",
  stop = "ratch",
  tol = 1e-06,
  iter.max = 20,
  method = "ProxN",
  gamma = 1e+08,
  btr = "dynamic",
  tau = 0.5,
  parallel = FALSE,
  threads = 2L,
  fixedstep = FALSE
)
```

Arguments

event	failure event response variable of length nobs, where nobs denotes the number of observations. It should be a vector containing 0 or 1.
z	input covariate matrix, with nobs rows and nvars columns; each row is an observation.
time	observed event times, which should be a vector with non-negative values.
strata	a vector of indicators for stratification. Default = NULL (i.e. no stratification group in the data), an unstratified model is implemented.
nsplines	number of basis functions in the splines to span the time-varying effects. The default value is 8. We use the R function splines::bs to generate the B-splines.
knots	the internal knot locations (breakpoints) that define the B-splines. The number of the internal knots should be nsplines-degree-1. If NULL, the locations of knots are chosen as quantiles of distinct failure time points. This choice leads to more stable results in most cases. Users can specify the internal knot locations by themselves.
degree	degree of the piecewise polynomial for generating the B-spline basis functions—default is 3 for cubic splines. degree = 2 results in the quadratic B-spline basis functions.
ties	a character string specifying the method for tie handling. If there are no tied events, the methods are equivalent. By default "Breslow" uses the Breslow approximation, which can be faster when many ties are present. If ties = "none", no approximation will be used to handle ties.
stop	a character string specifying the stopping rule to determine convergence. "incre" means we stop the algorithm when Newton's increment is less than the tol. See details in Convex Optimization (Chapter 10) by Boyd and Vandenberghe (2004). "relch" means we stop the algorithm when the $(loglik(m) - loglik(m-1))/(loglik(m))$ is less than the tol, where $loglik(m)$ denotes the log-partial likelihood at iteration step m. "ratch" means we stop the algorithm when $(loglik(m) - loglik(m-1))/(loglik(m) - loglik(0))$ is less than the tol. "all" means we stop the algorithm when all the stopping rules ("incre", "relch", "ratch") are met. The default value is ratch. If iter.max is achieved, it overrides any stop rule for algorithm termination.

tol	tolerance used for stopping the algorithm. See details in stop below. The default value is 1e-6.
iter.max	maximum iteration number if the stopping criterion specified by stop is not satisfied. The default value is 20.
method	a character string specifying whether to use Newton method or proximal Newton method. If "Newton" then Hessian is used, while the default method "ProxN" implements the proximal Newton which can be faster and more stable when there exists ill-conditioned second-order information of the log-partial likelihood. See details in Wu et al. (2022).
gamma	parameter for proximal Newton method "ProxN". The default value is 1e8.
btr	a character string specifying the backtracking line-search approach. "dynamic" is a typical way to perform backtracking line-search. See details in Convex Optimization by Boyd and Vandenberghe (2004). "static" limits Newton's increment and can achieve more stable results in some extreme cases, such as ill-conditioned second-order information of the log-partial likelihood, which usually occurs when some predictors are categorical with low frequency for some categories. Users should be careful with static, as this may lead to underfitting.
tau	a positive scalar used to control the step size inside the backtracking line-search. The default value is 0.5.
parallel	if TRUE, then the parallel computation is enabled. The number of threads in use is determined by threads.
threads	an integer indicating the number of threads to be used for parallel computation. The default value is 2. If parallel is false, then the value of threads has no effect.
fixedstep	if TRUE, the algorithm will be forced to run iter.max steps regardless of the stopping criterion specified.

Details

The model is fit by Newton method (proximal Newton method).

Value

An object with S3 class coxtv.

call	the call that produced this object.
beta	the estimated time-varying coefficient for each predictor at each unique time. It is a matrix of dimension len_unique_t by nvars, where len_unique_t is the length of unique observed event times.
bases	the basis matrix used in model fitting. If ties="none", the dimension of the basis matrix is nvars by nsplines; if ties="Breslow", the dimension is len_unique_t by nsplines. The matrix is constructed using the bs::splines function.
ctrl.pts	estimated coefficient of the basis matrix of dimension nvars by nsplines. Each row represents a covariate's coefficient on the nsplines-dimensional basis functions.

`Hessian`	the Hessian matrix of the log-partial likelihood, of which the dimension is `nsplines * nvars` by `nsplines * nvars`.
`internal.knots`	the internal knot locations (breakpoints) that define the B-splines.
`nobs`	number of observations.
`theta.list`	the history of `ctrl.pts` of length `m` (the length of algorithm iterations), including `ctrl.pts` for each algorithm iteration.
`VarianceMatrix`	the variance matrix of the estimated coefficients of the basis matrix, which is the inverse of the negative Hessian matrix.

References

Boyd, S., and Vandenberghe, L. (2004) Convex optimization. *Cambridge University Press*.

Gray, R. J. (1992) Flexible methods for analyzing survival data using splines, with applications to breast cancer prognosis. *Journal of the American Statistical Association*, **87(420)**: 942-951.

Gray, R. J. (1994) Spline-based tests in survival analysis. *Biometrics*, **50(3)**: 640-652.

Luo, L., He, K., Wu, W., and Taylor, J. M. (2023) Using information criteria to select smoothing parameters when analyzing survival data with time-varying coefficient hazard models.

Perperoglou, A., le Cessie, S., and 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.

Wu, W., Taylor, J. M., Brouwer, A. F., Luo, L., Kang, J., Jiang, H., and He, K. (2022) Scalable proximal methods for cause-specific hazard modeling with time-varying coefficients. *Lifetime Data Analysis*, **28(2)**: 194-218.

See Also

`coef`, `plot`, and the `coxtp` function.

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtv(event = event, z = z, time = time)
```

