Package 'coxed'

August 2, 2020

Type Package

Title Duration-Based Quantities of Interest for the Cox Proportional Hazards Model

Version 0.3.3

Depends R (>= 3.5.0), rms, survival, mgcv

Description Functions for generating, simulating, and visualizing expected durations and marginal changes in duration from the Cox proportional hazards model as described in Kropko and Harden (2017) <doi:10.1017/S000712341700045X> and Harden and Kropko (2018) <doi:10.1017/ps

License GPL-2

Encoding UTF-8

URL https://github.com/jkropko/coxed

LazyData true

Imports PermAlgo, dplyr, tidyr, ggplot2, gridExtra, mediation, utils

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, bindrcpp

VignetteBuilder knitr

NeedsCompilation no

Author Kropko, Jonathan [aut, cre], Harden, Jeffrey J. [aut]

Maintainer ``Kropko, Jonathan'' <jkropko@virginia.edu>

Repository CRAN

Date/Publication 2020-08-02 01:20:07 UTC

R topics documented:

`coxed-package` *Duration-Based Quantities of Interest and Simulation Methods for the Cox Proportional Hazards Model*

Description

The Cox proportional hazards model (implemented in R with `coxph` in the `survival` package or with `cph` in the `rms` package) is one of the most frequently used estimators in duration (survival) analysis. Because it is estimated using only the observed durations' rank ordering, typical quantities of interest used to communicate results of the Cox model come from the hazard function (e.g., hazard ratios or percentage changes in the hazard rate). These quantities are substantively vague and difficult for many audiences of research to understand. The `coxed` package introduces a suite of methods to address these problems. The package allows researchers to calculate duration-based quantities from Cox model results, such as the expected duration (or survival time) given covariate values and marginal changes in duration for a specified change in a covariate. These duration-based quantities often match better with researchers' substantive interests and are easily understood by most readers. In addition, no standard method exists for simulating durations directly from the Cox model's data generating process because it does not assume a distributional form for the baseline hazard function. The `coxed` package also contains functions to simulate general duration data that does not rely on an assumption of any particular parametric hazard function.

Duration-Based Quantities of Interest for the Cox Model

The `coxed` function generates expected durations for individual observations and/or marginal changes in expected duration given a change in a covariate from the Cox proportional hazards model. Specifically, the methods can compute (1) the expected duration for each observation used to fit the Cox

model, given the covariates, (2) the expected duration for a "new" observation with a covariate profile set by the analyst, or (3) the first difference, or change, in expected duration given two new data frames.

There are two different methods of generating duration-based quantities in the package. coxed with type="npsf" calculates expected durations by using the method proposed by Cox and Oakes (1984, 107-109) for estimating the cumulative baseline hazard function. This method is nonparametric and results in a step-function representation of the cumulative baseline hazard. Cox and Oakes (1984, 108) show that the cumulative baseline hazard function can be estimated after fitting a Cox model by

$$\hat{H}_0(t) = \sum_{\tau_j < t} \frac{d_j}{\sum_{l \in \Re(\tau_j)} \hat{\psi}(l)},$$

where τ_j represents time points earlier than t, d_j is a count of the total number of failures at τ_j, $\Re(\tau_j)$ is the remaining risk set at τ_j, and $\hat{\psi}(l)$ represents the ELP from the Cox model for observations still in the risk set at τ_j. This equation is used calculate the cumulative baseline hazard at all time points in the range of observed durations. This estimate is a stepwise function because time points with no failures do not contribute to the cumulative hazard, so the function is flat until the next time point with observed failures.

We extend this method to obtain expected durations by first calculating the baseline survivor function from the cumulative hazard function, using

$$\hat{S}_0(t) = \exp[-\hat{H}_0(t)].$$

Each observation's survivor function is related to the baseline survivor function by

$$\hat{S}_i(t) = \hat{S}_0(t)^{\hat{\psi}(i)},$$

where $\hat{\psi}(i)$ is the exponentiated linear predictor (ELP) for observation i. These survivor functions can be used directly to calculate expected durations for each observation. The expected value of a non-negative random variable can be calculated by

$$E(X) = \int_0^\infty \Big(1 - F(t)\Big) dt,$$

where $F(.)$ is the cumulative distribution function for X. In the case of a duration variable t_i, the expected duration is

$$E(t_i) = \int_0^T S_i(t)\, dt,$$

where T is the largest possible duration and $S(t)$ is the individual's survivor function. We approximate this integral with a right Riemann-sum by calculating the survivor functions at every discrete time point from the minimum to the maximum observed durations, and multiplying these values by the length of the interval between time points with observed failures:

$$E(t_i) \approx \sum_{t_j \in [0,T]} (t_j - t_{j-1}) S_i(t_j).$$

coxed with type="gam" employs a generalized additive model (GAM) to map the model's estimated linear predictor values to duration times and proceeds according to five steps. First, it uses coefficient estimates from the Cox model, so researchers must first estimate the model just as they

always have. Then the method computes expected values of risk for each observation by matrix-multiplying the covariates by the estimated coefficients from the model, then exponentiating the result. This creates the exponentiated linear predictor (ELP). Then the observations are ranked from smallest to largest according to their values of the ELP. This ranking is interpreted as the expected order of failure; the larger the value of the ELP, the sooner the model expects that observation to fail, relative to the other observations. The next step is to connect the model's expected risk for each observation (ELP) to duration time (the observed durations). A `gam` fits a model to data by using a series of locally-estimated polynomial splines set by the user (see, for example, Wood, Pya, and Saefken 2016). It is a flexible means of allowing for the possibility of nonlinear relationships between variables. `coxed` with `type="gam"` uses a GAM to model the observed durations as a function of the linear predictor ranks generated in the previous step. More specifically, the method utilizes a cubic regression spline to draw a smoothed line summarizing the bivariate relationship between the observed durations and the ranks. The GAM fit can be used directly to compute expected durations, given the covariates, for each observation in the data.

See Kropko and Harden (2018) for further details about generating expected durations and marginal changes in expected duration from the Cox model. The `coxed` function can also generate these quantities from data with time-varying covariates (see `coxed.npsf.tvc` and `coxed.gam.tvc`).

Simulating duration data for the Cox model

The `sim.survdata` function generates simulated duration data. It can accept a user-supplied hazard function, or else it uses the flexible-hazard method described in Harden and Kropko (2018) to generate a hazard that does not necessarily conform to any parametric hazard function. It can generate data with time-varying covariates or coefficients. For time-varying covariates `type="tvc"` it employs the permutational algorithm by Sylvestre and Abrahamowicz (2008). For time-varying coefficients with `type="tvbeta"`, the first beta coefficient that is either supplied by the user or generated by the function is multiplied by the natural log of the failure time under consideration.

The flexible-hazard method employed when `hazard.fun` is `NULL` generates a unique baseline hazard by fitting a curve to randomly-drawn points. This produces a wide variety of shapes for the baseline hazard, including those that are unimodal, multimodal, monotonically increasing or decreasing, and many other shapes. The method then generates a density function based on each baseline hazard and draws durations from it in a way that circumvents the need to calculate the inverse cumulative baseline hazard. Because the shape of the baseline hazard can vary considerably, this approach matches the Cox model's inherent flexibility and better corresponds to the assumed data generating process (DGP) of the Cox model. Moreover, repeating this process over many iterations in a simulation produces simulated samples of data that better reflect the considerable heterogeneity in data used by applied researchers. This increases the generalizability of the simulation results. See Harden and Kropko (2018) for more detail.

When generating a marginal effect, first the user specifies a covariate by typing its column number in the `X` matrix into the `covariate` argument, then specifies the high and low values at which to fix this covariate. The function calculates the differences in expected duration for each observation when fixing the covariate to the high and low values. If `compare` is `median`, the function reports the median of these differences, and if `compare` is `mean`, the function reports the median of these differences, but any function may be employed that takes a vector as input and outputs a scalar.

