Package 'HQM'

October 30, 2026

Type Package

Title Superefficient Estimation of Future Conditional Hazards Based on Marker Information

Version 2.2

Maintainer Dimitrios Bagkavos <dimitrios.bagkavos@gmail.com>

Description Provides univariate and indexed (multivariate) nonparametric smoothed kernel estimators for the future conditional hazard rate function when time-dependent covariates are present, a bandwidth selector for the estimator's implementation and pointwise and uniform confidence bands. Methods used in the package refer to Bagkavos, Isakson, Mammen, Nielsen and Proust-Lima (2025) <doi:10.1093/biomet/asaf008>.

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports stats, utils, survival, pec, nlme, JM

License GPL (>= 2)

NeedsCompilation no

RoxygenNote 6.1.0

Author Dimitrios Bagkavos [aut, cre],
Alex Isakson [ctb],
Enno Mammen [ctb],
Jens Nielsen [ctb],
Cecile Proust-Lima [ctb]

Repository CRAN

Date/Publication 2026-10-30 15:10:07 UTC

Contents

AUC.hat 2
AUC.hat.HQM 6
bootstrap.CI.pivotal 7
bootstrap.CI.quantile 11

bootstrap.CI.studentized . . . 16
bootstrap.h.hat.star . . . 20
BS.hat.HQM . . . 22
c2.u.integrand . . . 24
ck.u . . . 24
confidence.bands . . . 25
CV.b . . . 27
CV.theta . . . 28
data.minus.I.j . . . 30
double.bootstrap.h.hat.star . . . 31
g.hat.i . . . 32
g.hat.t.x . . . 34
get.alpha.hat . . . 35
get.h.hat.x . . . 37
get.h.hat.x.theta . . . 40
get.h.tilde.x . . . 41
h.hat.minus.I.j . . . 46
h.hat.minus.I.j.theta . . . 47
h.hat.star.x.theta . . . 48
h.hat.x.t . . . 49
h.hat.x.t.vec . . . 51
h.tilde.x.t . . . 57
h.true.simulated . . . 60
iid.AUC.hat . . . 61
iid.G.hat . . . 63
K.b . . . 64
K.b.u.v . . . 65
K.Epanechnikov . . . 66
make.dN.Z.ds . . . 66
make.S.hat . . . 68
pbc2 . . . 69
Q.theta . . . 70
Q1 . . . 72
R.hat.K . . . 74
to.id . . . 75
wild.bootstrap.summands . . . 76
X.i.s.interpolate . . . 78

Index **80**

`AUC.hat` *Time-dependent ROC curve estimation*

Description

Inverse Probability of Censoring Weighting (IPCW) estimation of Cumulative/Dynamic time-dependent ROC curve. The function works in the usual survival setting as well as in the competing risks setting. Computation of the iid-representation of areas under time-dependent ROC curves is implemented. This enables computation of inference procedures: Confidence intervals and tests for comparing two AUCs of two different markers measured on the same subjects.

Usage

```
AUC.hat(T.i, delta.i, pi.i, other.markers = NULL, cause,
     weighting = "marginal", t, ROC = TRUE, iid = FALSE)
```

Arguments

`T.i`	The vector of (censored) event-times.
`delta.i`	The vector of event indicators at the corresponding value of the vector `T.i`. Censored observations must be denoted by the value `0`.
`pi.i`	The vector of the marker values for which we want to compute the time-dependent ROC curves. Without loss of generality, the function assumes that larger values of the marker are associated with higher risks of events. If lower values of the marker are associated with higher risks of events, then reverse the association adding a minus to the marker values.
`other.markers`	A matrix that contains values of other markers that we want to take into account for computing the inverse probability of censoring weights. The different columns represent the different markers. This argument is optional, and ignored if `method="marginal"`. Default value is `other.markers=NULL`.
`cause`	The value of the event indicator that represents the event of interest for which we aim to compute the time-dependent ROC curve. Without competing risks, it must be the value that indicates a non-censored obsevation (usually `1`). With competing risks, subjects can undergo different type of events; then, it must be the value corresponding to the event of interest, for which we aim to compute the ROC curve (usually `1` or `2`).
`weighting`	The method used to compute the weights. `weighting="marginal"` uses the Kaplan-Meier estimator of the censoring distribution. `weighting="cox"` and `weighting="aalen"` model the censoring by the Cox model and the additive Aalen model respectively. Default value is `weighting="marginal"`.
`t`	The vector of times points "t" at which we want to compute the time-dependent ROC curve. If vector `t` contains only a single value, then value zero is added.
`ROC`	A logical value that indicates if we want to save the estimates of sensitivities and specificties. Default value is `ROC = TRUE`.
`iid`	A logical value that indicates if we want to compute the iid-representation of the area under time-dependent ROC curve estimator. `iid = TRUE` is required for computation of all inference procedures (Confidence intervals or test for comparing AUCs). For large sample size (greater than 2000, say) and/or large length of vector `t`, the computation of the iid representations might be time-consuming. Default value is `iid = FALSE`.

Details

This function computes Inverse Probability of Censoring Weighting (IPCW) estimates of Cumulative/Dynamic time-dependent ROC curve. By definition, time-dependent ROC curve intrinsically depends on the definitions of time-dependent cases and controls.
Let T_i denote the event time of the subject i.

Without competing risks : A case is defined as a subject i with $T_i \leq t$. A control is defined as a subject i with $T_i > t$.

With competing risks : In this setting, subjects may undergo different type of events, denoted by δ_i in the following. Let suppose that we are interested in the event $\delta_i = 1$. Then, a case is defined as a subject i with $T_i \leq t$ and $\delta_i = 1$. With competing risks, two definitions of controls were suggested: **(i)** a control is defined as a subject i that is free of any event, i.e with $T_i > t$, and **(ii)** a control is defined as a subject i that is not a case, i.e with $T_i > t$ or with $T_i \leq t$ and $\delta_i \neq 1$. For all outputs of this package, objects named with `.1` refer to definition **(i)**. For instance `AUC.hat.1` or `se.1` refer to time-dependent area under the ROC curve and its estimated standard error according to the definition **(i)**. Objects named with `.2` refer to definition **(ii)** .

Value

Object of class "ipcwsurvivalROC" or "ipcwcompetingrisksROC", depending on if there is competing risk or not, that is a list. For these classes, there are print, plot and confint methods. Most objects that they contain are similar, but some are specific to each class.

Specific objects of class "ipcwsurvivalROC" :

- `AUC.hat` : vector of time-dependent AUC estimates at each time points.
- `TP` : matrix of time-dependent True Positive fraction (sensitivity) estimates.
- `FP` : matrix of time-dependent False Positive fraction (1-specificity) estimates.

Specific objects of class "ipcwcompetingrisksROC" :

- `AUC.hat.1` : vector of time-dependent AUC estimates at each time points with definition **(i)** of controls (see Details).
- `AUC.hat.2` : vector of time-dependent AUC estimates at each time points with definition **(ii)** of controls (see Details).
- `TP` : matrix of time-dependent True Positive fraction (sensitivity) estimates.
- `FP.1` : matrix of time-dependent False Positive fraction (1-specificity) estimates with definition **(i)** of controls (see Details).
- `FP.2` : matrix of time-dependent False Positive fraction (1-specificity) estimates with definition **(ii)** of controls (see Details).

Objects common to both classes :

- `t` : the time points for which the time-dependent ROC curves were computed.
- `G.hat` : a object of class "IPCW", containing all informations about the weights. See `ipcw` function of `pec` package.
- `computation.time` : the total computation time.
- `cum.incidence` : the vector of estimated probabilities of being a case at each time points.

- `S.t` : the vector of estimated probabilities of being event-free at each time points.
- `counts.at.t` : a matrix containing descriptive statistics at each time points (like numbers of observed cases or censored observations before each time points).
- `iid` : the logical value of parameter `iid` used in argument.
- `n` : the sample size, after having omitted missing vaues.
- `inference` : a list that contains, among other things, iid-representations and estimated standard errors of the estimators, and that is used for computation of comparison tests and confidence intervals.
- `computation.time` : the computation time, in seconds.

Author(s)

Paul Blanche <pabl@sund.ku.dk>

References

Hung, H. and Chiang, C. (2010). Estimation methods for time-dependent AUC with survival data. Canadian Journal of Statistics, 38(1):8-26

Uno, H., Cai, T., Tian, L. and Wei, L. (2007). Evaluating prediction rules for t-years survivors with censored regression models. Journal of the American Statistical Association, 102(478):527-537.

Blanche, P., Dartigues, J. F., & Jacqmin-Gadda, H. (2013). Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks. Statistics in medicine, 32(30), 5381-5397.

P. Blanche, A. Latouche, V. Viallon (2013). Time-dependent AUC with right-censored data: A Survey. Risk Assessment and Evaluation of Predictions, 239-251, Springer, https://arxiv.org/abs/1210.6805.

Examples

```
library(survival)
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))

timesS2 <- seq(Landmark,14,by=0.5)
b=0.9
h.hat.x<- get.h.hat.x(pbcT1, 'albumin', s.grid = s.grid,  T.name = 'years',
               s.name = 'year', delta.name = 'status2', 2, 0.9)
s.grid.2  = seq(Landmark, 14,  length=n.grid-1)
S.hat.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/4 , h.hat.x)
t <- 1.5
AUC.hat.s.t<-AUC.hat.HQM(pbcT1, S.hat.alb, Landmark,t,
              T.name = 'years', delta.name = 'status2')
AUC.hat.s.t
```

AUC.hat.HQM *AUC for the High Quality Marker estimator*

Description

Calculates the AUC for the HQM estimator.

Usage

```
AUC.hat.HQM(data, S.hat, s, t, T.name, delta.name)
```

Arguments

data	A data frame containing event times and the patient status.
S.hat	The HQM estimator values, typically the output of make.S.hat.
s	Landmark time.
t	Time horizon.
T.name	The column name of the event times in the data data frame.
delta.name	The column name of the status variable in the data frame.

Details

The function AUC.hat.HQM implements the AUC calculation for the HQM estimator estimator.

Value

A vector of two values: the landmark time of the calculation and the AUC value.

See Also

BS.hat.HQM

Examples

```
library(survival)
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))

timesS2 <- seq(Landmark,14,by=0.5)
b=0.9
h.hat.x<- get.h.hat.x(pbcT1, 'albumin', s.grid = s.grid,  T.name = 'years',
                s.name = 'year', delta.name = 'status2', 2, 0.9)
s.grid.2  = seq(Landmark, 14,  length=n.grid-1)
S.hat.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/4 , h.hat.x)
```

```
t <- 1.5
AUC.hat.s.t<-AUC.hat.HQM(pbcT1, S.hat.alb, Landmark,t,
              T.name = 'years', delta.name = 'status2')
AUC.hat.s.t
```

`bootstrap.CI.pivotal` *Compute Pivot Pointwise Confidence Intervals for the Indexed Hazard Rate Estimate*

Description

Computes bootstrap pivot and symmetric bootstrap-pivot confidence intervals for the hazard, log-hazard, and back-transformed (log-scale) hazard rate functions, using the indexed hazard estimator.

Usage

```
bootstrap.CI.pivotal(B, n.grid, h.hat.star.matrix, t.grid, h.hat.x, alpha.level)
```

Arguments

`B`	Integer. Number of bootsrap replications
`n.grid`	Integer. Number of estimation points at which the indexed hazard estimates and confidence intervals are evaluated.
`h.hat.star.matrix`	A matrix of bootstrap estimated hazard rates with dimensions `n.grid` × `B`, where each column corresponds to one bootstrap replicate.
`t.grid`	Numeric vector of length `n.grid`: the grid points at which the indexed hazard estimates and confidence intervals are calculated.
`h.hat.x`	Indexed hazard estimator, calculated at the grid points `t.grid` and using the original sample.
`alpha.level`	The significance level (e.g., 0.05) which will be used in computing the confidence intervals.

Details

This function computes several forms of pivot confidence intervals for indexed hazard rate estimates. Let $\hat{\sigma}$ be the sample standard deviation of the bootstrap estimators $\hat{h}_x^{(j)}(t), j = 1, \ldots, B$ and let $k^p_{\alpha/2}, k^p_{1-\alpha/2}$ and $\bar{k}^p_{1-\alpha}$ be the $\alpha/2, 1-\alpha/2$ and $1-\alpha$ quantiles respectively of

$$\frac{\hat{h}_x^{(j)}(t) - \hat{h}_x(t)}{\hat{\sigma}},\ j = 1, \ldots, B.$$

Then, the pivot bootstrap confidence interval (CI) for $\hat{h}_x(t)$ is given by

$$\left(\hat{h}_x(t) - \hat{\sigma} k^p_{1-\alpha/2}, \hat{h}_x(t) - \hat{\sigma} k^p_{\alpha/2}\right).$$

The symmetric pivot bootstrap CI for $\hat{h}_x(t)$ is

$$\left(\hat{h}_x(t) - \hat{\sigma}\bar{k}^p_{1-\alpha}, \hat{h}_x(t) + \hat{\sigma}\bar{k}^p_{1-\alpha}\right).$$

For the confidence intervals for the logarithm of the hazard rate function, first let

$$L_x(t) = \log\left\{h_x(t)\right\}, \hat{L}_x(t) = \log\left\{\hat{h}_x(t)\right\}, \hat{L}^{(j)}_x(t) = \log\left\{\hat{h}^{(j)}_x(t)\right\},$$

and $k^{L,p}_{\alpha/2}, k^{L,p}_{1-\alpha/2}$ and $\bar{k}^{L,p}_{1-\alpha}$ be respectively the $\alpha/2, 1-\alpha/2$ and $1-\alpha$ quantile of

$$\frac{|\hat{L}^{(j)}_x(t) - \hat{L}_x(t)|}{\hat{\sigma}}, j = 1, \ldots, B.$$

For the log hazard function $L_x(t)$ a pivotal bootstrap confidence interval is

$$\left(\hat{L}_x(t) - \hat{\sigma}k^{L,p}_{1-\alpha/2}, \hat{L}_x(t) - \hat{\sigma}k^{L,p}_{\alpha/2}\right).$$

A symmetric pivotal bootstrap CI for the log hazard is

$$\left(\hat{L}_x(t) - \hat{\sigma}\bar{k}^{L,p}_{1-\alpha}, \hat{L}_x(t) + \hat{\sigma}\bar{k}^{L,p}_{1-\alpha}\right).$$

These last two confidence intervals can be transformed back to yield confidence intervals for the hazard rate function $h_x(t)$. These are:

$$\left(\hat{h}_x(t)e^{-\hat{\sigma}k^{L,p}_{1-\alpha/2}}, \hat{h}_x(t)e^{-\hat{\sigma}k^{L,p}_{\alpha/2}}\right),$$

and

$$\left(\hat{h}_x(t)e^{-\hat{\sigma}\bar{k}^{L,p}_{1-\alpha}}, \hat{h}_x(t)e^{\hat{\sigma}\bar{k}^{L,p}_{1-\alpha}}\right)$$

respectively.

Note: The bootstrap matrix `h.hat.star.matrix` is assumed to contain estimates produced using the same time grid as `t.grid` and the same estimator used to generate `h.hat.x`.

Value

A data frame with the following columns:

`t`	The evaluation grid points.
`h.hat.x.t`	Indexed hazard rate estimator `h.hat.x`.
`CI.lower`, `CI.upper`	Lower and upper endpoints of basic pivot CIs.
`CI.sym.lower`, `CI.sym.upper`	Lower and upper endpoints of symmetric pivot CIs.

`CI.exp.lower`, `CI.exp.upper`
: Lower and upper endpoints of pivot CIs on the log-scale.

`CI.exp.sym.lower`, `CI.exp.sym.upper`
: Symmetric pivot CIs on the log-scale.

`L.hat.x.t`
: The logarithm of the indexed hazard rate estimate, `log(h.hat.x)`.

`CI.L.lower`, `CI.L.upper`
: Transformed-log CIs based on `2*log(h.hat.x) - log-quantiles`.

`CI.L.sym.lower`, `CI.L.sym.upper`
: Symmetric transformed-log CIs.