`cv.coxtp` *fit a cross-validated Cox non-proportional hazards model with P-spline or Smoothing-spline where penalization tuning parameter is provided by cross-validation*

Description

Fit a Cox non-proportional hazards model via penalized maximum likelihood. The penalization tuning parameter is provided by cross-validation.

Usage

```
cv.coxtp(
  event,
  z,
  time,
  strata = NULL,
  lambda = c(0.1, 1, 10),
  nfolds = 5,
  foldid = NULL,
  knots = NULL,
  penalty = "Smooth-spline",
  nsplines = 8,
  ties = "Breslow",
  tol = 1e-06,
  iter.max = 20L,
  method = "ProxN",
  gamma = 1e+08,
  btr = "dynamic",
  tau = 0.5,
  stop = "ratch",
  parallel = FALSE,
  threads = 1L,
  degree = 3L,
  fixedstep = FALSE
)
```

Arguments

`event`	failure event response variable of length `nobs`, where `nobs` denotes the number of observations. It should be a vector containing 0 or 1.
`z`	input covariate matrix, with `nobs` rows and `nvars` columns; each row is an observation.
`time`	observed event times, which should be a vector with non-negative values.
`strata`	a vector of indicators for stratification. Default = `NULL` (i.e. no stratification group in the data), an unstratified model is implemented.

`lambda` a user specified sequence as the penalization coefficients in front of the spline term specified by `penalty`. This is the tuning parameter for penalization. The function `IC` can be used to select the best tuning parameter based on the information criteria. Users can specify larger values when the absolute values of the estimated time-varying effects are too large. When `lambda` is `0`, Newton method without penalization is fitted.

`nfolds` number of folds for cross-validation, the default value is 5. The smallest value allowable is `nfolds=3`.

`foldid` an optional vector of values between 1 and `nfolds` identifying what fold each observation is in. If supplied, `nfolds` can be missing.

`knots` the internal knot locations (breakpoints) that define the B-splines. The number of the internal knots should be `nsplines-degree-1`. If `NULL`, the locations of knots are chosen as quantiles of distinct failure time points. This choice leads to more stable results in most cases. Users can specify the internal knot locations by themselves.

`penalty` a character string specifying the spline term for the penalized Newton method. This term is added to the log-partial likelihood, and the penalized log-partial likelihood serves as the new objective function to control the smoothness of the time-varying coefficients. Default is `P-spline`. Three options are `P-spline`, `Smooth-spline` and `NULL`. If `NULL`, the method will be the same as `coxtv` (unpenalized time-varying effects models) and `lambda` (defined below) will be set as 0.

`P-spline` stands for Penalized B-spline. It combines the B-spline basis with a discrete quadratic penalty on the difference of basis coefficients between adjacent knots. When `lambda` goes to infinity, the time-varying effects are reduced to be constant.