If `censor.cond` is `FALSE` then a proportion of the observations specified by `censor` is randomly and uniformly selected to be right-censored. If `censor.cond` is `TRUE` then censoring depends on the covariates as follows: new coefficients are drawn from normal distributions with mean 0 and

standard deviation of 0.1, and these new coefficients are used to create a new linear predictor using the X matrix. The observations with the largest (100 x censor) percent of the linear predictors are designated as right-censored.

Finally, link[coxed]{survsim.plot} is useful for visualizing the baseline functions, including hazard, that result from link[coxed]{sim.survdata} for a particular draw of simulated duration data. The function uses ggplot to create line plots for the baseline failure PDF, the baseline failure CDF, the baseline survivor function, and the baseline hazard function over time. The baseline functions and time are attributes of the sim.survdata output.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Harden, J. J. and Kropko, J. (2018) Simulating Duration Data for the Cox Model. *Political Science Research and Methods* https://doi.org/10.1017/psrm.2018.19

Kropko, J. and Harden, J. J. (2018) Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

Box-Steffensmeier, J. M. (1996) A Dynamic Analysis of The Role of War Chests in Campaign Strategy. *American Journal of Political Science* **40** 352-371

Hyman, J. M. (1983) Accurate monotonicity preserving cubic interpolation. *SIAM J. Sci. Stat. Comput.* **4**, 645–654. https://doi.org/10.1137/0904045

Martin, L. W and Vanberg, G. (2003) Wasting Time? The Impact of Ideology and Size on Delay in Coalition Formation. *British Journal of Political Science* **33** 323-344 https://doi.org/10.1017/S0007123403000140

Sylvestre M.-P., Abrahamowicz M. (2008) Comparison of algorithms to generate event times conditional on time-dependent covariates. *Statistics in Medicine* **27(14)**:2618–34. https://doi.org/10.1002/sim.3092

Wood, S.N., N. Pya and B. Saefken (2016) Smoothing parameter and model selection for general smooth models (with discussion). *Journal of the American Statistical Association* **111**, 1548-1575 http://dx.doi.org/10.1080/01621459.2016.1180986

Examples

```
## See the examples for coxed, sim.survdata, and survsim.plot
```

---

baseline.build *Generate simulated baseline hazard, cumulative hazard, survival, failure PDF, and failure CDF functions*

Description

This function is called by sim.survdata and is not intended to be used by itself.

Usage

```
baseline.build(T = 100, knots = 8, spline = TRUE)
```

Arguments

T	The latest time point during which an observation may fail. Failures can occur as early as 1 and as late as T
knots	The number of points to draw while using the flexible-hazard method to generate hazard functions (default is 8). Ignored if hazard.fun is not NULL.
spline	If TRUE (the default), a spline is employed to smooth the generated cumulative baseline hazard, and if FALSE the cumulative baseline hazard is specified as a step function with steps at the knots. Ignored if hazard.fun is not NULL

Details

This function employs the flexible hazard method described in Harden and Kropko (2018) to generate a baseline failure CDF: it plots points at (0, 0) and (T+1, 1), and it plots knots additional points with x-coordinates drawn uniformly from integers in [2, T] and y-coordinates drawn from U[0, 1]. It sorts these coordinates in ascending order (because a CDF must be non-decreasing) and if spline=TRUE it fits a spline using Hyman's (1983) cubic smoothing function to preserve the CDF's monotonicity. Next it constructs the failure-time PDF by computing the first differences of the CDF at each time point. It generates the survivor function by subtracting the failure CDF from 1. Finally, it computes the baseline hazard by dividing the failure PDF by the survivor function.

Value

A data frame containing every potential failure time and the baseline failure PDF, baseline failure CDF, baseline survivor function, and baseline hazard function at each time point.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Harden, J. J. and Kropko, J. (2018). Simulating Duration Data for the Cox Model. *Political Science Research and Methods* https://doi.org/10.1017/psrm.2018.19

Hyman, J. M. (1983) Accurate monotonicity preserving cubic interpolation. *SIAM J. Sci. Stat. Comput.* **4**, 645–654.

See Also

splinefun, sim.survdata

Examples

```
baseline.functions <- baseline.build(T=100, knots=8, spline=TRUE)
```

`baseline.plot` *Plot the simulated baseline functions*

Description

This function is called by `survsim.plot` and is not intended to be used by itself.

Usage

```
baseline.plot(baseline)
```

Arguments

`baseline`	A data frame containing five variables: time, and the values of the baseline failure PDF, baseline failure CDF, baseline survivor function, and baseline hazard function at each time point. Generally, this data frame is taken from the `baseline` attribute of the `sim.survdata` function

Details

This function reshapes the data for easy faceting with `facet_wrap` within a call to `ggplot`. Each function is plotted on the y-axis and time is plotted on the x-axis using `geom_line`

Value

A figure of class `"gg"` and `"ggplot"`

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

`survsim.plot`, `sim.survdata`

Examples

```
simdata <- sim.survdata(N=1000, T=100, num.data.frames=1)
baseline.plot(simdata$baseline)
```

bca *Bias-corrected and accelerated confidence intervals*

Description

This function uses the method proposed by DiCiccio and Efron (1996) to generate confidence intervals that produce more accurate coverage rates when the distribution of bootstrap draws is non-normal. This code is adapted from the `BC.CI()` function within the `mediate` function in the `mediation` package.

Usage

```
bca(theta, conf.level = 0.95)
```

Arguments

`theta`	a vector that contains draws of a quantity of interest using bootstrap samples. The length of `theta` is equal to the number of iterations in the previously-run bootstrap simulation.
`conf.level`	the level of the desired confidence interval, as a proportion. Defaults to .95 which returns the 95 percent confidence interval.

Details

BC_a confidence intervals are typically calculated using influence statistics from jackknife simulations. For our purposes, however, running jackknife simulation in addition to ordinary bootstrapping is too computationally expensive. This function follows the procedure outlined by DiCiccio and Efron (1996, p. 201) to calculate the bias-correction and acceleration parameters using only the draws from ordinary bootstrapping.

Value

returns a vector of length 2 in which the first element is the lower bound and the second element is the upper bound

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden@nd.edu>, based on the code for the `mediate` function in the `mediation` package by Dustin Tingley, Teppei Yamamoto, Kentaro Hirose, Luke Keele, and Kosuke Imai.

References

DiCiccio, T. J. and B. Efron. (1996). Bootstrap Confidence Intervals. *Statistical Science*. 11(3): 189–212. https://doi.org/10.1214/ss/1032280214

See Also

coxed, bootcov, mediate

Examples

```
theta <- rnorm(1000, mean=3, sd=4)
bca(theta, conf.level = .95)
```

---

bootcov2 *Boostrapping algorithm for* coxed

Description

This function uses bootstrapping to create standard errors and confidence intervals for the quantities produced by the coxed() function. It is adapted from the bootcov function in the rms package. It is called by the coxed function and is not intended to be used by itself. Please refer to the original bootcov function for general bootstrapping applications.