See Also

`bootstrap.h.hat.star`, `h.hat.star.x.theta`

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'



theta.1  <- 0.0702 #0.149
theta.2 <- 0.0856 #0.10
x.1 = 0 # refers to zero mean variables - slightly high
x.2 = 1.9 # refers to zero mean variable - high
b = 0.42#theta.alb * b.alb + theta.bil *b.bil # 7
x = theta.1 * x.1 + theta.2 *x.2
n.s.grid<-50

#Store original sample values:
data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )

data.id <- to.id(data)

X.1.obs=data[,X.1.name] -mean(data[, X.1.name])
X.1.i.T.i=data.id[,X.1.name] -mean(data.id[, X.1.name])
X.2.obs=data[,X.2.name]  -mean(data[, X.2.name])
X.2.i.T.i=data.id[,X.2.name] -mean(data.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)

# Calculate the indexed HQM estimator on the original sample:
h.hat.x.theta.output<- get.h.hat.x.theta(pbc2, id.name, n.s.grid,  X.obs.list, X.i.T.i.list,
                  T.name = 'years', s.name = 'year', delta.name = 'status2',
```

```
                  theta= c(theta.1,  theta.2), b, x)

h.hat.x<-h.hat.x.theta.output[,2] # Indexed HQM estimator on original sample
t.grid<-h.hat.x.theta.output[,1] # evaluation grid points
n.grid<- n.s.grid # length(h.hat.x)

#  Create bootstrap samples by group
set.seed(1)
B<- 50  #for display purposes only; for sensible results use B=1000 (slower)
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
  bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),]
}

# Simulate true hazard rate function:
h.true<- h.true.simulated(bootstrap.samples, id.name='id', n.grid,
              X.1.name=X.1.name, X.2.name= X.2.name,
              T.name = T.name, s.name = s.name,
              delta.name = delta.name, theta = c(theta.1,  theta.2), b)

# Bootstrap the original indexed HQM estimator:
h.hat.star.matrix <- bootstrap.h.hat.star(B, bootstrap.samples, X.1.name, X.2.name,
                  T.name, s.name,  delta.name, b = 0.4, x = x,
                  h.true = h.true, theta.start = c(0.07, 0.08), n.grid)

# Construct Ci's:
alpha.level<-0.05
CIs.pivotal<- bootstrap.CI.pivotal(B, n.grid, h.hat.star.matrix, t.grid, h.hat.x, alpha.level)

# extract Plain   + symmetric CIs
CI.upper<-CIs.pivotal[,"CI.upper"]
CI.lower<-CIs.pivotal[,"CI.lower"]
CI.sym.upper<-CIs.pivotal[,"CI.sym.upper"]
CI.sym.lower<-CIs.pivotal[,"CI.sym.lower"]

J <- 80 #select the first 80 grid points (for display purposes only)
#Plot the selected CIs
plot(t.grid[1:J],   h.hat.x[1:J], type="l", ylim=c(0,2), ylab="Hazard rate",
      xlab="time", lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.upper[1:J], rev(CI.lower[1:J])),
```

```
        col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.sym.lower[1:J], lty=2, lwd=2)

# extract transformed from Log HR + symmetric CIs
CI.exp.upper<-CIs.pivotal[,"CI.exp.upper"]
CI.exp.lower<-CIs.pivotal[,"CI.exp.lower"]
CI.exp.sym.upper<-CIs.pivotal[,"CI.exp.sym.upper"]
CI.exp.sym.lower<-CIs.pivotal[,"CI.exp.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J],   h.hat.x[1:J] , type="l", ylim=c(0,2), ylab="Log Hazard rate",
      xlab="time", lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.exp.upper[1:J], rev(CI.exp.lower[1:J])),
        col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.exp.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.exp.sym.lower[1:J], lty=2, lwd=2)

# extract Log HR + symmetric CIs
CI.L.upper<-CIs.pivotal[,"CI.L.upper"]
CI.L.lower<-CIs.pivotal[,"CI.L.lower"]
CI.L.sym.upper<-CIs.pivotal[,"CI.L.sym.upper"]
CI.L.sym.lower<-CIs.pivotal[,"CI.L.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J], log(h.hat.x[1:J] ), type="l", ylim=c(-5,4), ylab="Hazard rate",
      xlab="time",  lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.L.upper[1:J], rev(CI.L.lower[1:J])),
        col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.L.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.L.sym.lower[1:J], lty=2, lwd=2)
```

`bootstrap.CI.quantile` *Compute Quantile Pointwise Confidence Intervals for for the Indexed Hazard Rate Estimate*

Description

Computes quantile-bootstrap confidence intervals and its symmetric counterparts for the hazard rate, log-hazard rate, and back-transformed (from log scale) hazard rate functions, based on the indexed hazard estimator.

Usage

```
bootstrap.CI.quantile(B, n.grid, h.hat.star.matrix, t.grid, h.hat.x, alpha.level)
```

Arguments

B	Integer. Number of bootsrap replications.
n.grid	Integer. Number of estimation points at which the indexed hazard estimates and confidence intervals are evaluated.
h.hat.star.matrix	A matrix of bootstrap estimated hazard rates with dimensions n.grid × B, where each column corresponds to one bootstrap replicate.
t.grid	Numeric vector of length n.grid: the grid points at which the indexed hazard estimates and confidence intervals are calculated.
h.hat.x	Indexed hazard estimator, calculated at the grid points t.grid and using the original sample.
alpha.level	The significance level (e.g., 0.05) which will be used in computing the confidence intervals.

Details

This function computes several forms of pivot confidence intervals for indexed hazard rate estimates. Set $k_{\alpha/2} = \hat{h}_x^{[\alpha/2]}(t) - \hat{h}_x(t)$ and $k_{1-\alpha/2} = \hat{h}_x^{[1-\alpha/2]}(t) - \hat{h}_x(t)$ where $\hat{h}_x^{[\alpha/2]}(t)$ is the $\alpha/2$ quantile of $\hat{h}_x^{(j)}(t), j = 1, \ldots, B$, obtained by ordering the bootstrap estimators in ascending order and selecting the $\alpha/2$-th ordered value. For example, for $B = 1000$ bootstrap iterations and $\alpha = 0.05$, $\hat{h}_x^{[\alpha/2]}(t)$ will be the 25th smallest out of the 1000 values $\hat{h}_x^{(j)}(t), j = 1, \ldots, 1000$. Also denote with $\bar{k}_{1-\alpha}$ the $1 - \alpha$ quantile of

$$|\hat{h}_x^{(j)}(t) - \hat{h}_x(t)|, j = 1, \ldots, B.$$

Then, the quantile bootstrap CI for $\hat{h}_x(t)$ is given by

$$\left(\hat{h}_x(t) - k_{1-\alpha/2}, \hat{h}_x(t) - k_{\alpha/2}\right).$$

The symmetric quantile CI (basic CI) is

$$\left(\hat{h}_x(t) - \bar{k}_{1-\alpha}, \hat{h}_x(t) + \bar{k}_{1-\alpha}\right).$$

The confidence intervals for the logarithm of the hazard rate function are defined as follows. First set $k^L_{\alpha/2} = \hat{L}_x^{[\alpha/2]}(t) - \hat{L}_x(t)$ and $k^L_{1-\alpha/2} = \hat{L}_x^{[1-\alpha/2]}(t) - \hat{L}_x(t)$.

Also denote with $\bar{k}^L_{1-\alpha}$ the $1 - \alpha$ quantile of $|\hat{L}_x^{(j)}(t) - \hat{L}_x(t)|, j = 1, \ldots, B$. For the log hazard function $L_x(t)$ we have the quantile confidence interval is

$$\left(\hat{L}_x(t) - k^L_{1-\alpha/2}, \hat{L}_x(t) - k^L_{\alpha/2}\right).$$

The corresponding symmetric quantile CI for the log hazard is

$$\left(\hat{L}_x(t) - \bar{k}^L_{1-\alpha}, \hat{L}_x(t) + \bar{k}^L_{1-\alpha}\right).$$

These confidence intervals are transformed back to confidence intervals for the hazard rate function $h_x(t)$:

$$\left(\hat{h}_x(t)e^{-k^L_{1-\alpha/2}}, \hat{h}_x(t)e^{-k^L_{\alpha/2}}\right).$$

The corresponding symmetric confidence interval is

$$\left(\hat{h}_x(t)e^{-\bar{k}^L_{1-\alpha}}, \hat{h}_x(t)e^{\bar{k}^L_{1-\alpha}}\right).$$

Note: The bootstrap matrix `h.hat.star.matrix` is assumed to contain estimates produced using the same time grid as `t.grid` and the same estimator used to generate `h.hat.x`.

Value

A data frame with the following columns:

`t` The evaluation grid points.

`h.hat.x.t` Indexed hazard rate estimator `h.hat.x`.

`CI.lower`, `CI.upper` Lower and upper endpoints of basic quantile CIs.

`CI.sym.lower`, `CI.sym.upper` Lower and upper endpoints of symmetric quantile CIs.

`CI.exp.lower`, `CI.exp.upper` Lower and upper endpoints of quantile CIs on the log-scale.

`CI.exp.sym.lower`, `CI.exp.sym.upper` Symmetric log-scale CIs.

`L.hat.x.t` The logarithm of the indexed hazard rate estimate, `log(h.hat.x)`.

`CI.L.lower`, `CI.L.upper` Transformed-log CIs based on `2*log(h.hat.x) - log-quantiles`.

`CI.L.sym.lower`, `CI.L.sym.upper` Symmetric transformed-log CIs.

See Also

`bootstrap.h.hat.star`, `h.hat.star.x.theta`

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'


theta.1  <- 0.0702 #0.149
```

```
theta.2 <- 0.0856 #0.10
x.1 = 0 # refers to zero mean variables - slightly high
x.2 = 1.9 # refers to zero mean variable - high
b = 0.42# The result, on the indexed marker 'indmar' of
      #\code{CV.b(pbc2, 'indmar', 'years', 'year', 'status2', size.I.j=26, seq(0.2,0.4,by=0.01))}
x = theta.1 * x.1 + theta.2 *x.2
n.s.grid<-50

#Store original sample values:
data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )

data.id <- to.id(data)

X.1.obs=data[,X.1.name] -mean(data[, X.1.name])
X.1.i.T.i=data.id[,X.1.name] -mean(data.id[, X.1.name])
X.2.obs=data[,X.2.name]  -mean(data[, X.2.name])
X.2.i.T.i=data.id[,X.2.name] -mean(data.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)

# Calculate the indexed HQM estimator on the original sample:
h.hat.x.theta.output<- get.h.hat.x.theta(pbc2, id.name, n.s.grid,  X.obs.list,
                          X.i.T.i.list, T.name = 'years', s.name = 'year',
                          delta.name = 'status2',
                         theta= c(theta.1,  theta.2), b, x)

h.hat.x<-h.hat.x.theta.output[,2] # Indexed HQM estimator on original sample
t.grid<-h.hat.x.theta.output[,1] # evaluation grid points
n.grid<- n.s.grid # length(h.hat.x)

#  Create bootstrap samples by group:
set.seed(1)
B<- 50   #for display purposes only; for sensible results use B=1000 (slower)
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
  bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),]
}
```

```
# Simulate true hazard rate function:
h.true<- h.true.simulated(bootstrap.samples, id.name='id', n.grid,
              X.1.name=X.1.name, X.2.name= X.2.name,
              T.name = T.name, s.name = s.name,
              delta.name = delta.name, theta = c(theta.1, theta.2), b)

# Bootstrap the original indexed HQM estimator:
h.hat.star.matrix <- bootstrap.h.hat.star(B, bootstrap.samples, X.1.name, X.2.name, T.name,
                           s.name,  delta.name, b = 0.4, x = x, h.true = h.true,
                           theta.start = c(0.07, 0.08), n.grid   )
J <- 80
alpha.level<-0.05

# Construct Ci's:
CIs.quantile<- bootstrap.CI.quantile(B, n.grid, h.hat.star.matrix, t.grid, h.hat.x, alpha.level)

# extract Plain   + symmetric CIs and plot them:
CI.upper<-CIs.quantile[,"CI.upper"]
CI.lower<-CIs.quantile[,"CI.lower"]
CI.sym.upper<-CIs.quantile[,"CI.sym.upper"]
CI.sym.lower<-CIs.quantile[,"CI.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J],   h.hat.x[1:J], type="l", ylim=c(0,2), ylab="Hazard rate",
      xlab="time", lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.upper[1:J], rev(CI.lower[1:J])),
        col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.sym.lower[1:J], lty=2, lwd=2)

# extract transformed from Log HR + symmetric CIs
CI.exp.upper<-CIs.quantile[,"CI.exp.upper"]
CI.exp.lower<-CIs.quantile[,"CI.exp.lower"]
CI.exp.sym.upper<-CIs.quantile[,"CI.exp.sym.upper"]
CI.exp.sym.lower<-CIs.quantile[,"CI.exp.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J],   h.hat.x[1:J], type="l", ylim=c(0,2), ylab="Hazard rate",
      xlab="time", lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.exp.upper[1:J], rev(CI.exp.lower[1:J])),
        col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.exp.sym.upper[1:J], lty=2, lwd=2 )#, lwd=3
lines(t.grid[1:J],  CI.exp.sym.lower[1:J], lty=2, lwd=2)

# extract Log HR + symmetric CIs
CI.L.upper<-CIs.quantile[,"CI.L.upper"]
CI.L.lower<-CIs.quantile[,"CI.L.lower"]
CI.L.sym.upper<-CIs.quantile[,"CI.L.sym.upper"]
CI.L.sym.lower<-CIs.quantile[,"CI.L.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J],  log(h.hat.x[1:J]), type="l", ylim=c(-5,4), ylab="Log Hazard rate",
```

```
     xlab="time", lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.L.upper[1:J], rev(CI.L.lower[1:J])),
      col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.L.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.L.sym.lower[1:J], lty=2, lwd=2)
```

`bootstrap.CI.studentized`

Compute Studentized Double Bootstrap Pointwise Confidence Intervals for the Indexed Hazard Rate Estimate

---

Description

Computes bootstrap confidence intervals—studentized (double bootstrap) and its symmetric versions for the hazard rate, log-hazard rate, and back-transformed (from the log scale) hazard rate functions, based on the indexed hazard estimator.

Usage

```
bootstrap.CI.studentized(n.grid, double.bootstrap.output, t.grid, h.hat.x, alpha.level)
```

Arguments

`n.grid`	Integer. Number of estimation points at which the indexed hazard estimates and confidence intervals are evaluated.
`double.bootstrap.output`	A list of matrices of bootstrap estimated hazard and log hazard rates with dimensions `n.grid` × `B`, where each column corresponds to one bootstrap replicate.
`t.grid`	Numeric vector of length `n.grid`: the grid points at which the indexed hazard estimates and confidence intervals are calculated.
`h.hat.x`	Indexed hazard estimator, calculated at the grid points `t.grid` and using the original sample.
`alpha.level`	The significance level (e.g., 0.05) which will be used in computing the confidence intervals.

Details

This function computes several forms of studentized confidence intervals for the indexed hazard rate function. First, for each bootstrap iteration $j = 1, \ldots, B$ we construct estimators of the standard deviation, denoted by $\hat{\sigma}_j^2$ as follows: each bootstrap sample is itself bootstrapped, say B_1 times, we estimate the corresponding B_1 index parameters $\hat{\theta}_{j,k} = (\hat{\theta}_{1,j,k}, \hat{\theta}_{2,j,k}), k = 1, \ldots, B_1$, and use them to calculate the corresponding hazard estimators $\hat{h}_x^{(j,k)}(t), k = 1, \ldots, B_1$. Finally we calculate $\hat{\sigma}_j^2$ as the sample variance of $\hat{h}_x^{(j,1)}(t), \ldots, \hat{h}_x^{(j,B_1)}(t)$. Let $k^s_{\alpha/2}, k^s_{1-\alpha/2}$ and $\bar{k}^s_{1-\alpha}$ be respectively the $\alpha/2, 1-\alpha/2$ and $1-\alpha$ quantiles of

$$\frac{|\hat{h}_x^{(j)}(t) - \hat{h}_x(t)|}{\hat{\sigma}_j}, \; j = 1, \ldots, B.$$

Given the bootstrap estimators $\hat{h}_x^{(j)}(t), j = 1, \ldots, B$, the estimator of the original data set $\hat{h}_x(t)$ and the quantiles $k^s_{\alpha/2}, k^s_{1-\alpha/2}$ and $\bar{k}^s_{1-\alpha}$, the studentized (double bootstrap) confidence interval (CI) for $\hat{h}_x(t)$ is given by

$$\left(\hat{h}_x(t) - \hat{\sigma}k^s_{1-\alpha/2}, \hat{h}_x(t) - \hat{\sigma}k^s_{\alpha/2}\right).$$

The symmetric studentized confidence interval (CI) for $\hat{h}_x(t)$ defined by

$$\left(\hat{h}_x(t) - \hat{\sigma}\bar{k}^s_{1-\alpha}, \hat{h}_x(t) + \hat{\sigma}\bar{k}^s_{1-\alpha}\right).$$

For the confidence intervals for the logarithm of the hazard rate function first set $k^{L,s}_{\alpha/2}, k^{L,s}_{1-\alpha/2}$ and $\bar{k}^{L,s}_{1-\alpha}$ be the $\alpha/2, 1-\alpha/2$ and $1-\alpha$ quantile of $|\hat{L}_x^{(j)}(t) - \hat{L}_x(t)|/\hat{\sigma}_j, j = 1, \ldots, B$.