`Smooth-spline` refers to the Smoothing-spline, the derivative-based penalties combined with B-splines. See `degree` for different choices. When `degree=3`, we use the cubic B-spline penalizing the second-order derivative, which reduces the time-varying effect to a linear term when `lambda` goes to infinity. When `degree=2`, we use the quadratic B-spline penalizing first-order derivative, which reduces the time-varying effect to a constant when `lambda` goes to infinity. See Wood (2017) for details.

If `P-spline` or `Smooth-spline`, then `lambda` is initialized as a sequence (0.1, 1, 10). Users can modify `lambda`. See details in `lambda`.

`nsplines` number of basis functions in the splines to span the time-varying effects. The default value is 8. We use the R function `splines::bs` to generate the B-splines.

`ties` a character string specifying the method for tie handling. If there are no tied events, the methods are equivalent. By default `"Breslow"` uses the Breslow approximation, which can be faster when many ties are present. If `ties = "none"`, no approximation will be used to handle ties.

`tol` tolerance used for stopping the algorithm. See details in `stop` below. The default value is `1e-6`.

`iter.max` maximum iteration number if the stopping criterion specified by `stop` is not satisfied. The default value is `20`.

`method` a character string specifying whether to use Newton method or proximal Newton method. If `"Newton"` then Hessian is used, while the default method `"ProxN"` implements the proximal Newton which can be faster and more stable when there exists ill-conditioned second-order information of the log-partial likelihood. See details in Wu et al. (2022).

`gamma` parameter for proximal Newton method `"ProxN"`. The default value is `1e8`.

`btr` a character string specifying the backtracking line-search approach. `"dynamic"` is a typical way to perform backtracking line-search. See details in Convex Optimization by Boyd and Vandenberghe (2004). `"static"` limits Newton's increment and can achieve more stable results in some extreme cases, such as ill-conditioned second-order information of the log-partial likelihood, which usually occurs when some predictors are categorical with low frequency for some categories. Users should be careful with `static`, as this may lead to underfitting.

`tau` a positive scalar used to control the step size inside the backtracking line-search. The default value is `0.5`.

`stop` a character string specifying the stopping rule to determine convergence. `"incre"` means we stop the algorithm when Newton's increment is less than the `tol`. See details in Convex Optimization (Chapter 10) by Boyd and Vandenberghe (2004). `"relch"` means we stop the algorithm when the $(loglik(m) - loglik(m-1))/(loglik(m))$ is less than the `tol`, where $loglik(m)$ denotes the log-partial likelihood at iteration step m. `"ratch"` means we stop the algorithm when $(loglik(m) - loglik(m-1))/(loglik(m) - loglik(0))$ is less than the `tol`. `"all"` means we stop the algorithm when all the stopping rules (`"incre"`, `"relch"`, `"ratch"`) are met. The default value is `ratch`. If `iter.max` is achieved, it overrides any stop rule for algorithm termination.

`parallel` if `TRUE`, then the parallel computation is enabled. The number of threads in use is determined by `threads`.

`threads` an integer indicating the number of threads to be used for parallel computation. The default value is `2`. If `parallel` is false, then the value of `threads` has no effect.

`degree` degree of the piecewise polynomial for generating the B-spline basis functions—default is 3 for cubic splines. `degree = 2` results in the quadratic B-spline basis functions.

If `penalty` is `P-spline` or `NULL`, `degree`'s default value is 3.

If `penalty` is `Smooth-spline`, `degree`'s default value is 2.

`fixedstep` if `TRUE`, the algorithm will be forced to run `iter.max` steps regardless of the stopping criterion specified.

Details

The function runs `coxtp` length of `lambda` by `nfolds` times; each is to compute the fit with each of the folds omitted.

Value

An object of class "cv.coxtp" is returned, which is a list with the ingredients of the cross-validation fit.

model.cv	a "coxtp" object with tuning parameter chosen based on cross-validation.
lambda	the values of lambda used in the fits.
cve	the mean cross-validated error - a vector having the same length as lambda. For the k-th testing fold (k = 1, ..., nfolds), we take the remaining folds as the training folds. Based on the model trained on the training folds, we calculate the log-partial likelihood on all the folds $loglik0$ and training folds $loglik1$. The cve is equal to $-2 * (loglik0 - loglik1)$. See details in Verweij (1993). This approach avoids the construction of a partial likelihood on the test set so that the risk set is always sufficiently large.
lambda.min	the value of lambda that gives minimum cve.