Usage

```
bootcov2(fit, cluster, B = 200, fitter, coef.reps = TRUE,
  loglik = FALSE, pr = FALSE, maxit = 15, group = NULL,
  stat = NULL)
```

Arguments

fit	an estimated Cox proportional hazards model object with class "coxph" or "cph"
cluster	a variable indicating groupings. cluster may be any type of vector (factor, character, integer). Unique values of cluster indicate possibly correlated groupings of observations. Note the data used in the fit and stored in fit$x and fit$y may have had observations containing missing values deleted. It is assumed that if there were any NAs, an naresid function exists for the class of fit. This function restores NAs so that the rows of the design matrix coincide with cluster
B	Number of bootstrap simulation iterations
fitter	the name of a function with arguments (x,y) that will fit bootstrap samples. Default is taken from the class of fit if it is ols, lrm, cph, psm, Rq. If fitter="tvc" the function employs agreg.fit
coef.reps	set to TRUE if you want to store a matrix of all bootstrap regression coefficient estimates in the returned component boot.Coef.
loglik	set to TRUE to store -2 log likelihoods for each bootstrap model, evaluated against the original x and y data. The default is to do this when coef.reps is specified as TRUE. The use of loglik=TRUE assumes that an oos.loglik method exists for the type of model being analyzed, to calculate out-of-sample -2 log likelihoods (see rmsMisc). After the B -2 log likelihoods (stored in the element named boot.loglik in the returned fit object), the B+1 element is the -2 log likelihood for the original model fit

pr	set to TRUE to print the current sample number to monitor progress
maxit	maximum number of iterations, to pass to fitter
group	a grouping variable used to stratify the sample upon bootstrapping. This allows one to handle k-sample problems, i.e., each bootstrap sample will be forced to select the same number of observations from each level of group as the number appearing in the original dataset. You may specify both group and cluster
stat	a single character string specifying the name of a stats element produced by the fitting function to save over the bootstrap repetitions. The vector of saved statistics will be in the boot.stats part of the list returned by bootcov

Details

This function contains the same code as the bootcov function in the rms package, with a few alterations to work better with the coxed function. First, we output a result attribute b.ind, which contains the observation numbers from the estimation sample that are drawn with replacement to produce the bootstrap sample and takes into account clustering. Second, we program a new class, tvc, for fitter to use agreg.fit instead of coxph.fit when the data contain time-varying covariates.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden@nd.edu>, based on the code for the bootcov function in the rms package by Frank Harrell and Bill Pikounis

See Also

coxed, coxph, cph, bootcov

---

boxsteffensmeier *Data from "A Dynamic Analysis of The Role of War Chests in Campaign Strategy" by Janet M. Box-Steffensmeier*

Description

The data cover 397 races for the United States House of Representatives in which an incumbent ran for reelection in 1990.

Usage

```
boxsteffensmeier
```

Format

A data frame with 1376 observations and 8 variables:

caseid	ID number
start	Beginning of the measured time interval, in weeks
te	Time to challenger entry (end of the time interval), in weeks
ec	The amount of money (in millions of USD) the incumbent has in reserve
dem	1 if the incumbent is a Democrat, 0 if the incumbent is a Republican
south	1 if the district is in the south, 0 else
iv	Prior vote percent
cut_hi	Indicates whether or not a high quality challenger enters the race (censoring variable)

Source

Box-Steffensmeier, J. M. (1996) A Dynamic Analysis of The Role of War Chests in Campaign Strategy. *American Journal of Political Science* **40** 352-371

---

censor.x *Generate right-censoring to be dependent on the data*

Description

This function is called by `sim.survdata` and is not intended to be used by itself.

Usage

```
censor.x(x, censor = 0.1)
```

Arguments

x	A matrix or data frame containing covariates
censor	The proportion of observations to designate as being right-censored

Details

The purpose of this function is to efficiently generate indicators for whether or not an observation in simulated duration data is right-censored. In this case, whether or not an observation is right-censored depends on the covariates in the data.

This function randomly draws new coefficients, one for every column of `x`, from the normal distribution with mean 0 and standard deviation of 0.1. It uses these new coefficients to build a linear predictor, to which is added a disturbance term which is also drawn from N(0,.1). A fixed proportion, given by `censor`, of the observations with the highest values of this linear predictor are set to be `TRUE` and the others are set to `FALSE`.

Value

A vector of logical values

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

sim.survdata

Examples

```
Xdata <- matrix(rnorm(300), 100, 3)
censor.x(Xdata, .1)
```

---

coxed *Expected durations and marginal changes in expected duration from the Cox proportional hazards model*

Description

coxed() returns expected durations for every observation in the data used to fit the model, or in new data, or returns the mean or median of these durations, or differences in duration for two pre-defined covariate profiles. Standard errors and confidence intervals for all quantities produced by coxed() are calculated via bootstrapping.

Usage

```
coxed(cox.model, newdata = NULL, newdata2 = NULL, bootstrap = FALSE,
  method = "npsf", k = -1, B = 200, confidence = "studentized",
  level = 0.95, id = NULL, ...)
```

Arguments

cox.model	The output from a Cox proportional hazards model estimated with the coxph function in the survival package or with the cph function in the rms package
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used
newdata2	An optional data frame that can only be specified if newdata is not omitted, and must have the same dimensions as newdata. If specified, marginal changes are calculated by subtracting the expected durations for newdata2 from the expected durations for newdata
bootstrap	Should bootstrapped standard errors and confidence intervals be calculated?
method	If "npsf" (the default), expected durations are calculated using the non-parametric step function approach described in Kropko and Harden (2018). If "gam", expected durations are calculated using the GAM method

k	The number of knots in the GAM smoother. The default is -1, which employs the choose.k function from the mgcv package to choose the number of knots
B	Number of bootstrap simulation iterations
confidence	If "studentized" (the default), bootstrapped CIs are calculated from the tails of a normal distribution where the mean and standard deviation are the point estimate and boostrapped SE of each duration estimate. If "empirical", bootstrapped confidence intervals are calculated empirically. If "bca", bootstrapped confidence intervals are calculated using the bias-correction and acceleration method described by DiCiccio and Efron (1996).
level	The level of the confidence interval to calculate (default is .95 for a 95 percent confidence interval)
id	Cluster variable if bootstrapping is to be done by clusters of observations rather than individual observations. If the data are coded with time-varying covariates (using the time2 argument in the Surv function), this variable must be the ID variable in the *data that are used to estimate the Cox PH model*, and not the ID variable in new data.
...	Additional arguments to be passed to the bootcov2 function, an adaptation of the bootcov function in the rms package

Details

The coxed function generates expected durations for individual observations and/or marginal changes in expected duration given a change in a covariate from the Cox proportional hazards model. Specifically, the methods can compute (1) the expected duration for each observation used to fit the Cox model, given the covariates, (2) the expected duration for a "new" observation with a covariate profile set by the analyst, or (3) the first difference, or change, in expected duration given two new data frames.

There are two different methods, described in Kropko and Harden (2018), of generating duration-based quantities in the package. The first method calculates expected durations by using a nonparametric estimate of the baseline hazard and survivor functions (see coxed.npsf for details). The second method employs a generalized additive model (GAM) to map the model's estimated linear predictor values to duration times (see coxed.gam for details). Both methods are also implemented for data structures with time-varying covariates (see coxed.npsf.tvc and coxed.gam.tvc).