The studentized (double bootstrap) CI confidence interval for the logarithm of the hazard rate function is

$$\left(\hat{L}_x(t) - \hat{\sigma}k^{L,s}_{1-\alpha/2}, \hat{L}_x(t) - \hat{\sigma}k^{L,s}_{\alpha/2}\right).$$

Its symmetric studentized (double bootstrap) CI for the log hazard is

$$\left(\hat{L}_x(t) - \hat{\sigma}\bar{k}^{L,s}_{1-\alpha}, \hat{L}_x(t) + \hat{\sigma}\bar{k}^{L,s}_{1-\alpha}\right).$$

These confidence intervals are transformed back to yield the following confidence intervals for the hazard rate function $h_x(t)$:

$$\left(\hat{h}_x(t)e^{-\hat{\sigma}k^{L,s}_{1-\alpha/2}}, \hat{h}_x(t)e^{-\hat{\sigma}k^{L,s}_{\alpha/2}}\right),$$

and the symmetric version

$$\left(\hat{h}_x(t)e^{-\hat{\sigma}\bar{k}^{L,s}_{1-\alpha}}, \hat{h}_x(t)e^{\hat{\sigma}\bar{k}^{L,s}_{1-\alpha}}\right).$$

Note: The bootstrap matrix `studentized.h.hat.star` is assumed to contain estimates produced using the same time grid as `t.grid` and the same estimator used to generate `h.hat.x`.

Value

A data frame with the following columns:

`t`	The evaluation grid points.
`h.hat.x.t`	Indexed hazard rate estimator `h.hat.x`.
`CI.lower`, `CI.upper`	Lower and upper endpoints of basic studentized CIs.
`CI.sym.lower`, `CI.sym.upper`	Lower and upper endpoints of symmetric CIs.

`CI.L.lower`, `CI.L.upper`
: Lower and upper endpoints of studentized CIs on the log-scale.

`CI.L.sym.lower`, `CI.L.sym.upper`
: Symmetric log-scale CIs.

`L.hat.x.t`
: The logarithm of the indexed hazard rate estimate, `log(h.hat.x)`.

`CI.exp.lower`, `CI.exp.upper`
: Transformed-log CIs based on `2*log(h.hat.x) - log-quantiles`.

`CI.exp.sym.lower`, `CI.exp.sym.upper`
: Symmetric transformed-log CIs.

See Also

`bootstrap.h.hat.star`, `h.hat.star.x.theta`

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'

data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )

data.id <- to.id(data)

theta.1  <- 0.0702 #0.149
theta.2 <- 0.0856 #0.10
x.1 = 0 # refers to zero mean variables - slightly high
x.2 = 1.9 # refers to zero mean variable - high
b = 0.42#theta.alb * b.alb + theta.bil *b.bil # 7
x = theta.1 * x.1 + theta.2 *x.2
n.s.grid<-50

X.1.obs=data[,X.1.name] -mean(data[, X.1.name])
X.1.i.T.i=data.id[,X.1.name] -mean(data.id[, X.1.name])
X.2.obs=data[,X.2.name]  -mean(data[, X.2.name])
X.2.i.T.i=data.id[,X.2.name] -mean(data.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)

# Calculate the indexed HQM estimator on the original sample:
h.hat.x.theta.output<- get.h.hat.x.theta(pbc2, id.name, n.s.grid,  X.obs.list, X.i.T.i.list,
                      T.name = 'years', s.name = 'year',  delta.name = 'status2',
                      theta= c(theta.1,  theta.2), b, x)

h.hat.x<-h.hat.x.theta.output[,2] # Indexed HQM estimator on original sample
```

```
t.grid<-h.hat.x.theta.output[,1] # evaluation grid points
n.grid<- n.s.grid # length(h.hat.x)



#  Create bootstrap samples by group
set.seed(1)
B<-10 # 20 # 5 for display purposes only; for sensible results use B=200 (slower)
B1<- 10 # 20  # 5 for display purposes only; use B1=20 (slower)
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
  bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),]
}



# Simulate true hazard rate function:
h.true<- h.true.simulated(bootstrap.samples, id.name='id', n.grid,
              X.1.name=X.1.name, X.2.name= X.2.name,
              T.name = T.name, s.name = s.name,
              delta.name = delta.name, theta = c(theta.1,  theta.2), b)

# Bootstrap the original indexed HQM estimator:
double.bootstrap.output<-double.bootstrap.h.hat.star(B, B1, bootstrap.samples, X.1.name, X.2.name,
             T.name, s.name, delta.name, b, x, h.true=h.true,
            theta.start=c(theta.1,  theta.2), h.hat.x=h.hat.x, id.name= 'id', data=data)

# Construct Ci's:
alpha.level<-0.05
CIs.studentized<-bootstrap.CI.studentized(n.grid, double.bootstrap.output, t.grid, h.hat.x,
                                          alpha.level)

# extract Plain + symmetric CIs
CI.upper<-CIs.studentized[,"CI.upper"]
CI.lower<-CIs.studentized[,"CI.lower"]
CI.sym.upper<-CIs.studentized[,"CI.sym.upper"]
CI.sym.lower<-CIs.studentized[,"CI.sym.lower"]

#Plot the selected CIs
J<-80 #select the first 80 grid points (for display purposes only)
```

```
plot(t.grid[1:J], h.hat.x[1:J], type="l", ylim=c(0,2), ylab="Hazard rate", xlab="time",
           lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.upper[1:J],  rev(CI.lower[1:J])),
      col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.sym.lower[1:J], lty=2, lwd=2)

# extract transformed from Log HR + symmetric CIs
CI.L.upper<-CIs.studentized[,"CI.L.upper"]
CI.L.lower<-CIs.studentized[,"CI.L.lower"]
CI.L.sym.upper<-CIs.studentized[,"CI.L.sym.upper"]
CI.L.sym.lower<-CIs.studentized[,"CI.L.sym.lower"]

#Plot the selected CI's
plot(t.grid[1:J], log(h.hat.x[1:J]), type="l", ylim=c(-5,4), ylab="Log Hazard rate",
      xlab="time",  lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.L.upper[1:J], rev(CI.L.lower[1:J])),
      col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.L.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.L.sym.lower[1:J], lty=2, lwd=2)

# extract Log HR + symmetric CIs
CI.exp.upper<-CIs.studentized[,"CI.exp.upper"]
CI.exp.lower<-CIs.studentized[,"CI.exp.lower"]
CI.exp.sym.upper<-CIs.studentized[,"CI.exp.sym.upper"]
CI.exp.sym.lower<-CIs.studentized[,"CI.exp.sym.lower"]

#Plot the selected CIs
plot(t.grid[1:J],  h.hat.x[1:J] , type="l", ylim=c(0,2), ylab="Hazard rate", xlab="time",
      lwd=2)
polygon(x = c(t.grid[1:J], rev(t.grid[1:J])), y = c(CI.exp.upper[1:J], rev(CI.exp.lower[1:J])),
    col =  adjustcolor("red", alpha.f = 0.50), border = NA )
lines(t.grid[1:J], CI.exp.sym.upper[1:J], lty=2, lwd=2 )
lines(t.grid[1:J],  CI.exp.sym.lower[1:J], lty=2, lwd=2)
```

`bootstrap.h.hat.star` *Bootstrap Estimation of Hazard Function and Index Parameters*

Description

Performs bootstrap estimation of hazard rates and corresponding index parameters for a given set of bootstrap samples. For each bootstrap replicate, the function re-estimates the index parameters via optimisation and computes hazard estimates using `h.hat.star.x.theta`. The output is used as input in functions `bootstrap.CI.quantile` and `bootstrap.CI.pivotal`

Usage

```
bootstrap.h.hat.star(B, bootstrap.samples, X.1.name, X.2.name,T.name,
                          s.name, delta.name, b, x, h.true, theta.start, n.grid)
```

Arguments

B	Integer. Number of bootstrap iterations.
bootstrap.samples	A list of bootstrap datasets. Each element corresponds to one replicate.
X.1.name	Character string. Name of the first longitudinal marker.
X.2.name	Character string. Name of the second longitudinal marker.
T.name	Name of the event time variable in the data.
s.name	Name of the time variable for the longitudinal marker measurements.
delta.name	Name of the event indicator variable.
b	Numeric. Bandwidth parameter used in hazard estimation.
x	Numeric. Evaluation point for the conditional hazard.
h.true	Numeric vector. The true or reference hazard used in the optimisation criterion.
theta.start	Numeric vector. Starting values of the indexing parameters for the optimisation of the index coefficients.
n.grid	Integer. Number of estimation grid points for the hazard curve.

Details

For each bootstrap iteration $k = 1, \ldots, B$, the function:

1. Extracts the bootstrap sample `data.star`.
2. Computes centred marker values at the subject and observation level.
3. Estimates index parameters by minimising `Q.theta` using `optim`.
4. Computes the bootstrap hazard estimate via `h.hat.star.x.theta`.

The outputs are matrices collecting the hazard estimates and estimated index parameter vectors across bootstrap replicates.

Value

A matrix of dimension `n.grid` × `B` containing the bootstrap hazard estimates.

See Also

`h.hat.star.x.theta`, `Q.theta`, `to.id`

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'

theta.1  <- 0.0702
```

```
theta.2 <- 0.0856
x.1 = 0 # refers to zero mean variables - slightly high
x.2 = 1.9 # refers to zero mean variable - high
b = 0.42
x = theta.1 * x.1 + theta.2 *x.2

# first simulate true HR function:
data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )
data.id <- to.id(data)


#  Create bootstrap samples by group:
set.seed(1)
B<- 10 # 400 #50
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
 bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),] #data[rows.star,]
}
h.true<- h.true.simulated(bootstrap.samples, id.name='id', 100,  X.1.name=X.1.name,
                    X.2.name= X.2.name, T.name = T.name,
                    s.name = s.name,  delta.name = delta.name,
                    theta = c(theta.1,  theta.2), b)

h.hat.star.matrix <- bootstrap.h.hat.star( B,  bootstrap.samples, X.1.name, X.2.name,
          T.name,  s.name,  delta.name ,  b = 0.4, x = 1.0,
          h.true = h.true, theta.start = c(0.07, 0.08), n.grid = 100)

#return bootstrap hazard rate estimators in marix format:
h.hat.star.matrix
```

BS.hat.HQM *Brier score for the High Quality Marker estimator*

Description

Calculates the Brier score for the HQM estimator.

Usage

```
BS.hat.HQM(data, S.hat, s, t, T.name, delta.name)
```

Arguments

data	A data frame containing event times and the patient status.
S.hat	The HQM estimator values, typically the output of make.S.hat.
s	Landmark time.
t	Time horizon.
T.name	The column name of the event times in the data frame data.
delta.name	The column name of the status variable in the data frame data.

Details

The function BS.hat.HQM implements the Brier score calculation for the HQM estimator estimator.

Value

Scalar: the Brier score of the HQM estimator.

See Also

AUC.hat.HQM

Examples

```
library(pec)
library(survival)
Landmark <- 2

pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))

timesS2 <- seq(Landmark,14,by=0.5)


b=0.9
h.hat.x<- get.h.hat.x(pbcT1, 'albumin', s.grid = s.grid, T.name = 'years',
               s.name = 'year', delta.name = 'status2', 2, 0.9)
s.grid.2  = seq(Landmark, 14,  length=n.grid-1)
S.hat.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/4 , h.hat.x)


t <- 1.5
BS.hat.s.t<-BS.hat.HQM(pbcT1, S.hat.alb, Landmark,t,
                T.name = 'years', delta.name = 'status2')
BS.hat.s.t
```

c2.u.integrand *D matrix entries, used for the implementation of the local linear kernel*

Description

Calculates the entries of the D matrix in the definition of the local linear kernel

Usage

```
c2.u.integrand(b,u,v, K)
```

Arguments

u	A vector of design points where the kernel will be evaluated.
v	A vector of sample data points.
b	The bandwidth to use (a scalar).
K	The kernel function to use.

Details

Implements the caclulation of all $d \times d$ entries of matrix D, which is part of the definition of the local linear kernel. The actual calculation is performed by

$$d_{jk} = \sum_{i=1}^{n} \int_0^T K_b(x - X_i(s))\{x - X_{ij}(s)\}\{x - X_{ik}(s)\}Z_i(s)ds,$$

Value

scalar value, the result of d_{jk}.

ck.u *Local linear weight functions*

Description

Implements the weights to be used in the local linear HQM estimator.

Usage

```
c0.u(w, u, b, K)
c1.u(w, u, b, K)
c2.u(w, u, b, K)
```

Arguments

w	Sample values.
u	Grid points where the estimator will be evaluated.
b	Bandwidth parameter.
K	Kernel function.

Details

The function implements the local linear weights in the definition of the estimator $\hat{h}_x(t)$, see also h.hat.x.t

Value

A vector of $s_n(x)$ for all values x on the marker grid.

---

confidence.bands *Confidence bands*

Description

Implements the uniform and pointwise confidence bands for the future conditional hazard rate based on the last observed marker measure.

Usage

```
confidence.bands(data,   X.name, T.name = 'years',
            s.name = 'year', delta.name = 'status2', x, b)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
X.name	The column name of the marker values in the data frame data.
T.name	The column name of the event times in the data frame data.
s.name	The column name of the times the marker values were observed in the data frame data.
delta.name	The column name of the events in the data frame data.
x	Numeric value of the last observed marker value.
b	Bandwidth.

Details

The function `confidence.bands` implements the pointwise and uniform confidence bands for the estimator of the future conditional hazard rate $\hat{h}_x(t)$. The confidence bands are based on a wild bootstrap approach ${h^*}_{x_*,B}(t)$.

Pointwise: For a given $t \in (0,T)$ generate ${h^*}^{(1)}_{x_*,B}(t), ..., {h^*}^{(N)}_{x_*,B}(t)$ for $N = 1000$ and order it ${h^*}^{[1]}_{x_*,B}(t) \leq ... \leq {h^*}^{[N]}_{x_*,B}(t)$. Then

$$\hat{I}^1_{n,N} = \left[\hat{h}_{x_*}(t) - \hat{\sigma}_{G_{x_*}}(t) \frac{{h^*}^{[N(1-\frac{\alpha}{2})]}_{x_*,B}(t)}{\sqrt{n}}, \hat{h}_{x_*}(t) - \hat{\sigma}_{G_x}(t) \frac{{h^*}^{[N\frac{\alpha}{2}]}_{x_*,B}(t)}{\sqrt{n}} \right]$$

is a $1-\alpha$ pointwise confidence band for $h_{x_*}(t)$, where $\hat{\sigma}_{G_{x_*}}(t)$ is a bootrap estimate of the variance. For more details on the wild bootstrap approach, please see `wild.bootstrap.summands` and `g.hat.t.x`.

Uniform: Generate $\bar{h}^{(1)}_{x_*,B}(t), ..., \bar{h}^{(N)}_{x_*,B}(t)$ for $N = 1000$ for all $t \in [\delta_T, T - \delta_T]$ and define $W^{(i)} = \sup_{t\in[0,T]} \left|\bar{h}^{(i)}_{x_*,B}(t)\right|$ for $i = 1, ..., N$. Order $W^{[1]} \leq ... \leq W^{[N]}$. Then

$$\hat{I}^2_{n,N} = \left[\hat{h}_{x_*}(t) \pm \hat{\sigma}_{G_{x_*}}(t) \frac{W^{[N(1-\alpha)]}}{\sqrt{n}} \right]$$

is a $1-\alpha$ uniform confidence band for $h_{x_*}(t)$.

Value

A list with pointwise, uniform confidence bands and the estimator $\hat{h}_x(t)$ for all possible time points t:

`h.star.x.B` Matrix of the wild bootstrap replicates ${h^*}_{x_*,B}(t)$, one row per replicate and one column per time grid point.

`h.star.x.B.0.975`, `h.star.x.B.0.025`
The order statistics of the replicates used for the lower and upper pointwise limits.

`max.t.abs.h.star.x.B.0.95`
Maximum over t of the 0.95 order statistic of $|{h^*}_{x_*,B}(t)|$.

`var.h.star.x.B` Sample variance of the replicates at each time grid point.

`sigma.hat.G.squared`
Bootstrap variance estimate $\hat{\sigma}^2_{G_{x_*}}(t)$.

`h.hat.x.t` The estimator $\hat{h}_x(t)$.

`I.hat.1.lower`, `I.hat.1.upper`
Pointwise band $\hat{I}^1_{n,N}$.

`I.hat.max.t.lower`, `I.hat.max.t.upper`
Uniform band based on `max.t.abs.h.star.x.B.0.95`.

`I.hat.2.lower`, `I.hat.2.upper`
Uniform band $\hat{I}^2_{n,N}$.

See Also

g.hat.t.x, wild.bootstrap.summands

Examples

```
b = 10
x = 3
n.s.grid<-100
s.obs = pbc2$year
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))


bands = confidence.bands(pbc2,   'serBilir', T.name = 'years',
                    s.name = 'year', delta.name = 'status2', x, b)

J = 80
plot(s.grid[1:J], bands$h.hat.x.t[1:J], type = "l", ylim = c(0,1), ylab = 'Hazard', xlab = 'Years')

lines(s.grid[1:J], bands$I.hat.1.upper[1:J], col = "red")
lines(s.grid[1:J], bands$I.hat.1.lower[1:J], col = "red")
lines(s.grid[1:J], bands$I.hat.2.upper[1:J], col = "blue")
lines(s.grid[1:J], bands$I.hat.2.lower[1:J], col = "blue")
```

CV.b *Cross validation bandwidth selection*

Description

Implements the bandwidth selection for the future conditional hazard rate $\hat{h}_x(t)$ based on K-fold cross validation.

Usage

```
CV.b(data,   X.name, T.name = 'years',
           s.name = 'year', delta.name = 'status2', size.I.j, b.candidates)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
X.name	The column name of the marker values in the data frame data.
T.name	The column name of the event times in the data frame data.
s.name	The column name of the times the marker values were observed in the data frame data.
delta.name	The column name of the events in the data frame data.
size.I.j	Number of observations leave out for a K cross validation.
b.candidates	Vector of bandwidths that need to be tested.