References

Boyd, S., and Vandenberghe, L. (2004) Convex optimization. *Cambridge University Press*.

Gray, R. J. (1992) Flexible methods for analyzing survival data using splines, with applications to breast cancer prognosis. *Journal of the American Statistical Association*, **87(420)**: 942-951.

Gray, R. J. (1994) Spline-based tests in survival analysis. *Biometrics*, **50(3)**: 640-652.

Luo, L., He, K., Wu, W., and Taylor, J. M. (2023) Using information criteria to select smoothing parameters when analyzing survival data with time-varying coefficient hazard models. *Statistical Methods in Medical Research*, **in press**.

Perperoglou, A., le Cessie, S., and 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.

Verweij, P. J., and Van Houwelingen, H. C. (1993) Cross-validation in survival analysis. *Statistics in Medicine*, **12(24)**: 2305-2314.

Wu, W., Taylor, J. M., Brouwer, A. F., Luo, L., Kang, J., Jiang, H., and He, K. (2022) Scalable proximal methods for cause-specific hazard modeling with time-varying coefficients. *Lifetime Data Analysis*, **28(2)**: 194-218.

Wood, S. N. (2017) P-splines with derivative based penalties and tensor product smoothing of unevenly distributed data. *Statistics and Computing*, **27(4)**: 985-989.

Examples

```
data(ExampleData)
z <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event
lambda  = c(0.1, 1)
fit  <- cv.coxtp(event = event, z = z, time = time, lambda=lambda, nfolds = 5)
```

ExampleData *example data with 2000 observations of 2 continuous variables*

Description

A simulated data set containing 2 continuous variables.

Usage

```
data(ExampleData)
```

Format

A list containing the following elements:

z simulated continuous covariates V1 and V2, with a time-independent coefficient $\beta_1(t) = 1$ and a time-varying coefficient $\beta_2(t) = sin(3\pi t/4)$.

event simulated failure event response; binary variable with 0 or 1.

time simulated observed event times; continuous variable with non-negative values.

ExampleDataBinary *example data with 2000 observations of 2 binary variables*

Description

A simulated data set containing 2 binary variables.

Usage

```
data(ExampleDataBinary)
```

Format

A list containing the following elements:

z simulated binary covariates V1 and V2, with a time-independent coefficient $\beta_1(t) = 1$ and a time-varying coefficient $\beta_2(t) = exp(-1.5t)$.

event simulated failure event response; binary variable with 0 or 1.

time simulated observed event times; continuous variable with non-negative values.

get.tvcoef *helper function to get time-varying coefficients*

Description

The function gives the time-varying coefficients based on a fitted coxtv or coxtp subject. Users can specify the time points to calculate the time-varying coefficients.

Usage

```
get.tvcoef(fit, time)
```

Arguments

fit	model from coxtv or coxtp.
time	time points to calculate the time-varying coefficients. If NULL, the observed event times for fitting the model will be used.

Value

A matrix of the time-varying coefficients. The dimension is the length of time by nvars, where nvars is the number of covariates in the fitted mode. Each row represents the time-varying coefficients at the corresponding time.

Examples

```
z     <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event
fit   <- coxtv(event = event, z = z, time = time, degree = 2)
coef  <- get.tvcoef(fit)
```

IC *calculating information criteria from a* coxtp *object*

Description

This function is to calculate information criteria from a coxtp object to select the penalization tuning parameter.

Usage

```
IC(fit, IC.prox)
```

Arguments

fit	model from coxtp.
IC.prox	when calculating information criteria, there might be numerical issues (e.g. the Hessian matrix is close to be singular). In such cases, warnings will be given. If IC.prox = TRUE, we modify the diagonal of the Hessian matrix following the same approach as the proximal method detailed in Wu et al. (2022), which can lead to more stable estimates. The default value is FALSE.

Details

In order to select the proper smoothing parameter, we utilize the idea of information criteria. We provide four different information criteria to select the optimal smoothing parameter λ. Generally, mAIC, TIC and GIC select similar parameters and the difference of resulting estimates are barely noticeable. See details in the Luo et al. (2023).

Value

model.mAIC	an object with S3 class "coxtp" using mAIC to select the tuning parameter.
model.TIC	an object with S3 class "coxtp" using TIC to select the tuning parameter.
model.GIC	an object with S3 class "coxtp" using GIC to select the tuning parameter.
mAIC	a sequence of mAIC values corresponding to each of the tuning parameter lambda from "coxtp".
TIC	a sequence of TIC values corresponding to each of the tuning parameter lambda from "coxtp".
GIC	a sequence of GIC values corresponding to each of the tuning parameter lambda from "coxtp".