Value

coxed returns an object of class "coxedExpdur" or "coxedMargin", which is a list containing some of the following components, depending on the implementation of coxed:

exp.dur	A vector of predicted mean durations for the estimation sample if newdata is omitted, or else for the
mean	The mean of the predicted durations. If bootstrap is TRUE bootstrapped standard errors are also prov
median	The median of the predicted durations. If bootstrap is TRUE bootstrapped standard errors are also pr
baseline.functions	The estimated cumulative baseline hazard function and survivor function.
gam.model	Output from the gam function in which the durations are fit against the exponentiated linear predictor
gam.data	Fitted values and confidence intervals from the GAM model.
exp.dur1	A vector of predicted mean durations for the observations in newdata1 when calculating marginal eff
exp.dur2	A vector of predicted mean durations for the observations in newdata2 when calculating marginal eff

mean1	The mean of the predicted mean durations for the observations in newdata1 when calculating margin
mean2	The mean of the predicted mean durations for the observations in newdata2 when calculating margin
median1	The median of the predicted mean durations for the observations in newdata1 when calculating marg
median2	The median of the predicted mean durations for the observations in newdata2 when calculating marg
diff	A vector of the difference between the predicted mean durations for each observation under the covar
mean.diff	The mean of the differences in duration across observations.
median.diff	The median of the differences in duration across observations.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Kropko, J. and Harden, J. J. (2018). Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

DiCiccio, T. J. and B. Efron. (1996). Bootstrap Confidence Intervals. *Statistical Science*. 11(3): 189–212. https://doi.org/10.1214/ss/1032280214

See Also

coxph, cph, bootcov2, coxed.gam, coxed.gam.tvc, coxed.npsf, coxed.npsf.tvc

Examples

```
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm + pgovno +
     tpgovno + minority, method = "breslow", data = martinvanberg)
summary(mv.cox)

# NPSF method
ed1 <- coxed(mv.cox, method="npsf")
ed1$baseline.functions
ed1$exp.dur
summary(ed1, stat="mean")
summary(ed1, stat="median")

## Not run: ed1 <- coxed(mv.cox, method="npsf", bootstrap = TRUE)
ed1$exp.dur
summary(ed1, stat="mean")
summary(ed1, stat="median")

## End(Not run)

me <- coxed(mv.cox, method="npsf", bootstrap = FALSE,
            newdata = dplyr::mutate(martinvanberg, pgovno=1),
            newdata2 = dplyr::mutate(martinvanberg, pgovno=6))
summary(me, stat="mean")
```

```
# GAM method
ed2 <- coxed(mv.cox, method="gam")
summary(ed2$gam.data)
summary(ed2$gam.model)
ed2$exp.dur
summary(ed2, stat="mean")

## Not run: me <- coxed(mv.cox, method="gam", bootstrap = TRUE,
            newdata = dplyr::mutate(martinvanberg, pgovno=1),
            newdata2 = dplyr::mutate(martinvanberg, pgovno=6))
summary(me, stat="mean")
summary(me, stat="median")

## End(Not run)

#Plotting the GAM fit
## Not run: ggplot(ed2$gam.data, aes(x=rank.xb, y=y)) +
    geom_point() +
    geom_line(aes(x=rank.xb, y=gam_fit)) +
    geom_ribbon(aes(ymin=gam_fit_95lb, ymax=gam_fit_95ub), alpha=.5) +
    xlab("Cox model LP rank (smallest to largest)") +
    ylab("Duration")

## End(Not run)

#Time-varying covariates
bs.surv <- Surv(time = boxsteffensmeier$start, time2 = boxsteffensmeier$te,
     event = boxsteffensmeier$cut_hi)
bs.cox <- coxph(bs.surv ~ ec + dem + south + iv, data = boxsteffensmeier, method = "breslow")
summary(bs.cox)

ed1 <- coxed(bs.cox, method="npsf", id=boxsteffensmeier$caseid)
ed1$exp.dur
summary(ed1, stat="mean")
```

`coxed.gam` *Predict expected durations using the GAM method*

Description

This function is called by `coxed` and is not intended to be used by itself.

Usage

```
coxed.gam(cox.model, newdata = NULL, k = -1, coef = NULL,
  b.ind = NULL, warn = TRUE)
```

Arguments

cox.model	The output from a Cox proportional hazards model estimated with the coxph function in the survival package or with the cph function in the rms package
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used
k	The number of knots in the GAM smoother. The default is -1, which employs the choose.k function from the mgcv package to choose the number of knots
coef	A vector of new coefficients to replace the coefficients attribute of the cox.model. Used primarily for bootstrapping, to recalculate durations using new coefficients derived from a bootstrapped sample. If NULL, the original coefficients are employed
b.ind	A vector of observation numbers to pass to the estimation sample to construct the a bootstrapped sample with replacement
warn	If TRUE, displays warnings, and if FALSE suppresses them

Details

This function employs the GAM method of generating expected durations described in Kropko and Harden (2018), which proceeds according to five steps. First, it uses coefficient estimates from the Cox model, so researchers must first estimate the model just as they always have. Then the method computes expected values of risk for each observation by matrix-multiplying the covariates by the estimated coefficients from the model, then exponentiating the result. This creates the exponentiated linear predictor (ELP). Then the observations are ranked from smallest to largest according to their values of the ELP. This ranking is interpreted as the expected order of failure; the larger the value of the ELP, the sooner the model expects that observation to fail, relative to the other observations.

The next step is to connect the model's expected risk for each observation (ELP) to duration time (the observed durations). A gam fits a model to data by using a series of locally-estimated polynomial splines set by the user (see, for example, Wood, Pya, and Saefken 2016). It is a flexible means of allowing for the possibility of nonlinear relationships between variables. coxed.gam uses a GAM to model the observed utilizes a cubic regression spline to draw a smoothed line summarizing the bivariate relationship between the observed durations and the ranks. The GAM fit can be used directly to compute expected durations, given the covariates, for each observation in the data.

Value

Returns a list containing the following components:

exp.dur	A vector of predicted mean durations for the estimation sample if newdata is omitted, or else for the specified n
gam.model	Output from the gam function in which the durations are fit against the exponentiated linear predictors from the C
gam.data	Fitted values and confidence intervals from the GAM model.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Kropko, J. and Harden, J. J. (2018). Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

Wood, S.N., N. Pya and B. Saefken (2016). Smoothing parameter and model selection for general smooth models (with discussion). *Journal of the American Statistical Association* **111**, 1548-1575 http://dx.doi.org/10.1080/01621459.2016.1180986

See Also

gam, coxed

Examples

```
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm +
pgovno + tpgovno + minority, method = "breslow", data = martinvanberg)

ed <- coxed.gam(mv.cox)
summary(ed$gam.data)
summary(ed$gam.model)
ed$exp.dur

#Plotting the GAM fit
## Not run: require(ggplot2)
ggplot(ed$gam.data, aes(x=rank.xb, y=y)) +
    geom_point() +
    geom_line(aes(x=rank.xb, y=gam_fit)) +
    geom_ribbon(aes(ymin=gam_fit_95lb, ymax=gam_fit_95ub), alpha=.5) +
    xlab("Cox model LP rank (smallest to largest)") +
    ylab("Duration")

## End(Not run)

#Running coxed.gam() on a bootstrap sample and with new coefficients
bsample <- sample(1:nrow(martinvanberg), nrow(martinvanberg), replace=TRUE)
newcoefs <- rnorm(8)
ed2 <- coxed.gam(mv.cox, b.ind=bsample, coef=newcoefs)
```

`coxed.gam.tvc` *Predict expected durations using the GAM method with time-varying covariates*

Description

This function is called by coxed and is not intended to be used by itself.

Usage

```
coxed.gam.tvc(cox.model, newdata = NULL, k = -1, coef = NULL,
  b.ind = NULL, warn = TRUE)
```

Arguments

cox.model	The output from a Cox proportional hazards model estimated with the coxph function in the survival package or with the cph function in the rms package
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used
k	The number of knots in the GAM smoother. The default is -1, which employs the choose.k function from the mgcv package to choose the number of knots
coef	A vector of new coefficients to replace the coefficients attribute of the cox.model. Used primarily for bootstrapping, to recalculate durations using new coefficients derived from a bootstrapped sample. If NULL, the original coefficients are employed
b.ind	A vector of observation numbers to pass to the estimation sample to construct the a bootstrapped sample with replacement
warn	If TRUE, displays warnings, and if FALSE suppresses them

Details

This function employs the GAM method of generating expected durations described in Kropko and Harden (2018). See coxed.gam for details. This code replicates the code for cox.gam, but works with data whose structure allows time-varying covariates, and requires using the time2 argument of the Surv function. This function requires the data to be reported as cumulative durations. The GAM model is fit using the ending times for each interval and using only the non-censored observations. Expected durations are drawn from this GAM as with coxed.gam.