Details

The function CV.b implements the cross validation bandwidth selection for the future conditional hazard rate $\hat{h}_x(t)$ given by

$$b_{CV} = argmin_b \sum_{i=1}^{N} \int_0^T \int_s^T Z_i(t) Z_i(s) (\hat{h}_{X_i(s)}(t-s) - h_{X_i(s)}(t-s))^2 dt ds,$$

where $\hat{h}_x(t)$ is a smoothed kernel density estimator of $h_x(t)$ and Z_i the exposure process of individual i. Note that $\hat{h}_x(t)$ is dependent on b.

Value

A list with the tested bandwidths and its cross validation scores.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

See Also

g.hat.i, Q1, R.hat.K, h.hat.minus.I.j, data.minus.I.j

Examples

```
size.I.j = 26
# For Albumin marker:
b.candidates = seq(0.9, 1.7, 0.1)


CV.b.alb = CV.b(pbc2,  'albumin', 'years', 'year', 'status2', size.I.j, b.candidates)
CV.b.alb[[2]][which.min(CV.b.alb[[1]])]

# For Bilirubin marker:
b.candidates = seq(3, 4, 0.1)
CV.b.bil = CV.b(pbc2, 'serBilir', 'years', 'year', 'status2', size.I.j, b.candidates)
CV.b.bil[[2]][which.min(CV.b.bil[[1]])]
CV.b.bil
```

CV.theta *Cross validation index parameter selection*

Description

Implements the index parameter selection for two markers based on K-fold cross validation.

Usage

```
CV.theta(theta, data, X.1.name, X.2.name,
         T.name = 'years', s.name = 'year', delta.name = 'status2', size.I.j, b)
```

Arguments

theta	Vector of candidate values for the index parameters.
data	A data frame of time dependent data points. Missing values are allowed.
X.1.name	The column name of the first marker values in the data frame data.
X.2.name	The column name of the second marker values in the data frame data.
T.name	The column name of the event times in the data frame data.
s.name	The column name of the times the marker values were observed in the data frame data.
delta.name	The column name of the events in the data frame data.
size.I.j	Number of observations leave out for a K cross validation.
b	scalar bandwidth for the HQM estimator.

Details

The function CV.theta implements the cross validation index parameter selection for the indexing of two markers and given by

$$\hat{\theta} = argmin_{\theta_1,\theta_2} \sum_{i=1}^{N} \int_0^T \int_s^T Z_i(t) Z_i(s) (\hat{h}_{\theta_0^T X_i(s)}(t-s) - h_{\theta_0^T X_i(s)}(t-s))^2 dt ds,$$

where $\hat{h}_x(t)$ is the HQM estimator of $h_x(t)$, see Bagkavos et al. (2025), doi:10.1093/biomet/asaf008, and Z_i the exposure process of individual i. Note that $\hat{h}_x(t)$ is dependent on b.

Value

A list with the tested parameters and its cross validation scores.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

See Also

g.hat.i, Q1, R.hat.K, h.hat.minus.I.j, data.minus.I.j

Examples

```
# Obtain the indexing parameters for the combination of albumin and bilirubin markers
# These are the values used in the example of function h.hat.x.t.vec
# and yielded the values:  (theta.alb, theta.bil) = ( 0.0702, 0.0856 ) that were used there.

size.I.j = 26
b.candidates = seq(1.1, 1.2, by=0.1)

for(i in 1:length(b.candidates))
{
  theta.fit<- optim(c(0.5, 0.5), CV.theta,  data=pbc2, X.1.name='albumin',
              X.2.name= 'serBilir', T.name = 'years', s.name = 'year',
           delta.name = 'status2', size.I.j=26, b=b.candidates[i], method="Nelder-Mead")
  cat("i= ", i, " ", theta.fit$par, " ", theta.fit$value, "count calls to fn = ",
      theta.fit$counts, " converge? ", theta.fit$convergence,   "\n")
  theta.fit
}
```

`data.minus.I.j` *Split dataset for K-fold cross validation*

Description

Creates multiple splits of a dataset which is then used in the bandwidth selection with K-fold cross validation.

Usage

```
data.minus.I.j(size.I.j, data)
```

Arguments

`data`	A data frame of time dependent data points. Missing values are allowed.
`size.I.j`	The number of individuals that should be left out. Optimally, $K = n/I$ should be an integer, where n is the number of individuals.

Details

The function `data.minus.I.j` takes a data frame and transforms it into $K = n/I$ data frames with I individuals missing from each data frame. Let I_j be sets of indices with $\cup_{j=1}^{K} I_j = \{1, ..., n\}$, $I_k \cap I_j = \emptyset$ and $|I_j| = |I_k| = I$ for all $j, k \in \{1, ..., K\}$. Then data frames with $\{1, ..., n\}/I_j$ individuals are created.

Value

A list of data frames with `size.I.j` individuals missing in the above way.

See Also

CV.b

Examples

```
data.minus.I.j.list = data.minus.I.j(26, pbc2)
```

---

double.bootstrap.h.hat.star
Bootstrap Estimation of Hazard Function and Index Parameters

Description

Performs bootstrap estimation of hazard rates and their standard deviation at each grid point (double boostrap) together with the corresponding index parameters for a given set of bootstrap samples. The output of the function is used as input in bootstrap.CI.studentized.

Usage

```
double.bootstrap.h.hat.star(B, B1, bootstrap.samples, X.1.name, X.2.name,
                            T.name, s.name, delta.name, b, x, h.true,
                            theta.start,  h.hat.x, id.name, data)
```

Arguments

B	Integer. Number of bootstrap samples.
B1	Integer. Number of bootstrap re-samples.
bootstrap.samples	A list of bootstrap datasets. Each element corresponds to one replicate.
X.1.name	Character string. Name of the first longitudinal marker.
X.2.name	Character string. Name of the second longitudinal marker.
T.name	Name of the event time variable in the data.
s.name	Name of the time variable for the longitudinal marker measurements.
delta.name	Name of the event indicator variable.
b	Numeric. Bandwidth parameter used in hazard estimation.
x	Numeric. Evaluation point for the conditional hazard.
h.true	Numeric vector. The true or reference hazard used in the optimisation criterion.
theta.start	Numeric vector. Starting values of the indexing parameters for the optimisation of the index coefficients.
h.hat.x	HQM estimator on the original sample.
id.name	label of id variable of dataset.
data	original sample.

Details

For each bootstrap iteration $k = 1, \ldots, B$, the function:

1. Extracts the bootstrap sample `data.star`.
2. Computes centred marker values at the subject and observation level.
3. Estimates index parameters by minimising `Q.theta` using `optim`.
4. Computes the bootstrap hazard estimate via `h.hat.star.x.theta`.

The outputs are matrices collecting the hazard estimates and estimated index parameter vectors across bootstrap replicates.

Value

A list of matrices of dimension `n.grid` × `B` containing the bootstrap hazard estimates, the logarithm of the hazard rate estimates and and two vectors of the estimate's standard deviations at each grid point.

See Also

`h.hat.star.x.theta`, `Q.theta`, `to.id`

Examples

```
# See the example for  function: bootstrap.CI.studentized
```

`g.hat.i` *Preparations for bandwidth selection*

Description

Calculates an intermediate part for the K-fold cross validation.

Usage

```
g.hat.i(h.hat.matrix, X.pos, n.s.grid, n, Z.i.ds)
```

Arguments

`h.hat.matrix`	A matrix of the estimator for the future conditional hazard rate for all values `x` and `t`.
`X.pos`	Vector of the position of the observed marker values in the grid for marker values.
`n.s.grid`	Numeric value indicating the number of grid points of the time grid.
`n`	Number of individuals.
`Z.i.ds`	A matrix made by `make.Z.i.ds` indicating the exposure.

Details

The function `g.hat.i` calculates a key component for the bandwidth selection

$$\hat{g}_i^{-I_j}(t) = \int_0^t Z_i(s)\hat{h}_{X_i(s)}^{-I_j}(t-s)ds,$$

where $\hat{h}^{-I_j}$ is estimated without information from all counting processes i with $i \in I_j$ and Z is the exposure.

Value

A matrix with $\hat{g}_i^{-I_j}(t)$ for all individuals `i` and time grid points `t`.

See Also

`CV.b`

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid,
            s.pos, X.pos, 'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos,
              X.pos,'years', n)
dN.i  <- make.dN.i(s.grid=s.grid, n.s.grid, T.i, delta.i, n)

t = 2

h.hat.matrix = t(sapply(s.grid[1:99], function(t.point){
          h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, t, b, Z.i.ds, X.pos, n)}))

g.hat.i(h.hat.matrix, X.pos, n.X.grid, n, Z.i.ds)
```

`g.hat.t.x` *Computation of a key component for wild bootstrap*

Description

Implements a key part for the wild bootstrap of the hqm estimator.

Usage

```
g.hat.t.x(X.grid, s.grid, n.s.grid, X.pos, x, t, b, Z.i.ds, Z.ds, n)
```

Arguments

`X.grid`	Marker value grid points that will be used in the evaluatiuon.
`s.grid`	Time value grid points that will be used in the evaluatiuon.
`n.s.grid`	Size of the time grid.
`X.pos`	Position of the linear interpolated marker values on the marker grid.
`x`	Numeric value of the last observed marker value.
`t`	Numeric value of the time the function should be evaluated.
`b`	Bandwidth.
`Z.i.ds`	A matrix made by `make.Z.i.ds` indicating the exposure.
`Z.ds`	A matrix made by `make.Z.ds` indicating the exposure.
`n`	Number of individuals.

Details

The function implements

$$\hat{g}_{t,x}(z) = \frac{1}{n}\sum_{j=1}^{n}\int_0^{T-t} \hat{E}(X_j(t+s))^{-1} K_b(z, X_j(t+s)) Z_j(t+s) Z_j(s) K_b(x, X_j(s)) ds,$$

for every value z on the marker grid, where $\hat{E}(x) = \frac{1}{n}\sum_{j=1}^{n}\int_0^T K_b(x, X_j(s)) Z_j(s) ds$, Z the exposure and X the marker.

Value

A vector of $\hat{g}_{t,x}(z)$ for all values z on the marker grid.

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
X.obs = pbc2$serBilir
s.obs = pbc2$year
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

Z.i.ds<-make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
            'years', n)
Z.ds<-make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid,n.s.grid,n.X.grid, s.pos,X.pos,
            'years', n)

t = 2
x = 2
b = 10

g.hat.t.x(X.grid, s.grid, n.s.grid, X.pos, x, t, b, Z.i.ds, Z.ds, n)
```

---

get.alpha.hat *Marker-only hazard rate*

Description

Calculates the marker-only hazard rate for time dependent data.

Usage

```
get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )
```

Arguments

dN	A matrix made by make.dN indicating the occurences of events.
Z.ds	A matrix made by make.Z.ds indicating the exposure.
b	Bandwidth.
X.grid	Vector of grid points for the marker values X.
K	Used kernel function.

Details

The function `get.alpha.hat` implements the marker-only hazard estimator

$$\hat{\alpha}_i(z) = \frac{\sum_{k \neq i} \int_0^T K_{b_1}(z - X_k(s)) \mathrm{d}N_k(s)}{\sum_{k \neq i} \int_0^T K_{b_1}(z - X_k(s)) Z_k(s) \mathrm{d}s},$$

where X is the marker and Z is the exposure. The marker-only hazard is defined as the underlying hazard which is not dependent on time

$$\alpha(X(t), t) = \alpha(X(t)).$$

Value

A vector of marker-only values for `X.grid`.

See Also

`h.hat.x.t`

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN  <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid,
            n.X.grid, s.pos, X.pos, T.name = 'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )
```

`get.h.hat.x` *Local constant future conditional hazard rate estimator*

Description

Calculates the (indexed) local constant future hazard rate function, conditional on a marker value `x`, across across a set of time values `t`.

Usage

```
get.h.hat.x(data, X.name, s.grid, T.name, s.name, delta.name, x, b)
```

Arguments

`data`	A data frame of time dependent data points. Missing values are allowed.
`X.name`	The column name of the marker values in the data frame `data`.
`s.grid`	User defined grid mesh on s.name variable
`T.name`	The column name of the event times in the data frame `data`.
`s.name`	The column name of the times the marker values were observed in the data frame `data`.
`delta.name`	The column name of the events in the data frame `data`.
`x`	Numeric value of the last observed marker value.
`b`	Bandwidth parameter.

Details

The function `get.h.hat.x` implements the indexed local linear future conditional hazard estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(\hat{\theta}^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s},$$

across a grid of possible time values t, where, for a positive integer p, $\hat{\theta}^T = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ is the vector of the estimated indexing parameters, $X_i = (X_{1,i}, \ldots, X_{i,p})$ is a vector of markers for indexing, Z_i is the exposure and $\alpha(z)$ is the marker-only hazard, see `get.alpha.hat` for more details and $K_b = b^{-1}K(./b)$ is an ordinary kernel function, e.g. the Epanechnikov kernel. For $p = 1$ and $\hat{\theta} = 1$ the above estimator becomes the HQM hazard rate estimator conditional on one covariate,

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

defined in equation (2) of Bagkavos et al. (2025), doi:10.1093/biomet/asaf008.

Value

A vector of $\hat{h}_x(t)$ for a grid of possible time values t.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

See Also

get.alpha.hat, h.hat.x.t, get.h.tilde.x

Examples

```
library(survival)
b = 10
x = 3
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]

n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))

b=0.9

h.tilde.x<-get.h.tilde.x(pbcT1,'albumin', s.grid = s.grid, T.name='years',
                  s.name='year',delta.name='status2',2,0.9)
h.hat.x<-get.h.hat.x(pbcT1,'albumin', s.grid = s.grid, T.name='years',
                  s.name='year',delta.name='status2',2,0.9)

#Caclulate the local contant and local linear survival functions
s.grid.2  = seq(Landmark, 14,  length= n.grid-1 )
S.tilde.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/4 , h.tilde.x)
S.hat.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/4 , h.hat.x)

#For comparison, also calculate the Kaplan-Meier
kma2<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

#Plot the survival functions:
plot(s.grid.2, S.tilde.alb,  type="l", col=1, lwd=2, ylab="Survival probability",
     xlab="Marker level")
lines(s.grid.2, S.hat.alb,  lty=2, lwd=2, col=2)
lines(kma2$time, kma2$surv, type="s",  lty=2, lwd=2, col=3)

legend("topright", c(  "Local linear HQM", "Local constant HQM",
        "Kaplan-Meier"), lty=c(1, 2, 2), col=1:3, lwd=2, cex=1.7)


## Not run:
#Example of get.h.hat.x with two indexed covariates:
#First, estimate the joint model for Albumin and Bilirubin combined:
library(JM)
lmeFit <- lme(albumin + serBilir~ year, random = ~ year | id, data = pbc2)
coxFit <- coxph(Surv(years, status2) ~ albumin + serBilir, data = pbc2.id, x = TRUE)
jointFit <- jointModel(lmeFit, coxFit, timeVar = "year", method = "piecewise-PH-aGH")
```

```
Landmark <- 1
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]


n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))


# Index Albumin and Bilirubin:
x.alb = 3  # slightly low albumin value
x.bil = 1.9  # slightly  high bilirubin value

theta.alb  <- 0.0702
theta.bil <- 0.0856
theta.T.X.obs = theta.alb * pbcT1$albumin + theta.bil *pbcT1$serBilir
 # theta.T.X.obs is now the indexed marker
x = theta.alb * x.alb + theta.bil *x.bil #conditioning value
## store the combine variable in place of 'drug' column which is redundant
pbcT1$drug<- theta.T.X.obs
## i.e. 'drug' corresponds to indexed bilirubin and albumin

timesS2 <- seq(Landmark,14,by=0.5)
predT1 <- survfitJM(jointFit, newdata = pbcT1, survTimes = timesS2)
nm<-length(predT1$summaries)

mat.out1<-matrix(nrow=length(timesS2), ncol=nm)
for(r in 1:nm)
{
  SurvLand <- predT1$summaries[[r]][,"Mean"][1] #obtain mean predictions
  mat.out1[,r] <- predT1$summaries[[r]][,"Mean"]/SurvLand
}
JM.surv.est<-rowMeans(mat.out1, na.rm=TRUE) #average the resulting JM estimates

# calculate indexed local linear HQM estimator for bilirubin and albumin
b.alb = 1.5
b.bil = 4
b.hqm   =  theta.alb * b.alb + theta.bil *b.bil # bandwidth for HQM estimator
h.hat.x<- get.h.hat.x(pbcT1, 'drug', s.grid  =s.grid, T.name = 'years',  s.name = 'year',
                delta.name = 'status2', x, b.hqm)
s.grid.2  = seq(Landmark, 14,  length=n.grid-1) #grid points for HMQ estimator
S.hat.x<- make.S.hat((s.grid.2[2]-s.grid.2[1])/5, h.hat.x) # transform HR to Survival func.