References

Akaike, H. (1998) Information theory and an extension of the maximum likelihood principle. *In Selected Papers of Hirotugu Akaike*. 199–213.

Luo, L., He, K., Wu, W., and Taylor, J. M. (2023) Using information criteria to select smoothing parameters when analyzing survival data with time-varying coefficient hazard models. *Statistical Methods in Medical Research*, **in press**.

Takeuchi, K. (1976) Distribution of information statistics and criteria for adequacy of models. *Mathematical Sciences*, **153**: 12–18.

Wu, W., Taylor, J. M., Brouwer, A. F., Luo, L., Kang, J., Jiang, H., and He, K. (2022) Scalable proximal methods for cause-specific hazard modeling with time-varying coefficients. *Lifetime Data Analysis*, **28(2)**: 194-218.

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtp(event = event, z = z, time = time)
IC  <- IC(fit)
```

`plot.baseline` *plotting the baseline hazard*

Description

Plotting the baseline hazard from a fitted `baseline` object.

Usage

```
## S3 method for class 'baseline'
plot(x, xlab, ylab, xlim, ylim, title, ...)
```

Arguments

`x`	fitted object from `baseline` function.
`xlab`	the title for the x axis.
`ylab`	the title for the y axis.
`xlim`	the limits of the x axis.
`ylim`	the limits of the y axis.
`title`	the title for the plot.
`...`	other graphical parameters to plot

Value

A plot is produced, and nothing is returned.

Examples

```
data(ExampleData)
z <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event

fit   <- coxtv(event = event, z = z, time = time)
base.est <- baseline(fit)
plot(base.est)
```

plot.coxtp *plotting results from a fitted* coxtp *object*

Description

This function creates a plot of the time-varying coefficients from a fitted coxtp model.

Usage

```
## S3 method for class 'coxtp'
plot(
  x,
  parm,
  CI = TRUE,
  level = 0.95,
  exponentiate = FALSE,
  xlab,
  ylab,
  xlim,
  ylim,
  allinone = FALSE,
  title,
  linetype,
  color,
  fill,
  time,
  ...
)
```

Arguments

x	model obtained from coxtp.
parm	covariate names fitted in the model to be plotted. If NULL, all covariates are plotted.
CI	if TRUE, confidence intervals are displayed. The default value is TRUE.
level	the level of confidence intervals. The default value is 0.95.
exponentiate	if TRUE, exponential scale of the fitted coefficients (hazard ratio) for each covariate is plotted. If FALSE, the fitted time-varying coefficients (log hazard ratio) are plotted.
xlab	the title for the x axis.
ylab	the title for the y axis.
xlim	the limits for the x axis.
ylim	the limits for the y axis.
allinone	if TRUE, the time-varying trajectories for different covariates are combined into a single plot. The default value is FALSE.

title	the title for the plot.
linetype	the line type for the plot.
color	the aesthetics parameter for the plot.
fill	the aesthetics parameter for the plot.
time	the time points for which the time-varying coefficients to be plotted. The default value is the unique observed event times in the dataset fitting the time-varying effects model.
...	other graphical parameters to plot

Value

A plot is produced, and nothing is returned.

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtp(event = event, z = z, time = time)
plot(fit$lambda1)
```

---

plot.coxtv *plotting results from a fitted* coxtv *object*

Description

This function creates a plot of the time-varying coefficients from a fitted coxtv model.

Usage

```
## S3 method for class 'coxtv'
plot(
  x,
  parm,
  CI = TRUE,
  level = 0.95,
  exponentiate = FALSE,
  xlab,
  ylab,
  xlim,
  ylim,
  allinone = FALSE,
  title,
  linetype,
  color,
```

```
  fill,
  time,
  ...
)
```