Value

Returns a list containing the following components:

exp.dur	A vector of predicted mean durations for the estimation sample if newdata is omitted, or else for the specified n
gam.model	Output from the gam function in which the durations are fit against the exponentiated linear predictors from the C
gam.data	Fitted values and confidence intervals from the GAM model.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Kropko, J. and Harden, J. J. (2018). Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

See Also

gam, coxed, coxed.gam

Examples

```
bs.surv <- Surv(time = boxsteffensmeier$start, time2 = boxsteffensmeier$te,
     event = boxsteffensmeier$cut_hi)
bs.cox <- coxph(bs.surv ~ ec + dem + south + iv, data = boxsteffensmeier, method = "breslow")
summary(bs.cox)

ed <- coxed.gam.tvc(bs.cox)
ed$exp.dur
```

---

coxed.npsf *Predict expected durations using the GAM method*

Description

This function is called by coxed and is not intended to be used by itself.

Usage

```
coxed.npsf(cox.model, newdata = NULL, coef = NULL, b.ind = NULL)
```

Arguments

cox.model	The output from a Cox proportional hazards model estimated with the coxph function in the survival package or with the cph function in the rms package
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used
coef	A vector of new coefficients to replace the coefficients attribute of the cox.model. Used primarily for bootstrapping, to recalculate durations using new coefficients derived from a bootstrapped sample. If NULL, the original coefficients are employed
b.ind	A vector of observation numbers to pass to the estimation sample to construct the a bootstrapped sample with replacement

Details

The non-parametric step function (NPSF) approach to calculating expected durations from the Cox proportional hazards model, described in Kropko and Harden (2018), uses the method proposed by Cox and Oakes (1984, 107-109) for estimating the cumulative baseline hazard function. This method is nonparametric and results in a step-function representation of the cumulative baseline hazard.

Cox and Oakes (1984, 108) show that the cumulative baseline hazard function can be estimated after fitting a Cox model by

$$\hat{H}_0(t) = \sum_{\tau_j < t} \frac{d_j}{\sum_{l \in \Re(\tau_j)} \hat{\psi}(l)},$$

where τ_j represents time points earlier than t, d_j is a count of the total number of failures at τ_j, $\Re(\tau_j)$ is the remaining risk set at τ_j, and $\hat{\psi}(l)$ represents the ELP from the Cox model for observations still in the risk set at τ_j. This equation is used calculate the cumulative baseline hazard at all time points in the range of observed durations. This estimate is a stepwise function because time points with no failures do not contribute to the cumulative hazard, so the function is flat until the next time point with observed failures.

We extend this method to obtain expected durations by first calculating the baseline survivor function from the cumulative hazard function, using

$$\hat{S}_0(t) = \exp[-\hat{H}_0(t)].$$

Each observation's survivor function is related to the baseline survivor function by

$$\hat{S}_i(t) = \hat{S}_0(t)^{\hat{\psi}(i)},$$

where $\hat{\psi}(i)$ is the exponentiated linear predictor (ELP) for observation i. These survivor functions can be used directly to calculate expected durations for each observation. The expected value of a non-negative random variable can be calculated by

$$E(X) = \int_0^\infty \Big(1 - F(t)\Big) dt,$$

where $F(.)$ is the cumulative distribution function for X. In the case of a duration variable t_i, the expected duration is

$$E(t_i) = \int_0^T S_i(t)\, dt,$$

where T is the largest possible duration and $S(t)$ is the individual's survivor function. We approximate this integral with a right Riemann-sum by calculating the survivor functions at every discrete time point from the minimum to the maximum observed durations, and multiplying these values by the length of the interval between time points with observed failures:

$$E(t_i) \approx \sum_{t_j \in [0,T]} (t_j - t_{j-1}) S_i(t_j).$$

Value

Returns a list containing the following components:

`exp.dur`	A vector of predicted mean durations for the estimation sample if `newdata` is omitted, or else for the
`baseline.functions`	The estimated cumulative baseline hazard function and survivor function.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Kropko, J. and Harden, J. J. (2018). Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

Cox, D. R., and Oakes, D. (1984). *Analysis of Survival Data. Monographs on Statistics & Applied Probability*

See Also

coxed

Examples

```
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm + pgovno + tpgovno +
     minority, method = "breslow", data = martinvanberg)

ed <- coxed.npsf(mv.cox)
ed$baseline.functions
ed$exp.dur

#Running coxed.npsf() on a bootstrap sample and with new coefficients
bsample <- sample(1:nrow(martinvanberg), nrow(martinvanberg), replace=TRUE)
newcoefs <- rnorm(8)
ed2 <- coxed.npsf(mv.cox, b.ind=bsample, coef=newcoefs)
```

---

coxed.npsf.tvc *Predict expected durations using the GAM method with time-varying covariates*

Description

This function is called by coxed and is not intended to be used by itself.

Usage

```
coxed.npsf.tvc(cox.model, newdata = NULL, coef = NULL, b.ind = NULL)
```

Arguments

cox.model	The output from a Cox proportional hazards model estimated with the coxph function in the survival package or with the cph function in the rms package
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used

coef	A vector of new coefficients to replace the coefficients attribute of the cox.model. Used primarily for bootstrapping, to recalculate durations using new coefficients derived from a bootstrapped sample. If NULL, the original coefficients are employed
b.ind	A vector of observation numbers to pass to the estimation sample to construct the a bootstrapped sample with replacement

Details

This function employs the NPSF method of generating expected durations described in Kropko and Harden (2018). See coxed.npsf for details. This code replicates the code for cox.npsf, but works with data whose structure allows time-varying covariates, and requires using the time2 argument of the Surv function. This function requires the data to be reported as cumulative durations. The cumulative baseline hazard function model is estimated using the ending times for each interval. Then the expected durations are drawn from the Cox model and the NPSF method as with coxed.npsf.

Value

Returns a list containing the following components:

exp.dur	A vector of predicted mean durations for the estimation sample if newdata is omitted, or else for the
baseline.functions	The estimated cumulative baseline hazard function and survivor function.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Kropko, J. and Harden, J. J. (2018). Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X

See Also

coxed, coxed.npsf

Examples

```
bs.surv <- Surv(time = boxsteffensmeier$start, time2 = boxsteffensmeier$te,
     event = boxsteffensmeier$cut_hi)
bs.cox <- coxph(bs.surv ~ ec + dem + south + iv, data = boxsteffensmeier, method = "breslow")

ed <- coxed.npsf.tvc(bs.cox)
ed$exp.dur
```

`data.plot` *Plot the histograms of simulated data*

Description

This function is called by `survsim.plot` and is not intended to be used by itself.

Usage

```
data.plot(data, xb, bins = 30)
```

Arguments

`data`	A data frame of simulated duration data in which the duration variable is called y. Generally, this data frame is taken from the `data` attribute of the `sim.survdata` function
`xb`	A vector of the linear predictors, combining the covariates and coefficients from the simulation. Generally, this data frame is taken from the `xb` attribute of the `sim.survdata` function
`bins`	The number of bins to draw in each histogram

Details

This function produces three histograms, one of the simulated durations, one of the linear predictors, and one of the exponentiated linear predictors. It uses `facet_wrap` within a call to `ggplot` to arrange the plots.

Value

A figure of class `"gg"` and `"ggplot"`

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

`survsim.plot`, `sim.survdata`

Examples

```
simdata <- sim.survdata(N=1000, T=100, num.data.frames=1)
data.plot(simdata$data, simdata$xb)
```

`generate.lm` *Generate simulated durations using a baseline survivor function and proportional hazards*

Description

This function is called by `sim.survdata` and is not intended to be used by itself.