km.land<- survfit(Surv(years , status2) ~ 1, data = pbcT1) #KM estimate

#Plot the survival functions:
plot(s.grid.2, S.hat.x, type="l", ylim=c(0,1), xlim=c(Landmark,14),
            ylab="Survival probability", xlab="years",lwd=2)
lines(timesS2, JM.surv.est, col=2,  lwd=2, lty=2)
lines(km.land$time, km.land$surv, type="s",lty=2, lwd=2, col=3)
legend("bottomleft", c("HQM est.", "Joint Model est.", "Kaplan-Meier"),
      lty=c(1,2,2),  col=1:3, lwd=2, cex=1.7)
```

```
## End(Not run)
```

`get.h.hat.x.theta` *Local linear future conditional hazard estimator (wrapper)*

Description

Compute the indexed local linear future conditional hazard rate estimator on a grid.

Usage

```
get.h.hat.x.theta(data, id.name, n.grid,  X.obs.list, X.i.T.i.list, T.name = 'years',
                          s.name = 'year', delta.name = 'status2', theta, b, x)
```

Arguments

`data`	Data frame with longitudinal observations.
`id.name`	Name of id column (string).
`n.grid`	user supplied grid length on the s.name argument.
`X.obs.list`	List of vectors for indexing: each vector corresponds to a single biomarker and contains its longitudinal measurements across all individuals/time points.
`X.i.T.i.list`	List of vectors for indexing: each vector corresponds to a biomarker and contains one summary measurement per individual.
`T.name`	Name of event time column.
`s.name`	Name of observation time column.
`delta.name`	Name of event indicator column.
`theta`	Numeric vector length 2 with indexing parameters.
`b`	Bandwidth parameter.
`x`	conditioning parameter.

Value

A data frame with columns `t` and `h.hat.x.t`.

Examples

```
b.alb = 0.9
b.bil = 4

x.alb = 1 # refers to zero mean variables - slightly high
x.bil = 1.9 # refers to zero mean variable - high
```

```
theta.alb  <- 0.0702 #0.149
theta.bil <- 0.0856 #0.10


b = 0.42
x = x.alb * theta.alb + x.bil *theta.bil

X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'
n.s.grid<-50

data.use<-pbc2
data.use.id<-to.id(data.use)
#data.use.id<-data.use.id[complete.cases(data.use.id), ]

# mean adjust the data:
X.1.obs=data.use[,X.1.name] -mean(data.use[, X.1.name])
X.1.i.T.i=data.use.id[,X.1.name] -mean(data.use.id[, X.1.name])
X.2.obs=data.use[,X.2.name]  -mean(data.use[, X.2.name])
X.2.i.T.i=data.use.id[,X.2.name] -mean(data.use.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)

h.hat.x.theta.output<-get.h.hat.x.theta(pbc2, id.name, n.s.grid,  X.obs.list,
                                        X.i.T.i.list, T.name = 'years',
    s.name = 'year',  delta.name = 'status2', theta= c(theta.alb,  theta.bil), b, x)
h.hat.x<-h.hat.x.theta.output[,2]
t.grid<-h.hat.x.theta.output[,1]
n.grid<-length(h.hat.x)
```

`get.h.tilde.x` *Local linear future conditional hazard rate estimator*

Description

Calculates the (indexed) local linear future hazard rate function, conditional on a marker value `x`, across a set of time values `t`.

Usage

```
get.h.tilde.x(data, X.name, s.grid, T.name, s.name, delta.name, x, b)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
X.name	The column name of the marker values in the data frame data.
s.grid	User defined grid mesh on s.name variable
T.name	The column name of the event times in the data frame data.
s.name	The column name of the times the marker values were observed in the data frame data.
delta.name	The column name of the events in the data frame data.
x	Numeric value of the last observed marker value.
b	Bandwidth parameter.

Details

The function get.h.tilde.x uses a local linear kernel to implement the indexed local linear future conditional hazard estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(\hat{\theta}^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s},$$

across a grid of possible time values t, where, for a positive integer p, $\hat{\theta}^T = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ is the vector of the estimated indexing parameters, $X_i = (X_{1,i}, \ldots, X_{i,p})$ is a vector of markers for indexing, Z_i is the exposure and $\alpha(z)$ is the marker-only hazard, see get.alpha.hat for more details. For $p = 1$ and $\hat{\theta} = 1$ the above estimator becomes the HQM hazard rate estimator conditional on one covariate,

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

defined in equation (2) of Bagkavos et al. (2025) doi:10.1093/biomet/asaf008. In the place of $K_b()$, get.h.tilde.x uses the kernel

$$K_{x,b}(u) = \frac{K_b(u) - K_b(u) u^T D^{-1} c_1}{c_0 - c_1^T D^{-1} c_1},$$

where $K_b() = b^{-1} K(./b)$ with K being an ordinary kernel, e.g. the Epanechnikov kernel, $c_1 = (c_{11}, \ldots, c_{1d})^T$, $D = (d_{ij})_{(d+1)\times(d+1)}$ with

$$c_0 = \sum_{i=1}^{n} \int_0^T K_b(x - \hat{\theta}^T X_i(s)) Z_i(s) ds,$$

$$c_{ij} = \sum_{i=1}^{n} \int_0^T K_b(x - \hat{\theta}^T X_i(s)) \{x - \hat{\theta}^T X_{ij}(s)\} Z_i(s) ds,$$

$$d_{jk} = \sum_{i=1}^{n} \int_0^T K_b(x - \hat{\theta}^T X_i(s)) \{x - \hat{\theta}^T X_{ij}(s)\} \{x - \hat{\theta}^T X_{ik}(s)\} Z_i(s) ds,$$

see also Nielsen (1998), doi:10.1080/03461238.1998.10413997.

Value

A vector of $\hat{h}_x(t)$ for a grid of possible time values t.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

Nielsen (1998), Marker dependent kernel hazard estimation from local linear estimation, Scandinavian Actuarial Journal, pp. 113-124. doi:10.1080/03461238.1998.10413997

See Also

get.alpha.hat, h.hat.x.t, get.h.hat.x

Examples

```
library(survival)
library(JM)

# Compare Local constant and local linear estimator for a single covariate,
# use KM for reference.
# Albumin marker, use landmarking
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
b=0.9

n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))


h.tilde.x<-get.h.tilde.x(pbcT1, 'albumin', s.grid = s.grid, T.name = 'years', s.name = 'year',
                  delta.name = 'status2', 2, 0.9)
h.hat.x<-get.h.hat.x(pbcT1, 'albumin', s.grid = s.grid, T.name = 'years', s.name = 'year',
                delta.name = 'status2', 2, 0.9)

#Calculate the local contant and local linear survival functions
s.grid.2  = seq(Landmark, 14,  length=n.grid-1)
S.tilde.alb<- make.S.hat((s.grid.2[2]-s.grid.2[1])/4 , h.tilde.x)
S.hat.alb<- make.S.hat((s.grid.2[2]-s.grid.2[1])/4 , h.hat.x)

#For comparison, also calculate the Kaplan-Meier
kma2<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

#Plot the survival functions:
plot(s.grid.2, S.tilde.alb,  type="l", col=1, lwd=2, ylab="Survival probability",
                                                      xlab="Marker level")
lines(s.grid.2, S.hat.alb,  lty=2, lwd=2, col=2)
lines(kma2$time, kma2$surv, type="s",  lty=2, lwd=2, col=3)
legend("topright", c(  "Local linear HQM", "Local constant HQM", "Kaplan-Meier"),
        lty=c(1, 2, 2), col=1:3, lwd=2, cex=1.7)
```

```
## Not run:
#Example of get.h.tilde.x with a single covariate (no indexing):
#Compare JM, HQM and KM for Bilirubin
b = 10
Landmark <- 1
lmeFit <- lme(serBilir ~ year, random = ~ year | id, data = pbc2)
coxFit <- coxph(Surv(years, status2) ~ serBilir, data = pbc2.id, x = TRUE)

jointFit0 <- jointModel(lmeFit, coxFit, timeVar = "year",
                                method = "piecewise-PH-aGH")
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))

timesS1 <- seq(1,14,by=0.5)
predT1 <- survfitJM(jointFit0, newdata = pbcT1,survTimes = timesS1)
nm<-length(predT1$summaries)

mat.out1<-matrix(nrow=length(timesS1), ncol=nm)
for(r in 1:nm)
{
  SurvLand <- predT1$summaries[[r]][,"Mean"][1]
  mat.out1[,r] <- predT1$summaries[[r]][,"Mean"]/SurvLand
}
sfit1y<-rowMeans(mat.out1, na.rm=TRUE)

h.tilde.x<- get.h.tilde.x(pbcT1, 'serBilir', s.grid= s.grid,  T.name = 'years',
        s.name = 'year', delta.name = 'status2', 1, 10)
s.grid.1  = seq(Landmark, 14,  length=n.grid-1)
S.tilde.bil<- make.S.hat((s.grid.1[2]-s.grid.1[1])/5.4 , h.tilde.x)
kma1<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

plot(s.grid.1, S.tilde.bil, type="l", ylim=c(0,1), xlim=c(Landmark,14),
                   ylab="Survival probability", xlab="years",lwd=2)
lines(timesS1, sfit1y, col=2,  lwd=2, lty=2)
lines(kma1$time, kma1$surv, type="s", lty=2, lwd=2, col=3 )
legend("bottomleft", c("HQM est.", "Joint Model est.", "Kaplan-Meier"),
                                  lty=c(1,2,2), col=1:3, lwd=2, cex=1.7)

#Example of get.h.tilde.x with two indexed covariates:

#First, estimate the joint model for Albumin and Bilirubin combined:
lmeFit <- lme(albumin + serBilir~ year, random = ~ year | id, data = pbc2)
coxFit <- coxph(Surv(years, status2) ~ albumin + serBilir, data = pbc2.id,
                                                          x = TRUE)
jointFit <- jointModel(lmeFit, coxFit, timeVar = "year",
                                            method = "piecewise-PH-aGH")

Landmark <- 1
pbcT1 <- pbc2[which(pbc2$year< Landmark  & pbc2$years> Landmark),]
```

```
n.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- pbcT1[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.grid-1))


# Index Albumin and Bilirubin:
x.alb = 3  # slightly low albumin value
x.bil = 1.9  # slightly  high bilirubin value

theta.alb  <- 0.0702
theta.bil <- 0.0856
# theta.T.X.obs is now the indexed marker
theta.T.X.obs = theta.alb * pbcT1$albumin + theta.bil *pbcT1$serBilir
x = theta.alb * x.alb + theta.bil *x.bil #conditioning value

pbcT1$drug<- theta.T.X.obs ## store theta.T.X.obs in place of 'drug' column which is redundant here
## i.e. 'drug' corresponds to indexed bilirubin and albumin

timesS2 <- seq(Landmark,14,by=0.5)
predT1 <- survfitJM(jointFit, newdata = pbcT1,survTimes = timesS2)
nm<-length(predT1$summaries)

mat.out1<-matrix(nrow=length(timesS2), ncol=nm)
for(r in 1:nm)
{
  SurvLand <- predT1$summaries[[r]][,"Mean"][1] #obtain mean predictions
  mat.out1[,r] <- predT1$summaries[[r]][,"Mean"]/SurvLand
}
JM.surv.est<-rowMeans(mat.out1, na.rm=TRUE) #average the resulting JM estimates

# calculate indexed local linear HQM estimator for bilirubin and albumin
b.alb = 1.5
b.bil = 4
b.hqm   =  theta.alb * b.alb + theta.bil *b.bil # bandwidth for HQM estimator
h.tilde.x<- get.h.tilde.x(pbcT1, 'drug', s.grid = s.grid, T.name = 'years',  s.name = 'year',
                                                delta.name = 'status2', x, b.hqm)
s.grid.2  = seq(Landmark, 14,  length=n.grid-1) #grid points for HMQ estimator
S.tilde.x<- make.S.hat((s.grid.2[2]-s.grid.2[1])/5 ,h.tilde.x) # transform HR to Survival func.

km.land<- survfit(Surv(years , status2) ~ 1, data = pbcT1) #KM estimate

#Plot the survival functions:
plot(s.grid.2, S.tilde.x, type="l", ylim=c(0,1), xlim=c(Landmark,14),
    ylab="Survival probability", xlab="years",lwd=2)
lines(timesS2, JM.surv.est, col=2,  lwd=2, lty=2)
lines(km.land$time, km.land$surv, type="s",lty=2, lwd=2, col=3)
legend("bottomleft", c("HQM est.", "Joint Model est.", "Kaplan-Meier"),
        lty=c(1,2,2),  col=1:3, lwd=2, cex=1.7)

## End(Not run)
```

`h.hat.minus.I.j` *Prepare for Cross validation bandwidth selection*

Description

Implements the calculation of the hqm estimator on cross validation data sets. This is a preparation for the cross validation bandwidth selection technique for future conditional hazard rate estimation based on marker information data.

Usage

```
h.hat.minus.I.j(data, data.id, X.name, T.name = 'years',
        s.name = 'year',delta.name = 'status2', n, size.I.j, b)
```

Arguments

`data`	A data frame of time dependent data points. Missing values are allowed.
`data.id`	An id data frame obtained from `to.id`.
`X.name`	The column name of the marker values in the data frame `data`.
`T.name`	The column name of the event times in the data frame `data`.
`s.name`	The column name of the times the marker values were observed in the data frame `data`.
`delta.name`	The column name of the events in the data frame `data`.
`n`	Number of individuals.
`size.I.j`	Number of observations leave out for a K cross validation.
`b`	Bandwidth.

Details

The function splits the data set via `data.minus.I.j` and calculates for every splitted data set the hqm estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

for all x on the marker grid and t on the time grid, where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see `get.alpha.hat` for more details.

Value

A list of matrices for every cross validation data set with $\hat{h}_x(t)$ for all x on the marker grid and t on the time grid.

See Also

`CV.b`

Examples

```
data.id = to.id(pbc2)
n = max(as.numeric(pbc2$id))
b = 1.5
size.I.j = 26
h.hat.minus.I.j.list = h.hat.minus.I.j(pbc2, data.id, 'serBilir', 'years', 'year', 'status2',
                                       n, size.I.j, b)
```

`h.hat.minus.I.j.theta` *Prepare for Cross validation index parameter selection*

Description

Implements the calculation of the hqm estimator on cross validation data sets. This is a preparation for the cross validation index selection technique for future conditional hazard rate estimation based on marker information data.

Usage

```
h.hat.minus.I.j.theta(theta, data, data.id, X.1.name, X.2.name, T.name = 'years',
        s.name = 'year',delta.name = 'status2', n, size.I.j, b)
```

Arguments

`theta`	Vector of candidate indexing values.
`data`	A data frame of time dependent data points. Missing values are allowed.
`data.id`	An id data frame obtained from `to.id`.
`X.1.name`	The column name of the marker values in the data frame `data`.
`X.2.name`	The column name of the marker values in the data frame `data`.
`T.name`	The column name of the event times in the data frame `data`.
`s.name`	The column name of the times the marker values were observed in the data frame `data`.
`delta.name`	The column name of the events in the data frame `data`.
`n`	Number of individuals.
`size.I.j`	Number of observations leave out for a K cross validation.
`b`	Bandwidth.

Details

The function splits the data set via `data.minus.I.j` and calculates for every splitted data set the hqm estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^{n} \int_0^T \hat{\alpha}_i(\theta_0^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \theta_0^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^{n} \int_0^T Z_i(t+s) Z_i(s) K_b(x - \theta_0^T X_i(s)) \mathrm{d}s},$$

for all x on the marker grid and t on the time grid, where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see `get.alpha.hat` for more details.

Value

A list of matrices for every cross validation data set with $\hat{h}_x(t)$ for all x on the marker grid and t on the time grid.

See Also

CV.theta

---

h.hat.star.x.theta *Indexed HQM hazard estimator for one bootstrap sample*

Description

Compute the bootstrap estimate of the indexed HQM hazard estimate for a single (bootstrap) sample. The function is meant to be used internally for the calculation of confidence intervals

Usage

```
h.hat.star.x.theta(theta, data, data.id, n.grid, X.obs.list, X.i.T.i.list,
                   T.name = 'years', s.name = 'year', delta.name = 'status2', b, x)
```

Arguments

theta	Numeric vector, the values of the indexing parameters.
data	Bootstrap data.frame.
data.id	Id-level data.frame (result of to.id).
n.grid	user supplied grid length on the s.name argument.
X.obs.list	List of vectors for indexing: each vector corresponds to a single biomarker and contains its longitudinal measurements across all individuals/time points.
X.i.T.i.list	List of vectors for indexing: each vector corresponds to a biomarker and contains one summary measurement per individual.
T.name	Name of event time column.
s.name	Name of observation time column.
delta.name	Name of event indicator column.
b	Bandwidth parameter.
x	Conditioning level.

Value

Numeric vector with estimated hazard on the grid.