Arguments

x	model obtained from coxtv.
parm	covariate names fitted in the model to be plotted. If NULL, all covariates are plotted.
CI	if TRUE, confidence intervals are displayed. The default value is TRUE.
level	the level of confidence intervals. The default value is 0.95.
exponentiate	if TRUE, exponential scale of the fitted coefficients (hazard ratio) for each covariate is plotted. If FALSE, the fitted time-varying coefficients (log hazard ratio) are plotted.
xlab	the title for the x axis.
ylab	the title for the y axis.
xlim	the limits for the x axis.
ylim	the limits for the y axis.
allinone	if TRUE, the time-varying trajectories for different covariates are combined into a single plot. The default value is FALSE.
title	the title for the plot.
linetype	the line type for the plot.
color	the aesthetics parameter for the plot.
fill	the aesthetics parameter for the plot.
time	the time points for which the time-varying coefficients to be plotted. The default value is the unique observed event times in the dataset fitting the time-varying effects model.
...	other graphical parameters to plot

Value

A plot is produced, and nothing is returned.

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtv(event = event, z = z, time = time)
plot(fit)
```

StrataExample *example data for stratified model illustration*

Description

A simulated data set containing 2 binary variables from 10 distinct stratums.

Usage

```
data(StrataExample)
```

Format

A list containing the following elements:

z simulated binary covariates V1 and V2, with a time-independent coefficient $\beta_1(t) = 1$ and a time-varying coefficient $\beta_2(t) = sin(3\pi t/4)$.

event simulated failure event response; binary variable with 0 or 1.

time simulated observed event times; continuous variable with non-negative values.

strata simulated strata variable; patients in different stratums have different baseline hazards.

support *Study to Understand Prognoses Preferences Outcomes and Risks of Treatment*

Description

The SUPPORT dataset tracks five response variables: hospital death, severe functional disability, hospital costs, and time until death and death itself. The patients are followed for up to 5.56 years. See Bhatnagar et al. (2020) for details.

Usage

```
data(support)
```

Format

A data frame with 9,104 observations and 34 variables after imputation and the removal of response variables like hospital charges, patient ratio of costs to charges and micro-costs following Bhatnagar et al. (2020). Ordinal variables, namely functional disability and income, were also removed. Finally, Surrogate activities of daily living were removed due to sparsity. There were 6 other model scores in the data-set and they were removed; only aps and sps were kept.

age stores a double representing age.

death death at any time up to NDI (National Death Index) date: 12/31/1994.

sex 0=female, 1=male.

slos days from study entry to discharge.

d.time days of follow-up.

dzgroup each level of dzgroup: ARF/MOSF w/Sepsis, COPD, CHF, Cirrhosis, Coma, Colon Cancer, Lung Cancer, MOSF with malignancy.

dzclass ARF/MOSF, COPD/CHF/Cirrhosis, Coma and cancer disease classes.

num.co the number of comorbidities.

edu years of education of patients.

scoma the SUPPORT coma score based on Glasgow D3.

avtisst average TISS, days 3-25.

race indicates race: White, Black, Asian, Hispanic or other.

hday day in Hospital at Study Admit.

diabetes diabetes (Com27-28, Dx 73).

dementia dementia (Comorbidity 6).

ca cancer state.

meanbp mean arterial blood pressure day 3.

wblc white blood cell count on day 3.

hrt heart rate day 3.

resp respiration rate day 3.

temp temperature, in Celsius, on day 3.

pafi PaO2/(0.01*FiO2) day 3.

alb serum albumin day 3.

bili bilirubin day 3.

crea serum creatinine day 3.

sod serum sodium day 3.

ph serum pH (in arteries) day 3.

glucose serum glucose day 3.

bun bun day 3.

urine urine output day 3.

adlp adl patient day 3.

adlsc imputed adl calibrated to surrogate, if a surrogate was used for a follow up.

sps SUPPORT physiology score.

aps apache III physiology score.

Details

Some of the original data was missing. Before imputation, there were a total of 9,104 individuals and 47 variables. Following Bhatnagar et al. (2020), a few variables were removed. Three response variables were removed: hospital charges, patient ratio of costs to charges and patient micro-costs. Hospital death was also removed as it was directly informative of the event of interest, namely death. Additionally, functional disability and income were removed as they are ordinal covariates. Finally, 8 covariates were removed related to the results of previous findings: SUPPORT day 3 physiology score (sps), APACHE III day 3 physiology score (aps), SUPPORT model 2-month survival estimate, SUPPORT model 6-month survival estimate, Physician's 2-month survival estimate for pt., Physician's 6-month survival estimate for pt., Patient had Do Not Resuscitate (DNR) order, and Day of DNR order (<0 if before study). Of these, sps and aps were added on after imputation, as they were missing only 1 observation. First the imputation is done manually using the normal values for physiological measures recommended by Knaus et al. (1995). Next, a single dataset was imputed using **mice** with default settings. After imputation, the covariate for surrogate activities of daily living was not imputed. This is due to collinearity between the other two covariates for activities of daily living. Therefore, surrogate activities of daily living were removed. See details in the R package (casebase) by Bhatnagar et al. (2020).

Source

Available at the following website: https://biostat.app.vumc.org/wiki/Main/SupportDesc.

References

Bhatnagar, S., Turgeon, M., Islam, J., Hanley, J. A., and Saarela, O. (2020) casebase: Fitting Flexible Smooth-in-Time Hazards and Risk Functions via Logistic and Multinomial Regression. *R package version 0.9.0*, https://CRAN.R-project.org/package=casebase.

Knaus, W. A., Harrell, F. E., Lynn, J., Goldman, L., Phillips, R. S., Connors, A. F., et al. (1995) The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, **122(3)**: 191-203.

Examples