Usage

```
generate.lm(baseline, X = NULL, N = 1000, type = "none",
  beta = NULL, xvars = 3, mu = 0, sd = 1, censor = 0.1)
```

Arguments

`baseline`	The baseline hazard, cumulative hazard, survival, failure PDF, and failure CDF as output by `baseline.build`
`X`	A user-specified data frame containing the covariates that condition duration. If `NULL`, covariates are generated from normal distributions with means given by the `mu` argument and standard deviations given by the `sd` argument
`N`	Number of observations in each generated data frame
`type`	If "none" (the default) data are generated with no time-varying covariates or coefficients. If "tvc", data are generated with time-varying covariates, and if "tvbeta" data are generated with time-varying coefficients (see details)
`beta`	A user-specified vector containing the coefficients that for the linear part of the duration model. If `NULL`, coefficients are generated from normal distributions with means of 0 and standard deviations of 0.1
`xvars`	The number of covariates to generate. Ignored if `X` is not `NULL`
`mu`	If scalar, all covariates are generated to have means equal to this scalar. If a vector, it specifies the mean of each covariate separately, and it must be equal in length to `xvars`. Ignored if `X` is not `NULL`
`sd`	If scalar, all covariates are generated to have standard deviations equal to this scalar. If a vector, it specifies the standard deviation of each covariate separately, and it must be equal in length to `xvars`. Ignored if `X` is not `NULL`
`censor`	The proportion of observations to designate as being right-censored

Details

If `type="none"` then the function generates idiosyncratic survival functions for each observation via proportional hazards: first the linear predictor is calculated from the X variables and beta coefficients, then the linear predictor is exponentiated and set as the exponent of the baseline survivor function. For each individual observation's survival function, a duration is drawn by drawing a single random number on U[0,1] and finding the time point at which the survival function first decreases past this value. See Harden and Kropko (2018) for a more detailed description of this algorithm.

If type="tvc", this function cannot accept user-supplied data for the covariates, as a time-varying covariate is expressed over time frames which themselves convey part of the variation of the durations, and we are generating these durations. If user-supplied X data is provided, the function passes a warning and generates random data instead as if X=NULL. Durations are drawn again using proportional hazards, and are passed to the permalgorithm function in the PermAlgo package to generate the time-varying data structure (Sylvestre and Abrahamowicz 2008).

If type="tvbeta" the first coefficient, whether coefficients are user-supplied or randomly generated, is interacted with the natural log of the time counter from 1 to T (the maximum time point for the baseline functions). Durations are generated via proportional hazards, and coefficients are saved as a matrix to illustrate their dependence on time.

Value

Returns a list with the following components:

data	The simulated data frame, including the simulated durations, the censoring variable, and covariates
beta	The coefficients, varying over time if type is "tvbeta"
XB	The linear predictor for each observation
exp.XB	The exponentiated linear predictor for each observation
survmat	An (N x T) matrix containing the individual survivor function at time t for the individual represented by row n
tvc	A logical value indicating whether or not the data includes time-varying covariates

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Harden, J. J. and Kropko, J. (2018). Simulating Duration Data for the Cox Model. *Political Science Research and Methods* https://doi.org/10.1017/psrm.2018.19

Sylvestre M.-P., Abrahamowicz M. (2008) Comparison of algorithms to generate event times conditional on time-dependent covariates. *Statistics in Medicine* **27(14)**:2618–34.

See Also

sim.survdata, permalgorithm

Examples

```
baseline <- baseline.build(T=100, knots=8, spline=TRUE)
simdata <- generate.lm(baseline, N=1000, xvars=5, mu=0, sd=1, type="none", censor=.1)
summary(simdata$data)
simdata <- generate.lm(baseline, N=1000, xvars=5, mu=0, sd=1, type="tvc", censor=.1)
summary(simdata$data)
simdata <- generate.lm(baseline, N=1000, xvars=5, mu=0, sd=1, type="tvbeta", censor=.1)
simdata$beta
```

`make.margeffect` *Calculating a simulated marginal effect*

Description

This function is called by `sim.survdata` and is not intended to be used by itself.

Usage

```
make.margeffect(baseline, xb, covariate = 1, low = 0, high = 1,
  compare = median)
```

Arguments

`baseline`	The baseline hazard functions, output by `baseline.build`
`xb`	The simulated data, output by `generate.lm`
`covariate`	Specification of the column number of the covariate in the `X` matrix for which to generate a simulated marginal effect (default is 1). The marginal effect is the difference in expected duration when the covariate is fixed at a high value and the expected duration when the covariate is fixed at a low value
`low`	The low value of the covariate for which to calculate a marginal effect
`high`	The high value of the covariate for which to calculate a marginal effect
`compare`	The statistic to employ when examining the two new vectors of expected durations (see details for `sim.survdata`). The default is `median`

Details

The idea is to simulate a marginal change in duration so that researchers can compare the performance of estimators of this statistic using simulated data.

The function calculates simulated durations for each observation conditional on a baseline hazard function and exogenous covariates and coefficients. The `covariate` argument specifies the variable in the X matrix to vary so as to measure the marginal effect. First the covariate is set to the value specified in `low` for all observations, then to the value specified in `high` for all observations. Given each value, new durations are drawn. The durations when the covariate equals the low value are subtracted from the durations when the covariate equals the high value. The marginal effect is calculated by employing the statistic given by `compare`, which is `median` by default.

Value

A list with three items:

`marg.effect`	A scalar containing the simulated marginal effect
`data.low`	The durations and covariates when the covariate of interest is set to the low value
`data.high`	The durations and covariates when the covariate of interest is set to the high value

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

baseline.build, generate.lm, sim.survdata

Examples

```
T <- 100
N <- 1000
X <- as.matrix(data.frame(X1=rnorm(N), X2=rnorm(N), X3=rnorm(N)))
beta <- as.matrix(rnorm(3))
baseline <- baseline.build(T=T, knots=8, spline=TRUE)
xb <- generate.lm(baseline, X=X, beta=beta, N=N, censor=.1, type="none")
me <- make.margeffect(baseline, xb, covariate=1, low=0, high=1)
me$marg.effect
```

---

martinvanberg — *Data from "Wasting Time? The Impact of Ideology and Size on Delay in Coalition Formation" by Lanny W. Martin and Georg Vanberg*

Description

The data cover all negotiations to form a government after parliamentary elections in Austria, Belgium, Denmark, Germany, Ireland, Italy, Luxembourg, Netherlands, Norway, and Sweden between 1950 and 1995.

Usage

```
martinvanberg
```

Format

A data frame with 203 observations and 8 variables:

formdur	The number of days until negotiations conclude
postel	Indicator for negotiations beginning after an election
prevdef	Indicator for governments following a parliamentary defeat
cont	Indicator for government with continuation rule
ident	Measure of voters' ease of identifying alternative coalitions
rgovm	Ideological range of the coalition parties
pgovno	Number of parties in coalition
tpgovno	Number of parties * log(time)
minority	Indicator for minority government

Source

Martin, L. W and Vanberg, G. (2003) Wasting Time? The Impact of Ideology and Size on Delay in Coalition Formation. *British Journal of Political Science* **33** 323-344 https://doi.org/10.1017/S0007123403000140

`rank.predict` *Generate predicted ranks for new observations given a new covariate profile*

Description

This function is called by `coxed` when `method="gam"` and new data are specified, and is not intended to be used by itself.

Usage

```
rank.predict(x, v, warn = TRUE)
```

Arguments

`x`	A vector of linear predictors for the estimation sample
`v`	A vector of linear predictors for the new data
`warn`	If `TRUE` the function warns the user when linear predictors in the new data are greater than or less than all of the linear predictors in the estimation sample

Details

The purpose of `rank.predict` is to determine for a single new observation what the rank of that observation's linear predictor would have been had the observation been in the original estimation sample. It calculates the predicted rank by appending the new observation to the vector of linear predictors for the estimation sample and calculating the `rank` of the observation in the new vector. If the new data contain more than one observation, `rank.predict` calculates the predicted rank for each observation independently, without taking the other observations in the new data into account.