Examples

```
#Single instance of the bootstrap version of the bootstrap version
#of the indexed HQM estimator

b.alb = 0.9
b.bil = 4

x.alb = 1 # refers to zero mean variables - slightly high
x.bil = 1.9 # refers to zero mean variable - high

theta.alb  <- 0.0702 #0.149
theta.bil <- 0.0856 #0.10


b = 0.42 # The result, on the indexed marker 'indmar' of
      #\code{CV.b(pbc2, 'indmar', 'years', 'year', 'status2', size.I.j=26, seq(0.2,0.4,by=0.01))}
x = x.alb * theta.alb + x.bil *theta.bil

X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'
n.s.grid<-50

data.use<-pbc2
data.use.id<-to.id(data.use)
data.use.id<-data.use.id[complete.cases(data.use.id), ]

# mean adjust the data:
X.1.obs=data.use[,X.1.name] -mean(data.use[, X.1.name])
X.1.i.T.i=data.use.id[,X.1.name] -mean(data.use.id[, X.1.name])
X.2.obs=data.use[,X.2.name]  -mean(data.use[, X.2.name])
X.2.i.T.i=data.use.id[,X.2.name] -mean(data.use.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)
h.hat.star<-h.hat.star.x.theta (c(theta.alb,theta.bil), data.use, data.use.id, n.grid=n.s.grid,
                            X.obs.list, X.i.T.i.list, T.name = 'years', s.name = 'year',
                               delta.name = 'status2', b, x)
```

---

h.hat.x.t *Local constant future conditional hazard rate estimation at a single time point*

Description

Calculates the (indexed) future conditional hazard rate for a marker value `x` and a time value `t`.

Usage

```
h.hat.x.t(X.grid, s.grid, X.pos, n.s.grid, alpha.hat, x,t, b, Z.i.ds,n)
```

Arguments

X.grid	Vector of grid points for the marker values X.
s.grid	Vector of grid points for the time values s.
X.pos	Position of the linear interpolated marker values on the marker grid.
n.s.grid	Size of the time grid.
alpha.hat	Marker-hazard obtained from get.alpha.hat.
x	Numeric value of the last observed marker value.
t	Numeric time value.
b	Bandwidth.
Z.i.ds	A matrix made by make.Z.i.ds indicating the exposure.
n	Number of individuals.

Details

Function h.hat.x.t implements the future conditional hazard estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(\hat{\theta}^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s},$$

where, for a positive integer p, $\hat{\theta}^T = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ is the vector of the estimated indexing parameters, $X_i = (X_{1,i}, \ldots, X_{i,p})$ is a vector of markers for indexing, Z_i is the exposure and $\alpha(z)$ is the marker-only hazard, see get.alpha.hat for more details and $K_b = b^{-1}K(./b)$ is an ordinary kernel function, e.g. the Epanechnikov kernel. For $p = 1$ and $\hat{\theta} = 1$ the above estimator becomes the HQM hazard rate estimator conditional on one covariate,

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

defined in equation (2) of Bagkavos et al. (2025) doi:10.1093/biomet/asaf008.

The future conditional hazard is defined as

$$h_{x,T}(t) = P\left(T_i \in (t+T, t+T+dt) | \theta_0^T X_i(T) = x, T_i > t+T\right),$$

where T_i is the survival time and X_i the marker of individual i observed in the time frame $[0, T]$.

Value

A single numeric value of $\hat{h}_x(t)$.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

See Also

get.alpha.hat

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
            'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
              'years', n)

x = 2
t = 2
h.hat = h.hat.x.t(X.grid, s.grid, X.pos, n.s.grid, alpha.hat, x, t, b, Z.i.ds, n)
```

---

h.hat.x.t.vec *Hqm estimator on the marker grid*

Description

Computes the (indexed) hqm estimator on the marker grid.

Usage

```
h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, t, b, Z.i.ds, X.pos, n)
```

Arguments

X.grid	Marker value grid points that will be used in the evaluatiuon.
s.grid	Time value grid points that will be used in the evaluatiuon.
n.s.grid	Size of the time grid.
alpha.hat	Marker-hazard obtained from get.alpha.hat.
t	Numeric value of the time the function should be evaluated.
b	Bandwidth.
Z.i.ds	A matrix made by make.Z.i.ds indicating the exposure.
X.pos	Position of the linear interpolated marker values on the marker grid.
n	Number of individuals.

Details

The function implements the future conditional hazard estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(\hat{\theta}^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s},$$

for every x on the marker grid, where, for a positive integer p, $\hat{\theta}^T = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ is the vector of the estimated indexing parameters, $X_i = (X_{1,i}, \ldots, X_{i,p})$ is a vector of markers for indexing, Z_i is the exposure and $\alpha(z)$ is the marker-only hazard, see get.alpha.hat for more details and $K_b = b^{-1}K(./b)$ is an ordinary kernel function, e.g. the Epanechnikov kernel. For $p = 1$ and $\hat{\theta} = 1$ the above estimator becomes the HQM hazard rate estimator conditional on one covariate,

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

defined in equation (2) of Bagkavos et al. (2025), doi:10.1093/biomet/asaf008.

Value

A vector of $\hat{h}_x(t)$ for all values x on the marker grid.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

Examples

```
# Longitudinal data example

data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
```

```
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid, s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
              n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)

t = 2

h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, t, b, Z.i.ds, X.pos, n)

# Time-invariant data example:
# pbc2 dataset, single event per individual version:
# 312 observations, most recent event per individual.
# Use landmarking to produce comparable curve with KM.
library(survival)
Landmark <- 3 #set the landmark to 3 years
pbc2.use<- to.id(pbc2) # keep only the most recent row per patient
pbcT1 <- pbc2.use[which(pbc2.use$year< Landmark  & pbc2.use$years> Landmark),]

timesS2 <- seq(Landmark,14,by=0.5)
b=0.9
h.hat.x<- get.h.hat.x(pbcT1, 'albumin', s.grid=s.grid,  T.name = 'years',  s.name = 'year',
                                                    delta.name = 'status2', 2, b)
s.grid.2  = seq(Landmark, 14,  length=99)
S.hat.alb<- make.S.hat(    (s.grid.2[2]-s.grid.2[1])/1.35 , h.hat.x)
kma2<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

#Plot the survival functions:
plot(s.grid.2, S.hat.alb, type="l", ylim=c(0,1), xlim=c(Landmark,14), ylab="Survival probability",
        xlab="years",lwd=2, main="HQM and KM survival functions, conditional on albumin=2,
            for the time-invariant pbc dataset")
lines(kma2$time, kma2$surv, type="s",lty=2, lwd=2, col=2)
legend("bottomleft", c("HQM est.", "Kaplan-Meier"), lty=c(1,2), col=1:2, lwd=2, cex=1.7)


############## Indexed HR estimator example 1: ###############
### The example is based on indexing two covariates: albumin and bilirubin
```

```
index.use<-84
t.grid<-s.grid[1:index.use] #non need to plot the last year

b.alb = 1.5 # results from CV rule of Bagkavos et al. (2025), i.e.:
 # b.candidates = seq(1.5, 3, 0.05)
 # CV.b.alb = CV.b(pbc2, 'albumin', 'years', 'year', 'status2', size.I.j, b.candidates)
 # b.alb =  CV.b.alb[[2]][which.min(CV.b.alb[[1]])]

b.bil = 4 # results from CV rule of Bagkavos et al. (2025) :
  # b.candidates.bil = seq(3, 4, 0.05)
 # CV.b.bil = CV.b(pbc2, 'serBilir', 'years', 'year', 'status2', size.I.j, b.candidates.bil)
  # b.bil =  CV.b.alb[[2]][which.min(CV.b.alb[[1]])] - result = 4

x.alb = 1 # refers to zero mean variables
          #corresponds to approximately normal albumin level
x.bil = 1.9 # refers to zero mean variable - corresponds to
          #elevated bilirubin levelm approximately 7mg/dl

theta.alb  <- 0.0702 # results from the indexing process
theta.bil <- 0.0856 # results from the indexing process

# First, mean adjust the albumin covariate
X.alb.obs = pbc2$albumin - mean(pbc2$albumin)
X.alb.i.T.i = data.id$albumin - mean(data.id$albumin)

X.alb.grid = seq(min(X.alb.obs), max(X.alb.obs), (max(X.alb.obs)-min(X.alb.obs))/( n.X.grid-1))
X.alb.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.alb.obs, s.obs)
X.alb.pos <- findInterval(X.alb.i.s, X.alb.grid)
dN.alb <- make.dN(pbc2, data.id, X.alb.grid, s.grid, T.i, X.alb.i.T.i, delta.i)
Z.ds.alb <- make.Z.ds(pbc2, data.id, X.alb.i.s, X.alb.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.alb.pos, T.name = 'years', n)

alpha.hat.alb<-get.alpha.hat(dN.alb, Z.ds.alb, b.alb, X.alb.grid, K=K.Epanechnikov )
Z.i.ds.alb <- make.Z.i.ds(pbc2, data.id, X.alb.i.s, X.alb.grid, s.grid, n.s.grid, n.X.grid,
                                        s.pos, X.alb.pos, T.name = 'years', n)

#calculate HQM estimator for original albumin covariate:
h.hat.alb<- h.hat.x.t.vec(X.alb.grid, s.grid, n.s.grid, alpha.hat.alb, x.alb, b.alb, Z.i.ds.alb,
                                                                    X.alb.pos, n)

y.grid.alb<-h.hat.alb[1:index.use]

# Now, mean adjust the bilirubon covariate:
X.bil.obs = pbc2$serBilir - mean(pbc2$serBilir)
X.bil.i.T.i = data.id$serBilir - mean(data.id$serBilir)

X.bil.grid = seq(min(X.bil.obs), max(X.bil.obs), (max(X.bil.obs)-min(X.bil.obs))/( n.X.grid-1))
X.bil.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.bil.obs, s.obs)
X.bil.pos <- findInterval(X.bil.i.s, X.bil.grid)
dN.bil <- make.dN(pbc2, data.id, X.bil.grid, s.grid, T.i, X.bil.i.T.i, delta.i)
Z.ds.bil <- make.Z.ds(pbc2, data.id, X.bil.i.s, X.bil.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.bil.pos, T.name = 'years', n)
```

```
alpha.hat.bil<-get.alpha.hat(dN.bil, Z.ds.bil, b.bil, X.bil.grid, K=K.Epanechnikov )
Z.i.ds.bil <- make.Z.i.ds(pbc2, data.id, X.bil.i.s, X.bil.grid, s.grid, n.s.grid, n.X.grid,
                                        s.pos, X.bil.pos, T.name = 'years', n)

#calculate HQM estimator for original bilirubin covariate:
h.hat.bil<- h.hat.x.t.vec(X.bil.grid, s.grid, n.s.grid, alpha.hat.bil, x.bil, b.bil, Z.i.ds.bil,
                                                                          X.bil.pos, n)
y.grid.bil<-h.hat.bil[1:index.use]

# calculate the indexed variables:
theta.T.X.obs = theta.alb * X.alb.obs +  theta.bil *X.bil.obs
theta.T.X.i.T.i = theta.alb * X.alb.i.T.i + theta.bil *X.bil.i.T.i
b = 0.4 # results from CV rule in Bagkavos et al (2025)
x = theta.alb * x.alb + theta.bil *x.bil

X.grid = seq(min(theta.T.X.obs), max(theta.T.X.obs),
         (max(theta.T.X.obs)-min(theta.T.X.obs))/( n.X.grid-1))

theta.T.X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, theta.T.X.obs, s.obs)
X.pos <- findInterval(theta.T.X.i.s, X.grid)
dN <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, theta.T.X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, theta.T.X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos,
                                                              X.pos, 'years', n)
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )
Z.i.ds <- make.Z.i.ds(pbc2, data.id, theta.T.X.i.s, X.grid, s.grid, n.s.grid, n.X.grid,
                                                     s.pos, X.pos,'years', n)

# Now calculate indexed HR estimator:
h.hat.x.theta<- h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, x, b, Z.i.ds, X.pos, n)
y.grid2<-h.hat.x.theta[1:index.use]


############### Plot the results:
plot(t.grid, y.grid2, type="l", ylim=c(0,2), xlab="Years", ylab="Hazard rate", lwd=2 )
lines(t.grid, y.grid.bil , lty=2, col=4, lwd=2)
lines(t.grid, y.grid.alb, lty=2, col=2, lwd=2)

legend("topleft", lty=c(1, 2, 2), lwd=c(2,2,2),
       c("Indexed (Bilirubin and Albumin) HQM hazard rate est.",
         "HQM hazard rate est. conditional on Albumin=2mg/dl",
         "HQM hazard rate est. conditional on Bilirubin=7mg/dl" ),
       col=c(1,2,4), cex=2, bty="n")

############## Indexed HR estimator example 2: ##############
### The example is based on indexing two covariates:  bilirubin and prothrombin time

index.use<-84
t.grid<-s.grid[1:index.use]

b.pro = 3
b.bil = 4

x.pro = 1.5 # refers to zero mean variables - slightly high
```

```
x.bil = 1.9 # refers to zero mean variable - high

theta.pro  <- 0.349 #0.5 #0.149
theta.bil <- 0.0776 #0.10

############ prothrombin time #############
ind1 <- which( is.na( pbc2$prothrombin ) )
X.pro.obs =   pbc2$prothrombin
X.pro.obs[ind1]<-mean(pbc2$prothrombin, na.rm=TRUE)
X.pro.obs <- X.pro.obs - mean(X.pro.obs)

ind2 <- which( is.na( data.id$prothrombin ) )
X.pro.i.T.i =   data.id$prothrombin
X.pro.i.T.i[ind2]<-mean(data.id$prothrombin, na.rm=TRUE)
X.pro.i.T.i <- X.pro.i.T.i - mean(X.pro.i.T.i)

X.pro.grid = seq(min(X.pro.obs), max(X.pro.obs), (max(X.pro.obs)-min(X.pro.obs))/( n.X.grid-1))
X.pro.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.pro.obs, s.obs)
X.pro.pos <- findInterval(X.pro.i.s, X.pro.grid)
dN.pro <- make.dN(pbc2, data.id, X.pro.grid, s.grid, T.i, X.pro.i.T.i, delta.i)
Z.ds.pro <- make.Z.ds(pbc2, data.id, X.pro.i.s, X.pro.grid, s.grid,
               n.s.grid, n.X.grid, s.pos, X.pro.pos, T.name = 'years', n)

alpha.hat.pro<-get.alpha.hat(dN.pro, Z.ds.pro, b.pro, X.pro.grid, K=K.Epanechnikov )
Z.i.ds.pro <- make.Z.i.ds(pbc2, data.id, X.pro.i.s, X.pro.grid, s.grid, n.s.grid, n.X.grid, s.pos,
                                                    X.pro.pos, T.name = 'years', n)
h.hat.pro<- h.hat.x.t.vec(X.pro.grid, s.grid, n.s.grid, alpha.hat.pro, x.pro, b.pro, Z.i.ds.pro,
                          X.pro.pos, n)
y.grid.pro<-h.hat.pro[1:index.use]


############ Bilirubin #############
X.bil.obs = pbc2$serBilir - mean(pbc2$serBilir)
X.bil.i.T.i = data.id$serBilir - mean(data.id$serBilir)

X.bil.grid = seq(min(X.bil.obs), max(X.bil.obs), (max(X.bil.obs)-min(X.bil.obs))/( n.X.grid-1))
X.bil.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.bil.obs, s.obs)
X.bil.pos <- findInterval(X.bil.i.s, X.bil.grid)
dN.bil <- make.dN(pbc2, data.id, X.bil.grid, s.grid, T.i, X.bil.i.T.i, delta.i)
Z.ds.bil <- make.Z.ds(pbc2, data.id, X.bil.i.s, X.bil.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.bil.pos, T.name = 'years', n)

alpha.hat.bil<-get.alpha.hat(dN.bil, Z.ds.bil, b.bil, X.bil.grid, K=K.Epanechnikov )
Z.i.ds.bil <- make.Z.i.ds(pbc2, data.id, X.bil.i.s, X.bil.grid, s.grid, n.s.grid, n.X.grid, s.pos,
                                                    X.bil.pos, T.name = 'years', n)
h.hat.bil<- h.hat.x.t.vec(X.bil.grid, s.grid, n.s.grid, alpha.hat.bil, x.bil, b.bil, Z.i.ds.bil,
                          X.bil.pos, n)
y.grid.bil<-h.hat.bil[1:index.use]

######### Index prothrombin time and bilirubin
theta.T.X.obs = theta.pro * X.pro.obs +  theta.bil *X.bil.obs
theta.T.X.i.T.i = theta.pro * X.pro.i.T.i + theta.bil *X.bil.i.T.i
```

```
b = 1.5
x = theta.pro * x.pro + theta.bil *x.bil

X.grid = seq(min(theta.T.X.obs), max(theta.T.X.obs),
          (max(theta.T.X.obs)-min(theta.T.X.obs))/( n.X.grid-1))

theta.T.X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, theta.T.X.obs, s.obs)
X.pos <- findInterval(theta.T.X.i.s, X.grid)
dN <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, theta.T.X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, theta.T.X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
                                                                          'years', n)
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )
Z.i.ds <- make.Z.i.ds(pbc2, data.id, theta.T.X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos,
                        X.pos,'years', n)
h.hat.x.theta<- h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, x, b, Z.i.ds, X.pos, n)
y.grid2<-h.hat.x.theta[1:index.use]