```
data(support)
support <- support[support$ca %in% c("metastatic"),]
time <- support$d.time
death <- support$death
diabetes <-  model.matrix(~factor(support$diabetes))[,-1]
#sex: female as the reference group
sex <- model.matrix(~support$sex)[,-1]
#age: continuous variable
age <-support$age
age[support$age<=50] <- "<50"
age[support$age>50 & support$age<=60] <- "50-59"
age[support$age>60 & support$age<70] <- "60-69"
age[support$age>=70] <- "70+"
age <- factor(age, levels = c("60-69", "<50", "50-59", "70+"))
z_age <- model.matrix(~age)[,-1]
z <- data.frame(z_age, sex, diabetes)
```

```
colnames(z) <- c("age_50", "age_50_59", "age_70", "diabetes", "male")
data <- data.frame(time, death, z)
fit.coxtv <- coxtv(event = death, z = z, time = time)
```

`tvef.ph` *testing the proportional hazards assumption from a* `coxtv` *or* `coxtp` *object*

Description

Testing the proportional hazards assumption using a Wald test statistic.

Usage

```
tvef.ph(fit, parm)
```

Arguments

`fit`	fitted `coxtv` or `coxtp` model.
`parm`	the names of parameters to be tested.

Value

`tvef.ph` produces a matrix. Each row corresponds to a covariate from the fitted object. The three columns give the test statistic, degrees of freedom and P-value.

See Also

`tvef.zero` `tvef.zero.time`

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtv(event = event, z = z, time = time)
tvef.ph(fit)
```

tvef.zero *testing the significance of the covariates from a* coxtv *or* coxtp *object*

Description

Testing the significance of the covariates from a coxtv or coxtp object using a Wald test statistic. The null hypothesis $H_0 : \beta(t) = 0$ for any t, where t denotes the event time.

Usage

```
tvef.zero(fit, parm)
```

Arguments

fit	fitted coxtv or coxtp model.
parm	the names of parameters to be tested.

Value

tvef.zero produces a matrix. Each row corresponds to a covariate from the fitted object. The three columns give the test statistic, degrees of freedom and P-value.

See Also

tvef.ph tvef.zero.time

Examples

```
data(ExampleData)
z <- ExampleData$z
time <- ExampleData$time
event <- ExampleData$event
fit <- coxtv(event = event, z = z, time = time)
tvef.ph(fit)
```

tvef.zero.time *testing the significance of the covariates from a* coxtv *or* coxtp *object using a Wald test statistic*

Description

Testing the significance of the covariates at each time point.

Usage

```
tvef.zero.time(fit, time, parm)
```

Arguments

fit	fitted coxtv or coxtp model.
time	the time points to test if the covariate is significant or not.
parm	the names of parameters to be tested.

Value

tvef.zero.time produces a list of length nvars. Each element of the list is a matrix with respect to a covariate. The matrix is of dimension len_unique_t by 4, where len_unique_t is the length of unique observed event time. Each row corresponds to the testing result at that time. The four columns give the estimations, standard error, test-statistic and P-value.

See Also

tvef.ph tvef.zero

Examples

```
data(ExampleData)
z <- ExampleData$z
time  <- ExampleData$time
event <- ExampleData$event
fit   <- coxtv(event = event, z = z, time = time)
test  <- tvef.zero.time(fit)
```

Index