Any observation with a linear predictor less than the minimum linear predictor in the estimation sample will have a predicted rank of 1; any observation with a linear predictor greater than the maximum linear predictor in the estimation sample will have a predicted rank of `length(v)`. If either condition exists, the function provides a warning.

Value

A numeric vector containing the predicted ranks for the observations in `x`.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

coxed, rank

Examples

```
estimationLPs <- rnorm(20)
cbind(estimationLPs, rank(estimationLPs))
newLPs <- rnorm(5)
newLP.rank <- rank.predict(x=newLPs, v=estimationLPs)
cbind(newLPs, newLP.rank)
```

---

sim.survdata *Simulating duration data for the Cox proportional hazards model*

Description

sim.survdata() randomly generates data frames containing a user-specified number of observations, time points, and covariates. It generates durations, a variable indicating whether each observation is right-censored, and "true" marginal effects. It can accept user-specified coefficients, covariates, and baseline hazard functions, and it can output data with time-varying covariates or using time-varying coefficients.

Usage

```
sim.survdata(N = 1000, T = 100, type = "none", hazard.fun = NULL,
  num.data.frames = 1, fixed.hazard = FALSE, knots = 8,
  spline = TRUE, X = NULL, beta = NULL, xvars = 3, mu = 0,
  sd = 0.5, covariate = 1, low = 0, high = 1, compare = median,
  censor = 0.1, censor.cond = FALSE)
```

Arguments

N	Number of observations in each generated data frame. Ignored if X is not NULL
T	The latest time point during which an observation may fail. Failures can occur as early as 1 and as late as T
type	If "none" (the default) data are generated with no time-varying covariates or coefficients. If "tvc", data are generated with time-varying covariates, and if "tvbeta" data are generated with time-varying coefficients (see details)
hazard.fun	A user-specified R function with one argument, representing time, that outputs the baseline hazard function. If NULL, a baseline hazard function is generated using the flexible-hazard method as described in Harden and Kropko (2018) (see details)
num.data.frames	The number of data frames to be generated

fixed.hazard	If TRUE, the same hazard function is used to generate each data frame. If FALSE (the default), different drawn hazard functions are used to generate each data frame. Ignored if hazard.fun is not NULL or if num.data.frames is 1
knots	The number of points to draw while using the flexible-hazard method to generate hazard functions (default is 8). Ignored if hazard.fun is not NULL
spline	If TRUE (the default), a spline is employed to smooth the generated cumulative baseline hazard, and if FALSE the cumulative baseline hazard is specified as a step function with steps at the knots. Ignored if hazard.fun is not NULL
X	A user-specified data frame containing the covariates that condition duration. If NULL, covariates are generated from normal distributions with means given by the mu argument and standard deviations given by the sd argument
beta	Either a user-specified vector containing the coefficients that for the linear part of the duration model, or a user specified matrix with rows equal to T for pre-specified time-varying coefficients. If NULL, coefficients are generated from normal distributions with means of 0 and standard deviations of 0.1
xvars	The number of covariates to generate. Ignored if X is not NULL
mu	If scalar, all covariates are generated to have means equal to this scalar. If a vector, it specifies the mean of each covariate separately, and it must be equal in length to xvars. Ignored if X is not NULL
sd	If scalar, all covariates are generated to have standard deviations equal to this scalar. If a vector, it specifies the standard deviation of each covariate separately, and it must be equal in length to xvars. Ignored if X is not NULL
covariate	Specification of the column number of the covariate in the X matrix for which to generate a simulated marginal effect (default is 1). The marginal effect is the difference in expected duration when the covariate is fixed at a high value and the expected duration when the covariate is fixed at a low value
low	The low value of the covariate for which to calculate a marginal effect
high	The high value of the covariate for which to calculate a marginal effect
compare	The statistic to employ when examining the two new vectors of expected durations (see details). The default is median
censor	The proportion of observations to designate as being right-censored
censor.cond	Whether to make right-censoring conditional on the covariates (default is FALSE, but see details)

Details

The sim.survdata function generates simulated duration data. It can accept a user-supplied hazard function, or else it uses the flexible-hazard method described in Harden and Kropko (2018) to generate a hazard that does not necessarily conform to any parametric hazard function. It can generate data with time-varying covariates or coefficients. For time-varying covariates type="tvc" it employs the permutational algorithm by Sylvestre and Abrahamowicz (2008). For time-varying coefficients with type="tvbeta", the first beta coefficient that is either supplied by the user or generated by the function is multiplied by the natural log of the failure time under consideration.

If fixed.hazard=TRUE, one baseline hazard is generated and the same function is used to generate all of the simulated datasets. If fixed.hazard=FALSE (the default), a new hazard function is generated with each simulation iteration.

The flexible-hazard method employed when hazard.fun is NULL generates a unique baseline hazard by fitting a curve to randomly-drawn points. This produces a wide variety of shapes for the baseline hazard, including those that are unimodal, multimodal, monotonically increasing or decreasing, and many other shapes. The method then generates a density function based on each baseline hazard and draws durations from it in a way that circumvents the need to calculate the inverse cumulative baseline hazard. Because the shape of the baseline hazard can vary considerably, this approach matches the Cox model's inherent flexibility and better corresponds to the assumed data generating process (DGP) of the Cox model. Moreover, repeating this process over many iterations in a simulation produces simulated samples of data that better reflect the considerable heterogeneity in data used by applied researchers. This increases the generalizability of the simulation results. See Harden and Kropko (2018) for more detail.

When generating a marginal effect, first the user specifies a covariate by typing its column number in the X matrix into the covariate argument, then specifies the high and low values at which to fix this covariate. The function calculates the differences in expected duration for each observation when fixing the covariate to the high and low values. If compare is median, the function reports the median of these differences, and if compare is mean, the function reports the median of these differences, but any function may be employed that takes a vector as input and outputs a scalar.

If censor.cond is FALSE then a proportion of the observations specified by censor is randomly and uniformly selected to be right-censored. If censor.cond is TRUE then censoring depends on the covariates as follows: new coefficients are drawn from normal distributions with mean 0 and standard deviation of 0.1, and these new coefficients are used to create a new linear predictor using the X matrix. The observations with the largest (100 x censor) percent of the linear predictors are designated as right-censored.

Value

Returns an object of class "simSurvdata" which is a list of length num.data.frames for each iteration of data simulation. Each element of this list is itself a list with the following components:

data	The simulated data frame, including the simulated durations, the censoring variable, and covariates
xdata	The simulated data frame, containing only covariates
baseline	A data frame containing every potential failure time and the baseline failure PDF, baseline failure CDF,
xb	The linear predictor for each observation
exp.xb	The exponentiated linear predictor for each observation
betas	The coefficients, varying over time if type is "tvbeta"
ind.survive	An (N x T) matrix containing the individual survivor function at time t for the individual represented by
marg.effect	The simulated marginal change in expected duration comparing the high and low values of the variable
marg.effect.data	The X matrix and vector of durations for the low and high conditions

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Harden, J. J. and Kropko, J. (2018). Simulating Duration Data for the Cox Model. *Political Science Research and Methods* https://doi.org/10.1017/psrm.2018.19

Sylvestre M.-P., Abrahamowicz M. (2008) Comparison of algorithms to generate event times conditional on time-dependent covariates. *Statistics in Medicine* **27(14)**:2618–34.