##############Plot the results
plot(t.grid, y.grid2, type="l", ylim=c(0,2), xlab="Years", ylab="Hazard rate", lwd=2 )
lines(t.grid, y.grid.bil , lty=2, col=4, lwd=2)

lines(t.grid, y.grid.pro, lty=2, col=2, lwd=2)
legend("topleft", lty=c(1, 2, 2), lwd=c(2,2,2),
       c("Indexed (Bilirubin and PT) HQM hazard rate est.",
        "HQM hazard rate est. conditional on PT=15.5s",
        "HQM hazard rate est. conditional on Bilirubin=7mg/dl" ),
        col=c(1,2,4), cex=2, bty="n")
```

`h.tilde.x.t` *Local linear future conditional hazard rate estimation at a single time point*

Description

Calculates the (indexed) local linear future conditional hazard rate for a marker value `x` and a time value `t`.

Usage

```
h.tilde.x.t(X.grid, s.grid, X.pos, n.s.grid, alpha.hat, x,t, b, Z.i.ds,n, Z.ds)
```

Arguments

`X.grid`	Vector of grid points for the marker values X.
`s.grid`	Vector of grid points for the time values s.
`X.pos`	Position of the linear interpolated marker values on the marker grid.

n.s.grid	Size of the time grid.
alpha.hat	Marker-hazard obtained from get.alpha.hat.
x	Numeric value of the last observed marker value.
t	Numeric time value.
b	Bandwidth.
Z.i.ds	A matrix made by make.Z.i.ds indicating the exposure.
n	Number of individuals.
Z.ds	A matrix made by make.Z.ds indicating the exposure.

Details

Function h.tilde.x.t implements the future local linear conditional hazard estimator

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(\hat{\theta}^T X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - \hat{\theta}^T X_i(s)) \mathrm{d}s},$$

for every x on the marker grid, where, for a positive integer p, $\hat{\theta}^T = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ is the vector of the estimated indexing parameters, $X_i = (X_{1,i}, \ldots, X_{i,p})$ is a vector of markers for indexing, Z_i is the exposure and $\alpha(z)$ is the marker-only hazard, see get.alpha.hat for more details. For $p = 1$ and $\hat{\theta} = 1$ the above estimator becomes the HQM hazard rate estimator conditional on a single covariate,

$$\hat{h}_x(t) = \frac{\sum_{i=1}^n \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s}{\sum_{i=1}^n \int_0^T Z_i(t+s) Z_i(s) K_b(x - X_i(s)) \mathrm{d}s},$$

defined in equation (2) of Bagkavos et al. (2025) doi:10.1093/biomet/asaf008. In the place of $K_b()$, h.tilde.x.t uses the kernel

$$K_{x,b}(u) = \frac{K_b(u) - K_b(u) u^T D^{-1} c_1}{c_0 - c_1^T D^{-1} c_1},$$

where $K_b() = b^{-1} K(./b)$ with K being an ordinary kernel, e.g. the Epanechnikov kernel, $c_1 = (c_{11}, \ldots, c_{1d})^T$, $D = (d_{ij})_{(d+1)\times(d+1)}$ with

$$c_0 = \sum_{i=1}^n \int_0^T K_b(x - \hat{\theta}^T X_i(s)) Z_i(s) ds,$$

$$c_{ij} = \sum_{i=1}^n \int_0^T K_b(x - \hat{\theta}^T X_i(s)) \{x - \hat{\theta}^T X_{ij}(s)\} Z_i(s) ds,$$

$$d_{jk} = \sum_{i=1}^n \int_0^T K_b(x - \hat{\theta}^T X_i(s)) \{x - \hat{\theta}^T X_{ij}(s)\} \{x - \hat{\theta}^T X_{ik}(s)\} Z_i(s) ds,$$

see also Nielsen (1998) doi:10.1080/03461238.1998.10413997.

The future conditional hazard is defined as

$$h_{x,T}(t) = P\left(T_i \in (t+T, t+T+dt) | \theta_0^T X_i(T) = x, T_i > t+T\right),$$

where T_i is the survival time and X_i the marker of individual i observed in the time frame $[0, T]$.

Value

A single numeric value of $\hat{h}_x(t)$.

References

Bagkavos, I., Isakson, R., Mammen, E., Nielsen, J., and Proust–Lima, C. (2025). Biometrika, 112(2), asaf008. doi:10.1093/biomet/asaf008

Nielsen (1998), Marker dependent kernel hazard estimation from local linear estimation, Scandinavian Actuarial Journal, pp. 113-124. doi:10.1080/03461238.1998.10413997

See Also

get.alpha.hat, c2.u.integrand

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid=X.grid, s.grid=s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
            'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid, n.s.grid, n.X.grid, s.pos, X.pos,
             'years', n)

x = 2
t = 2
h.tilde = h.tilde.x.t(X.grid, s.grid, X.pos, n.s.grid, alpha.hat, x, t, b, Z.i.ds, n, Z.ds)
```

`h.true.simulated` *Simulated true hazard (bootstrap average)*

Description

Compute the Monte Carlo / bootstrap averaged alpha (true hazard) across a list of datasets.

Usage

```
h.true.simulated(bootstrap.samples, id.name, n.s.grid, X.1.name = X.1.name,
       X.2.name = X.2.name, T.name = T.name,
       s.name = s.name, delta.name = delta.name, theta, b)
```

Arguments

`bootstrap.samples`	List of data.frames (bootstrap samples).
`id.name`	Id column name.
`n.s.grid`	user supplied grid length on the s.name argument
`X.1.name`	First marker name.
`X.2.name`	Second marker name.
`T.name`	Name of event time column.
`s.name`	Name of observation time column.
`delta.name`	Name of event indicator column.
`theta`	Numeric indexing parameters.
`b`	Bandwidth.

Value

Numeric vector (row means of alpha estimates).

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'

n.s.grid<-50 #user supplied grid length on the s.name argument

theta.1  <- 0.0702  #indexing parameter for marker 1
theta.2 <- 0.0856  #indexing parameter for marker 2
```

```
x.1 = 0 # conditioning level for marker 1, assumes zero mean variable
x.2 = 1.9 # conditioning level for marker 2, assumes zero mean variable
b = 0.42 # The result, on the indexed marker 'indmar' of
      #\code{CV.b(pbc2,'indmar','years','year','status2',size.I.j=26,seq(0.2,0.4,by=0.01))}
x = theta.1 * x.1 + theta.2 *x.2

data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )

#  Create bootstrap samples by group
set.seed(1)
B<-100 #set bootstrap iterations here
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
  bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),]
}

#compute the bootstrap simulated true HR function:
h.true<- h.true.simulated(bootstrap.samples, id.name='id', n.s.grid=n.s.grid,
              X.1.name=X.1.name, X.2.name= X.2.name,
              T.name = T.name, s.name = s.name,
              delta.name = delta.name, theta = c(theta.1,  theta.2), b)
```

`iid.AUC.hat` *IID decomposition for time-dependent AUC with IPCW*

Description

Internal function implementing the influence function (iid) decomposition of the time-dependent Area Under the Curve (AUC) in the presence of right-censoring, using inverse probability of censoring weighting (IPCW).

This function reproduces the internal behavior of the corresponding routines from the `timeROC` package. It supports both standard survival settings and competing risks settings through automatic dispatch.

Usage

```
iid.AUC.hat(t, n, cause, F.1.t, S.t,
                          G.hat, T.i, delta.i, pi.i, int.dM.C.over.S)
```

Arguments

t	Numeric scalar. Time point at which the AUC is evaluated.
n	Integer. Sample size.
cause	Integer or numeric. Event type of interest.
F.1.t	Numeric. Estimated cumulative incidence function at time t for the event of interest.
S.t	Numeric. Estimated survival probability at time t.
G.hat	List. IPCW weights object returned by pec::ipcw. Must contain components IPCW.subjectTimes, IPCW.times, and times.
T.i	Numeric vector. Observed event or censoring times.
delta.i	Integer or numeric vector. Event indicator: 0 for censoring, cause for the event of interest, and other values for competing events.
pi.i	Numeric vector. Prognostic marker or risk score.
int.dM.C.over.S	Numeric matrix. Influence function representation of the censoring distribution, typically obtained from iid.G.hat().

Details

The function computes the iid (influence function) representation of the time-dependent AUC estimator based on IPCW methodology.

Two cases are handled internally:

- **Survival setting:** only one event type is present (no competing risks).
- **Competing risks setting:** multiple event types are present, and the AUC is defined using different control definitions.

The implementation relies on pairwise comparisons between cases and controls, weighted by inverse probability of censoring weights. The iid representation is derived from a U-statistic formulation and incorporates the influence of the censoring distribution via martingale-based corrections.

The computational complexity is quadratic in the sample size.

Value

A list with the following components:

iid.AUC	Numeric vector of length n. Influence function values for the AUC estimator.
iid.AUC.star	Numeric vector of length n. Alternative influence function corresponding to an alternative control definition (mainly relevant in competing risks).
se.AUC	Numeric scalar. Estimated standard error of the AUC.

`se.AUC.star` Numeric scalar. Estimated standard error corresponding to `AUC.star`.

`computation.times`
Placeholder (numeric or `NA`). Included for compatibility with `timeROC`.

Note

This is an internal utility function and is not intended for direct use by end users. It is provided for reproducibility and to avoid dependency on archived packages.

Author(s)

Adapted from the `timeROC` package.

References

Blanche, P., Dartigues, J. F., and Jacqmin-Gadda, H. (2013). Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks. *Statistics in Medicine*, 32(30), 5381–5397.

Uno, H., Cai, T., Tian, L., and Wei, L. J. (2007). Evaluating prediction rules for t-year survivors with censored regression models. *Journal of the American Statistical Association*, 102(478), 527–537.

See Also

`timeROC` (in the original `timeROC` package), `pec::ipcw`, `iid.G.hat`

`iid.G.hat` *iid.G.hat function implementation*

Description

Calculates the AUC for the HQM estimator.

Usage

```
iid.G.hat(T.i,delta.i)
```

Arguments

`T.i`	The vector of (censored) event-times.
`delta.i`	The vector of event indicators at the corresponding value of the vector T. Censored observations must be denoted by the value 0.

Details

test

Value

A vector of two values: the landmark time of the calculation and the AUC value.

See Also

BS.hat.HQM

K.b *Classical (unmodified) kernel and related functionals*

Description

Implements the classical kernel function and related functionals

Usage

```
K.b(b,u,v, K)
c1.u.integrand(b,u,v, K)
K.b.matrix(b,u,v, K)
```

Arguments

u	A vector of design points where the kernel will be evaluated.
v	A vector of sample data points.
b	The bandwidth to use (a scalar).
K	The kernel function to use.

Details

The function K.b implements the classical kernel function calculation

$$h^{-1}K\left(\frac{x-y}{h}\right)$$

for scalars x and y while c1.u.integrand implements the functional

$$h^{-1}K\left(\frac{x-y}{h}\right)(x-y)$$

again for for scalars x and y. The function K.b.matrix is the vectorized version of K.b. It uses as inputs the vectors $(X_1,\ldots,X_n)$ and $(Y_1,\ldots,Y_n)$ and returns a $n \times n$ matrix with entries

$$h^{-1}K\left(\frac{X_i-Y_j}{h}\right)$$

Value

Scalar values for K.b and c1.u.integrand and matrix outputs for K.b.matrix.

`K.b.u.v` *Local linear kernel*

Description

Implements the local linear kernel function.

Usage

```
K.b.u.v(b,u,v, K)
```

Arguments

`u`	A vector of design points where the kernel will be evaluated.
`v`	A vector of sample data points.
`b`	The bandwidth to use (a scalar).
`K`	The kernel function to use.

Details

Implements the local linear kernel

$$K_{x,b}(u) = \frac{K_b(u) - K_b(u)u^T D^{-1} c_1}{c_0 - c_1^T D^{-1} c_1},$$

where $c_1 = (c_{11}, \ldots, c_{1d})^T$, $D = (d_{ij})_{(d+1)\times(d+1)}$ with

$$c_0 = \sum_{i=1}^{n} \int_0^T K_b(x - X_i(s)) Z_i(s) ds,$$

$$c_{ij} = \sum_{i=1}^{n} \int_0^T K_b(x - X_i(s)) \{x - X_{ij}(s)\} Z_i(s) ds,$$

$$d_{jk} = \sum_{i=1}^{n} \int_0^T K_b(x - X_i(s)) \{x - X_{ij}(s)\} \{x - X_{ik}(s)\} Z_i(s) ds,$$

see also Nielsen (1998), doi:10.1080/03461238.1998.10413997.

Value

Matrix output with entries the values of the kernel function at each point.

References

Nielsen (1998), Marker dependent kernel hazard estimation from local linear estimation, Scandinavian Actuarial Journal, pp. 113-124. doi:10.1080/03461238.1998.10413997

K.Epanechnikov *Epanechnikov kernel*

Description

Implements the Epanechnikov kernel function

Usage

```
K.Epanechnikov(u)
```

Arguments

u	A vector of design points where the kernel will be evaluated.

Details

Implements the Epanechnikov kernel function

$$K(x) = \frac{3}{4}(1 - x^2) * (|x| < 1)),$$

Value

Scalar, the value of the Epanechnikov kernel at u.

make.dN.Z.ds *Occurance and Exposure on grids*

Description

Auxiliary functions that help automate the process of calculating integrals with occurances or exposure processes.

Usage

```
make.dN(data, data.id, X.grid, s.grid, T.i, X.i.T.i, delta.i)
make.dN.i(s.grid, n.s.grid, T.i, delta.i, n)
make.Z.ds(data, data.id, X.i.s, X.grid, s.grid,
        n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)
make.Z.i.ds(data, data.id, X.i.s, X.grid, s.grid,
        n.s.grid, n.X.grid, s.pos,X.pos, T.name = 'years', n)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
data.id	An id data frame obtained from to.id.
X.grid	Marker value grid points where the function will be evaluated.
s.grid	Time value grid points where the function will be evaluated.
T.i	Vector with event times.
X.i.T.i	Vector of last observed marker values.
delta.i	0-1 vector of whether events happened.
n.s.grid	Size of the time grid.
n.X.grid	Size of the marker grid.
n	Number of individuals.
X.i.s	Linear interpolation of observed marker values evaluated on the marker grid.
s.pos	Position of the observed time values on the time grid.
X.pos	Position of the linear interpolated marker values on the marker grid.
T.name	String of the column name with the event times.

Details

Implements matrices for the computation of integrals with occurences and exposures of the form

$$\int f(s)Z(s)Z(s+t)ds, \int f(s)Z(s)ds, \int f(s)dN(s).$$

where N is a 0-1 counting process, Z the exposure and f an arbitrary function.

Value

The functions make.dN and make.Z.ds return a matrix on the time grid and marker grid for occurence and exposure, respectively, while make.dN.i and make.Z.i.ds return a matrix on the time grid for evey individual again for occurence and exposure, respectively.