Examples

```
simdata <- sim.survdata(N=1000, T=100, num.data.frames=2)
require(survival)
data <- simdata[[1]]$data
model <- coxph(Surv(y, failed) ~ X1 + X2 + X3, data=data)
model$coefficients ## model-estimated coefficients
simdata[[1]]$betas ## "true" coefficients

## User-specified baseline hazard
my.hazard <- function(t){ #lognormal with mean of 50, sd of 10
dnorm((log(t) - log(50))/log(10)) /
     (log(10)*t*(1 - pnorm((log(t) - log(50))/log(10))))
}
simdata <- sim.survdata(N=1000, T=100, hazard.fun = my.hazard)

## A simulated data set with time-varying covariates
## Not run: simdata <- sim.survdata(N=1000, T=100, type="tvc", xvars=5, num.data.frames=1)
summary(simdata$data)
model <- coxph(Surv(start, end, failed) ~ X1 + X2 + X3 + X4 + X5, data=simdata$data)
model$coefficients ## model-estimated coefficients
simdata$betas ## "true" coefficients

## End(Not run)

## A simulated data set with time-varying coefficients
simdata <- sim.survdata(N=1000, T=100, type="tvbeta", num.data.frames = 1)
simdata$betas
```

---

summary.coxedExpdur *The mean or median expected duration*

Description

This function takes the output of coxed and calculates the mean or median of the expected durations across the observations for which durations are estimated.

Usage

```
## S3 method for class 'coxedExpdur'
summary(object, stat = "mean", ...)
```

Arguments

object	The output from coxed. If newdata2=NULL, so that coxed is being used to predict duration instead of to calculate a marginal effect, then the class of the coxed output will be "coxedExpdur" and this function will be called by the generic summary function
stat	Either "mean" or "median"
...	For future methods

Details

If bootstrap=TRUE in the call to coxed then a bootstrapped standard error and confidence interval is reported for the given statistic as well.

Value

A scalar containing the mean or median duration, or a vector that also includes the bootstrapped standard error and confidence interval for this quantity.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

Examples

```
require(survival)
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm + pgovno + tpgovno +
     minority, method = "breslow", data = martinvanberg)
summary(mv.cox)

# NPSF method
ed1 <- coxed(mv.cox, method="npsf")
ed1$baseline.functions
ed1$exp.dur
summary(ed1, stat="mean")
summary(ed1, stat="median")

ed1 <- coxed(mv.cox, method="npsf", bootstrap = TRUE)
ed1$exp.dur
summary(ed1, stat="mean")
summary(ed1, stat="median")
```

summary.coxedMargin *Marginal changes in expected duration*

Description

This function reports summary statistics for the changes in duration that occur when comparing observations in newdata2 to observations in newdata as specified in the original call to coxed.

Usage

```
## S3 method for class 'coxedMargin'
summary(object, stat = "mean", ...)
```

Arguments

object	The output from coxed. If neither newdata or newdata2 are NULL, coxed is being calculate a marginal effect and the class of the coxed output will be "coxedMargin", so this function will be called by the generic summary function
stat	Either "mean" or "median"
...	For future methods

Details

coxed calculates a vector of expected durations for every observation in newdata2 and newdata (these data frames must have the same number of observations, and corresponding rows are assumed to refer to the same observation), and subtracts the expected duration from newdata2 from the expected duration from newdata. That generates a vector of marginal changes in duration for every observation. To generalize a finding across observations, either the mean (if type="mean") or the median (if type="median") of these differences is reported, along with the mean/median of the expected durations from each of the two covariate profiles. If bootstrap=TRUE in the call to coxed then a bootstrapped standard error and confidence interval is reported for each of these quantities as well.

Value

A data.frame containing the mean or median duration for each of newdata2 and newdata and the differences in these durations across the two covariate profiles. If bootstrap=TRUE in the call to coxed, the data frame also includes the bootstrapped standard error and confidence interval for these quantities.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

See Also

coxed

Examples

```
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm + pgovno + tpgovno +
     minority, method = "breslow", data = martinvanberg)
summary(mv.cox)

me <- coxed(mv.cox, method="npsf", bootstrap = FALSE,
            newdata = dplyr::mutate(martinvanberg, rgovm=0),
            newdata2 = dplyr::mutate(martinvanberg, rgovm=1.24))
summary(me, stat="mean")
summary(me, stat="median")
```

`survsim.plot` *Plot the simulated baseline functions and histograms of simulated data*

Description

This function takes the output of `sim.survdata` and plots the baseline failure PDF, the baseline failure CDF, the baseline survivor function, and the baseline hazard function that were generated and used to simulate the data. The function also produces histograms of the simulated durations, the linear predictor, and the exponentiated linear predictor.

Usage

```
survsim.plot(survsim, type = "both", bins = 30, df = 1)
```

Arguments

`survsim`	An object of class "`simSurvdata`" output by `sim.survdata`
`type`	If `type="baseline"` the function plots the baseline functions. If `type="hist"` the function plots the histograms. If `type="both"` the function plots both the baseline functions and the histograms, aligned in a 2 x 1 array
`bins`	The number of bins to draw in each histogram. Ignored if `type="baseline"`
`df`	If `survsim` is generated by a call to `sim.survdata` in which `num.data.frames` is greater than 1, the output is a list of iterated simulation output. In this case, `df` specifies the particular simulation iteration for which to produce visualizations

Details

A challenge that can limit research to develop methods to analyze survival data is that a pre-specified baseline hazard function must be used to generate simulated data, which contradicts the Cox proportional hazards model's feature of circumventing the hazard to estimate coefficients. The flexible-hazard method developed by Harden and Kropko (2018) and implemented in `sim.survdata` allows for the simulation of duration data without assuming a particular parametric form for hazard. `survsim.plot` is useful for visualizing the baseline functions, including hazard, that result from this algorithm for a particular draw of simulated duration data. The function uses `ggplot` to create line plots for the baseline failure PDF, the baseline failure CDF, the baseline survivor function, and the baseline hazard function over time. The baseline functions and time are attributes of the `sim.survdata` output.

Value

A figure of class "gg" and "ggplot" if type="baseline" or type="hist", and of classes "gtable", "gTree", "grob", and "gDesc" if type="both".

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

References

Harden, J. J. and Kropko, J. (2018). Simulating Duration Data for the Cox Model. *Political Science Research and Methods* https://doi.org/10.1017/psrm.2018.19

See Also

sim.survdata, ggplot, grid.arrange

Examples

```
simdata <- sim.survdata(N=1000, T=100, num.data.frames=1)
survsim.plot(simdata)
```

---

user.baseline *Calculating baseline functions from a user-specified baseline hazard function*

Description

This function is called by sim.survdata and is not intended to be used by itself.

Usage

```
user.baseline(user.fun, T)
```

Arguments

user.fun	A user-specified R function with one argument, representing time, that outputs the baseline hazard function
T	The latest time point during which an observation may fail. Failures can occur as early as 1 and as late as T

Details

user.baseline takes a function as a user-specified baseline hazard which must have only one argument: time. user.baseline approximates the cumulative baseline hazard by taking the cumulative sum of the user-specified hazard function. It calculates the survivor function by exponentiating the cumulative baseline hazard time -1, the baseline failure CDF by subtracting the survivor function from 1, and it approximates the baseline failure PDF by taking the first difference of the failure CDF. survivor function, and baseline failure-time PDF and CDF.

Value

A data frame with five columns representing time from 1 to T, and the user-specified baseline hazard, cumulative hazard, survivor function, failure PDF and failure CDF at each time point.

Author(s)

Jonathan Kropko <jkropko@virginia.edu> and Jeffrey J. Harden <jharden2@nd.edu>

Examples

```
## Writing the hazard to be lognormal with mean of 50, sd of 10
my.hazard <- function(t){
     dnorm((log(t) - log(50))/log(10)) /
          (log(10)*t*(1 - pnorm((log(t) - log(50))/log(10))))
}
lognormal.functions <- user.baseline(my.hazard, 100)
summary(lognormal.functions)

#A customized user-specified hazard
sine.squared.hazard <- user.baseline(function(t) sin(t/25)^2, 30)
summary(sine.squared.hazard)
```

Index