See Also

h.hat.x.t, g.hat.t.x, get.alpha.hat

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
```

```
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid, s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)
Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
              n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)
dN.i  <- make.dN.i(s.grid, n.s.grid, T.i, delta.i, n)
```

`make.S.hat` *Survival function from a hazard*

Description

Creates a survival function from a hazard rate which was calculated on a grid.

Usage

```
make.S.hat(du, h.hat.x)
```

Arguments

`du`	Numeric value indicating the distance between two grid continuous grid points.
`h.hat.x`	Vector of hazard values. Hazard rate must have been calculated on a time grid.

Details

The function `make.S.hat` calculates the survival function

$$S(t) = \exp(-\int_0^t h(t)dt),$$

where h is the hazard rate. Here, a discritisation via an equidistant grid $\{t_i\}$ on $[0, t]$ is used to calculate the integral and it is assumed that h has been calculated for exactly these time points t_i.

Value

A vector of values $S(t_i)$.

Examples

```
make.S.hat(0.1, rep(0.1,10))
```

pbc2 *Mayo Clinic Primary Biliary Cirrhosis Data*

Description

Followup of 312 randomised patients with primary biliary cirrhosis, a rare autoimmune liver disease, at Mayo Clinic.

Usage

```
pbc2
```

Format

A data frame with 1945 observations on the following 20 variables.

`id` patients identifier; in total there are 312 patients.

`years` number of years between registration and the earlier of death, transplantion, or study analysis time.

`status` a factor with levels `alive`, `transplanted` and `dead`.

`drug` a factor with levels `placebo` and `D-penicil`.

`age` at registration in years.

`sex` a factor with levels `male` and `female`.

`year` number of years between enrollment and this visit date, remaining values on the line of data refer to this visit.

`ascites` a factor with levels `No` and `Yes`.

`hepatomegaly` a factor with levels `No` and `Yes`.

`spiders` a factor with levels `No` and `Yes`.

`edema` a factor with levels `No edema` (i.e., no edema and no diuretic therapy for edema), `edema no diuretics` (i.e., edema present without diuretics, or edema resolved by diuretics), and `edema despite diuretics` (i.e., edema despite diuretic therapy).

`serBilir` serum bilirubin in mg/dl.

`serChol` serum cholesterol in mg/dl.

`albumin` albumin in gm/dl.

`alkaline` alkaline phosphatase in U/liter.

`SGOT` SGOT in U/ml.

`platelets` platelets per cubic ml / 1000.

`prothrombin` prothrombin time in seconds.

`histologic` histologic stage of disease.

`status2` a numeric vector with the value 1 denoting if the patient was dead, and 0 if the patient was alive or transplanted.

References

Fleming, T. and Harrington, D. (1991) *Counting Processes and Survival Analysis*. Wiley, New York.

Therneau, T. and Grambsch, P. (2000) *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York.

Examples

```
summary(pbc2)
```

`Q.theta` *Indexing parameter objective function*

Description

Objective used for optimizing indexing parameters via `optim`.

Usage

```
Q.theta(theta, data, data.id, s.grid, X.obs.list, X.i.T.i.list, T.name = 'years',
            s.name = 'year', delta.name = 'status2', b, x, h.true)
```

Arguments

`theta`	Numeric vector length 2.
`data`	Data frame.
`data.id`	Id-level data frame.
`s.grid`	user-supplied grid points ranging from the minimum to the maximum of the s.name argument
`X.obs.list`	List of vectors for indexing: each vector corresponds to a single biomarker and contains its longitudinal measurements across all individuals/time points.
`X.i.T.i.list`	List of vectors for indexing: each vector corresponds to a biomarker and contains one summary measurement per individual.
`T.name`	Name of event time column.
`s.name`	Name of observation time column.
`delta.name`	Name of event indicator column.
`b`	Bandwidth.
`x`	Evaluation point or grid.
`h.true`	Numeric vector of true hazard values for CV.

Value

Scalar cross-validation score.

Examples

```
X.1.name <- 'albumin'
X.2.name <-  'serBilir'
T.name <- 'years'
s.name <- 'year'
delta.name <- 'status2'
id.name<-'id'


theta.1  <- 0.0702 #0.149
theta.2 <- 0.0856 #0.10
x.1 = 0 # refers to zero mean variables - slightly high
x.2 = 1.9 # refers to zero mean variable - high
b = 0.42
x = theta.1 * x.1 + theta.2 *x.2


# first simulate true HR function:
data <- pbc2[,c(id.name, X.1.name, X.2.name, T.name, s.name, delta.name)]
n.s.grid<-50 # 50 grid points to evaluate the estimates
s.obs<- data[, 'year']
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))

n.obs <- length(data$id)
n<-max(  as.double(data[,'id']) )
####################  Create bootstrap samples by group ####################
set.seed(1)
B<- 10 # 400 #50
bootstrap.samples<-list()
for(j in 1:B)
{
  rows.star<-c()
  ids.star<-c()
  ids.drawn <- sample (n, replace = TRUE)
  for(l in 1:n)
  {
    rows.l<-which(data[,id.name]==ids.drawn[l])
    rows.star<-c(rows.star, rows.l)
    ids.l<-rep(ids.drawn[l], times=length(rows.l))
    ids.star<-c(ids.star, ids.l)
  }
  data.star.unsorted<-data[rows.star,]
  data.star.unsorted<-data[rows.star,]
 bootstrap.samples[[j]]<- data.star.unsorted[order(data.star.unsorted$id),] #data[rows.star,]
}
h.true<- h.true.simulated(bootstrap.samples, id.name='id', n.s.grid=n.s.grid,
              X.1.name=X.1.name, X.2.name= X.2.name,
              T.name = T.name,
              s.name = s.name,  delta.name = delta.name,
              theta = c(theta.1,  theta.2), b)
#############################################################################
```

```
# Then run the optimization for the indexing parameters:

data.use<-bootstrap.samples[[1]]
data.use.id<-to.id(data.use)
data.use.id<-data.use.id[complete.cases(data.use.id), ]
X.1.obs=data.use[,X.1.name] -mean(data.use[, X.1.name])
X.1.i.T.i=data.use.id[,X.1.name] -mean(data.use.id[, X.1.name])
X.2.obs=data.use[,X.2.name]  -mean(data.use[, X.2.name])
X.2.i.T.i=data.use.id[,X.2.name] -mean(data.use.id[, X.2.name])

X.obs.list=list(X.1.obs, X.2.obs)
X.i.T.i.list=list(X.1.i.T.i, X.2.i.T.i)

s.obs<- data.use[, s.name]
s.grid <-  seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
theta.fit<- optim(par=c(theta.1, theta.2), fn=Q.theta,  data=data.use, data.id=data.use.id,
        s.grid=s.grid, X.obs.list=X.obs.list, X.i.T.i.list=X.i.T.i.list,  T.name = T.name,
            s.name = s.name, delta.name = delta.name, b=b, x=x,
            h.true=h.true, method="Nelder-Mead")
```

Q1 *Bandwidth selection score Q1*

Description

Calculates a part for the K-fold cross validation score.

Usage

```
Q1(h.hat.matrix, X.pos, n.s.grid, n, Z.i.ds)
```

Arguments

`h.hat.matrix`	A matrix of the estimator for the future conditional hazard rate for all values `x` and `t`.
`X.pos`	Vector of the position of the observed marker values in the grid for marker values.
`n.s.grid`	Numeric value indicating the number of grid points of the time grid.
`n`	Number of individuals.
`Z.i.ds`	A matrix made by `make.Z.i.ds` indicating the exposure.

Details

The function implements

$$Q_1 = \sum_{i=1}^{N} \int_0^T \int_s^T Z_i(t) Z_i(s) \hat{h}^2_{X_i(s)}(t-s) dt ds,$$

where $\hat{h}$ is the hqm estimator, Z the exposure and X the marker.

Value

A value of the score Q1.

See Also

CV.b

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid, s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
            n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
              n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)
dN.i  <- make.dN.i(s.grid, n.s.grid, T.i, delta.i, n)

t = 2

h.hat.matrix = t(sapply(s.grid[1:99],
 function(t.point){h.hat.x.t.vec(X.grid, s.grid, n.s.grid, alpha.hat, t, b, Z.i.ds, X.pos, n)}))

Q1.b = Q1(h.hat.matrix, X.pos, n.X.grid, n, Z.i.ds)
```

R.hat.K *Bandwidth selection score R*

Description

Calculates a part for the K-fold cross validation score.

Usage

```
R.hat.K(h.hat.minus.I.j.list, X.pos, n.s.grid, Z.i.ds, dN.i, n)
```

Arguments

h.hat.minus.I.j.list	A list of matrices for all cross validation data sets. Each matrix contains the estimator with the future conditional hazard rate for all values x and t and the respected data set.
X.pos	Vector of the position of the observed marker values in the grid for marker values.
n.s.grid	Numeric value indicating the number of grid points of the time grid.
Z.i.ds	A matrix made by make.Z.i.ds indicating the exposure.
dN.i	A matrix made by make.dN.i indicating the occurence.
n	Number of individuals.

Details

The function implements the estimator

$$\hat{R}_K = \sum_{j=1}^{K} \sum_{i \in I_j} \int_0^T g_i^{-I_j}(t) dN_i(t),$$

where $\hat{g}_i^{-I_j}(t) = \int_0^t Z_i(s) \hat{h}_{X_i(s)}^{-I_j}(t-s) ds$, and $\hat{h}^{-I_j}$ is estimated without information from all counting processes i with $i \in I_j$. This function estimates

$$R = \sum_{i=1}^{N} \int_0^T \int_s^T Z_i(t) Z_i(s) \hat{h}_{X_i(s)}(t-s) h_{X_i(s)}(t-s) dt ds.$$

where $\hat{h}$ is the hqm estimator, Z the exposure and X the marker.

Value

A matrix with $\hat{g}_i^{-I_j}(t)$ for all individuals i and time grid points t.

See Also

CV.b

Examples

```
data.id = to.id(pbc2)
n = max(as.numeric(pbc2$id))
b = 1.5
size.I.j = 104
h.hat.minus.I.j.list = h.hat.minus.I.j(pbc2, data.id, 'serBilir', 'years',
                                        'year', 'status2', n, size.I.j, b)


n.s.grid <- n.X.grid <- 100
s.obs = pbc2$year
X.obs = pbc2$serBilir
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

T.i <- data.id$years
delta.i <- data.id$status2

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)
X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
              n.s.grid, n.X.grid, s.pos, X.pos, 'years', n)
dN.i  <- make.dN.i(s.grid, n.s.grid, T.i, delta.i, n)

two.R.hat.K = R.hat.K(h.hat.minus.I.j.list, X.pos, n.X.grid, Z.i.ds, dN.i, n)
two.R.hat.K
```

---

to.id *Event data frame*

Description

Creates a data frame with only one entry per individual from a data frame with time dependent data. The resulting data frame focusses on the event time and the last observed marker value.

Usage

```
to.id(data)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.

Details

The function to.id uses a data frame of time dependent marker data to create a smaller data frame with only one entry per individual, the last observed marker value and the event time. Note that the column indicating the individuals must have the name id. Note also that this data frame is similar to pbc2.id from the JM package with the difference that the last observed marker value instead of the first one is captured.

Value

A data frame with only one entry per individual.

Examples

```
data.id = to.id(pbc2)
```

---

wild.bootstrap.summands

Precomputation for wild bootstrap

Description

Implements key components for the wild bootstrap of the hqm estimator in preparation for obtaining confidence bands.

Usage

```
wild.bootstrap.summands(g.hat, alpha.hat, dN.i, Z.i.ds, n.s.grid, X.grid, s.grid,
                        t, b, X.pos, x, n)
```

Arguments

g.hat	A vector obtained by g.hat.t.x.
alpha.hat	A vector of the marker only hazard on the marker grid obtained by get.alpha.hat.
dN.i	A matrix made by make.dN.i indicating the occurence.
Z.i.ds	A matrix made by make.Z.i.ds indicating the exposure.
n.s.grid	Size of the time grid.
X.grid	Vector of grid points for the marker values.
s.grid	Time value grid points that will be used in the evaluatiuon.
t	Numeric value of the time the function should be evaluated.
b	Bandwidth.
X.pos	Position of the linear interpolated marker values on the marker grid.
x	Numeric value of the last observed marker value.
n	Number of individuals.

Details

The function implements

$$A_B(t) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \int_0^T \hat{g}_{i,t,x_*}(X_i(s)) V_i \{dN_i(s) - \hat{\alpha}_i(X_i(s)) Z_i(s) ds\},$$

and

$$B_B(t) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} V_i \{\hat{\Gamma}(t, x_*)^{-1} W_i(t, x_*) - \hat{h}_{x_*}(t)\},$$

where $V \sim N(0,1)$,

$$W_i(t) = \int_0^T \hat{\alpha}_i(X_i(t+s)) Z_i(t+s) Z_i(s) K_b(x_*, X_i(s)) \mathrm{d}s,$$

and

$$\hat{\Gamma}(t,x) = \frac{1}{n} \sum_{i=1}^{n} \int_0^{T-t} Z_i(t+s) Z_i(s) K_b(x, X_i(s)) ds,$$

with Z being the exposure and X the marker.

Value

A list of 5 items. The first two are vectors for calculating A_B and the third one a vector for B_B. The 4th one is the value of the hqm estimator that can also be obtained by `h.hat.x.t` and the last one is the value of Γ.

See Also

`confidence.bands`

Examples

```
data.id = to.id(pbc2)
n.s.grid <- n.X.grid <- 100
n = max(as.numeric(pbc2$id))
s.obs = pbc2$year
X.obs = pbc2$serBilir
X.i.T.i = data.id$serBilir
T.i <- data.id$years
delta.i <- data.id$status2
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
X.grid = seq(min(X.obs), max(X.obs), (max(X.obs)-min(X.obs))/( n.X.grid-1))

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)

X.pos <- findInterval(X.i.s, X.grid)
s.pos = rep(1:length(s.grid), n)

dN <- make.dN(pbc2, data.id, X.grid, s.grid, T.i, X.i.T.i, delta.i)
Z.ds <- make.Z.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
```

```
            n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)

b = 1.7
alpha.hat<-get.alpha.hat(dN, Z.ds, b, X.grid, K=K.Epanechnikov )

Z.i.ds <- make.Z.i.ds(pbc2, data.id, X.i.s, X.grid, s.grid,
              n.s.grid, n.X.grid, s.pos, X.pos, T.name = 'years', n)
dN.i  <- make.dN.i(s.grid, n.s.grid, T.i, delta.i, n)

t = 2
x = 2

g.hat = g.hat.t.x(X.grid, s.grid, n.s.grid, X.pos, x, t, b, Z.i.ds, Z.ds, n)

summands = wild.bootstrap.summands(g.hat, alpha.hat, dN.i, Z.i.ds, n.s.grid, X.grid,
                  s.grid, t, b, X.pos, x, n)
summands
```

`X.i.s.interpolate` *Linear interpolation*

Description

Implements a linear interpolation between observered marker values.

Usage

```
X.i.s.interpolate(s.grid, ids, id.obs, X.obs, s.obs)
```

Arguments

`s.grid`	A vector of time values where the function should be evaluated.
`ids`	A vector of ids of individuals for whom the marker values should be interpolated.
`id.obs`	The vector of ids from a data frame of time dependent variables.
`X.obs`	The vector of marker values from a data frame of time dependent variables.
`s.obs`	The vector of time values from a data frame of time dependent variables.

Details

Given time points $t_1, ..., t_K$ and marker values $m_1, ..., m_J$ at different time points $t_1^m, ..., t_J^m$, the function calculates a linear interpolation f with $f(t_i^m) = m_i$ at the time points $t_1, ..., t_K$ for all indicated individuals. Returned are then $(f(t_1), ..., f(t_K))$. Note that the first value is always observed at time point 0 and the function f is extrapolated constantly after the last observed marker value.

Value

A matrix with columns $(f(t_1), ..., f(t_K))$ as described above for every individual in the vector `ids`.

Examples

```
n.s.grid <- 100
X.obs = pbc2$serBilir
s.obs = pbc2$year
s.grid = seq(0, max(s.obs), max(s.obs)/( n.s.grid-1))
data.id = to.id(pbc2)

X.i.s = X.i.s.interpolate(s.grid, data.id$id, pbc2$id, X.obs, s.obs)
```

Index

AUC.hat, 2
AUC.hat.HQM, *6*, 6, *23*

bootstrap.CI.pivotal, 7
bootstrap.CI.quantile, 11
bootstrap.CI.studentized, 16
bootstrap.h.hat.star, *9*, *13*, *18*, 20
BS.hat.HQM, *6*, 22, *23*, *64*

c0.u *(*ck.u*)*, 24
c1.u *(*ck.u*)*, 24
c1.u.integrand *(*K.b*)*, 64
c2.u *(*ck.u*)*, 24
c2.u.integrand, 24, *59*
ck.u, 24
confidence.bands, 25, *26*, *77*
CV.b, 27, *28*, *31*, *33*, *46*, *73*, *74*
CV.theta, 28, *29*, *48*

data, *25*, *27*, *29*, *37*, *42*, *46*, *47*
data.minus.I.j, *28–30*, 30, *46*, *47*
double.bootstrap.h.hat.star, 31

g.hat.i, *28*, *29*, 32, *33*
g.hat.t.x, *26*, *27*, 34, *67*, *76*
get.alpha.hat, 35, *36–38*, *42*, *43*, *46*, *47*, *50–52*, *58*, *59*, *67*, *76*
get.h.hat.x, *37*, 37, *43*
get.h.hat.x.theta, 40
get.h.tilde.x, *38*, 41, *42*

h.hat.minus.I.j, *28*, *29*, 46
h.hat.minus.I.j.theta, 47
h.hat.star.x.theta, *9*, *13*, *18*, *21*, *32*, 48
h.hat.x.t, *25*, *36*, *38*, *43*, 49, *50*, *67*, *77*
h.hat.x.t.vec, 51
h.tilde.x.t, 57, *58*
h.true.simulated, 60

iid.AUC.hat, 61
iid.G.hat, *63*, 63

ipcw, *4*

K.b, 64
K.b.u.v, 65
K.Epanechnikov, 66

make.dN, *35*, *67*
make.dN *(*make.dN.Z.ds*)*, 66
make.dN.i, *67*, *74*, *76*
make.dN.Z.ds, 66
make.S.hat, *6*, *23*, *68*, 68
make.Z.ds, *34*, *35*, *58*, *67*
make.Z.ds *(*make.dN.Z.ds*)*, 66
make.Z.i.ds, *32*, *34*, *50*, *52*, *58*, *67*, *72*, *74*, *76*
make.Z.i.ds *(*make.dN.Z.ds*)*, 66

pbc2, 69

Q.theta, *21*, *32*, 70
Q1, *28*, *29*, 72

R.hat.K, *28*, *29*, 74

to.id, *21*, *32*, *46*, *47*, *67*, 75, *76*

wild.bootstrap.summands, *26*, *27*, 76

X.i.s.interpolate, 78