

Package ‘HQM’

January 20, 2025

Type Package

Title Superefficient Estimation of Future Conditional Hazards Based on Marker Information

Version 0.1.4

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

Description Provides a nonparametric smoothed kernel density estimator for the future conditional hazard when time-dependent covariates are present. It also provides pointwise and uniform confidence bands and a bandwidth selection.

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports stats, utils, survival, pec, timeROC, 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 2024-12-12 18:50:02 UTC

Contents

auc.hqm	2
bs.hqm	3
b_selection	4
b_selection_prep_g	5
Conf_bands	7
dataset_split	8

dij	9
Epan	10
get_alpha	11
get_h_x	12
get_h_xll	14
g_xt	16
h_xt	17
h_xtll	19
h_xt_vec	21
Kernels	23
lin_interpolate	24
llK_b	25
llweights	26
make_N, make_Ni, make_Y, make_Yi	26
make_sf	28
pb2	29
prep_boot	30
prep_cv	32
Q1	33
R_K	35
to_id	36

Index 38

auc.hqm	<i>AUC for the High Quality Marker estimator</i>
---------	--

Description

Calculates the AUC for the HQM estimator.

Usage

```
auc.hqm(xin, est, landm, th, event_time_name, status_name)
```

Arguments

xin	A data frame containing event times and the patient status.
est	The HQM estimator values, typically the output of get_h_x .
landm	Landmark time.
th	Time horizon.
event_time_name	The column name of the event times in the xin data frame.
status_name	The column name of the status variable in the xin frame.

Details

The function [auc.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.hqm](#)

Examples

```
library(timeROC)
library(survival)
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year< Landmark & pbc2$years> Landmark),]
timesS2 <- seq(Landmark,14,by=0.5)
b=0.9
arg1<- get_h_x(pbcT1, 'albumin', event_time_name = 'years',
               time_name = 'year', event_name = 'status2', 2, 0.9)
br_s2 = seq(Landmark, 14, length=99)
sfalb2<- make_sf( (br_s2[2]-br_s2[1])/4 , arg1)
tHor <- 1.5
auc.hq.use<-auc.hqm(pbcT1, sfalb2, Landmark,tHor,
                   event_time_name = 'years', status_name = 'status2')
auc.hq.use
```

 bs.hqm

Brier score for the High Quality Marker estimator

Description

Calculates the Brier score for the HQM estimator.

Usage

```
bs.hqm(xin, est, landm, th, event_time_name, status_name)
```

Arguments

xin	A data frame containing event times and the patient status.
est	The HQM estimator values, typically the output of get_h_x .
landm	Landmark time.
th	Time horizon.
event_time_name	The column name of the event times in the data frame xin.
status_name	The column name of the status variable in the data frame xin.

Details

The function [bs.hqm](#) implements the Brier score calculation for the HQM estimator estimator.

Value

Scalar: the Brier score of the HQM estimator.

See Also

[auc.hqm](#)

Examples

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

pbcT1 <- pbc2[which(pbc2$year< Landmark & pbc2$years> Landmark),]
timesS2 <- seq(Landmark,14,by=0.5)

b=0.9
arg1<- get_h_x(pbcT1, 'albumin', event_time_name = 'years',
              time_name = 'year', event_name = 'status2', 2, 0.9)
br_s2 = seq(Landmark, 14, length=99)
sfalb2<- make_sf( (br_s2[2]-br_s2[1])/4 , arg1)

tHor <- 1.5
bs.use<-bs.hqm(pbcT1, sfalb2, Landmark,tHor,
              event_time_name = 'years', status_name = 'status2')
bs.use
```

b_selection

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

```
b_selection(data, marker_name, event_time_name = 'years',
           time_name = 'year', event_name = 'status2', I, b_list)
```

Arguments

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

marker_name The column name of the marker values in the data frame [data](#).

event_time_name The column name of the event times in the data frame [data](#).

time_name	The column name of the times the marker values were observed in the data frame <code>data</code> .
event_name	The column name of the events in the data frame <code>data</code> .
I	Number of observations leave out for a K cross validation.
b_list	Vector of bandwidths that need to be tested.

Details

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

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

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.

See Also

[b_selection_prep_g](#), [Q1](#), [R_K](#), [prep_cv](#), [dataset_split](#)

Examples

```
I = 26
b_list = seq(0.9, 1.3, 0.1)

b_scores_alb = b_selection(pbc2, 'albumin', 'years', 'year', 'status2', I, b_list)
b_scores_alb[[2]][which.min(b_scores_alb[[1]])]
```

`b_selection_prep_g` *Preparations for bandwidth selection*

Description

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

Usage

```
b_selection_prep_g(h_mat, int_X, size_X_grid, n, Yi)
```

Arguments

h_mat	A matrix of the estimator for the future conditional hazard rate for all values x and t .
int_X	Vector of the position of the observed marker values in the grid for marker values.
size_X_grid	Numeric value indicating the number of grid points for marker values.
n	Number of individuals.
Yi	A matrix made by <code>make_Yi</code> indicating the exposure.

Details

The function `b_selection_prep_g` 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

[b_selection](#)

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/(size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/(size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, breaks_X=br_X, breaks_s=br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

b = 1.7
alpha<-get_alpha(N, Y, b, br_X, K=Epan )

```

```

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)
Ni <- make_Ni(breaks_s=br_s, size_s_grid, ss, delta, n)

t = 2

h_xt_mat = t(sapply(br_s[1:99], function(si){
  h_xt_vec(br_X, br_s, size_s_grid, alpha, t, b, Yi, int_X, n)}))

b_selection_prep_g(h_xt_mat, int_X, size_X_grid, n, Yi)

```

Conf_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

```

Conf_bands(data, marker_name, event_time_name = 'years',
           time_name = 'year', event_name = 'status2', x, b)

```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
marker_name	The column name of the marker values in the data frame <code>data</code> .
event_time_name	The column name of the event times in the data frame <code>data</code> .
time_name	The column name of the times the marker values were observed in the data frame <code>data</code> .
event_name	The column name of the events in the data frame <code>data</code> .
x	Numeric value of the last observed marker value.
b	Bandwidth.

Details

The function `Conf_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 $h_{x_*,B}^{*(1)}(t), \dots, h_{x_*,B}^{*(N)}(t)$ for $N = 1000$ and order it $h_{x_*,B}^{*[1]}(t) \leq \dots \leq h_{x_*,B}^{*[N]}(t)$. Then

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

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

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

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

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 .

See Also

[g_xt](#), [prep_boot](#)

Examples

```
b = 10
x = 3
size_s_grid <- 100
s = pbc2$year
br_s = seq(0, max(s), max(s)/(size_s_grid-1))

c_bands = Conf_bands(pbc2, 'serBilir', event_time_name = 'years',
                    time_name = 'year', event_name = 'status2', x, b)

J = 60
plot(br_s[1:J], c_bands$h_hat[1:J], type = "l", ylim = c(0,1), ylab = 'Hazard', xlab = 'Years')

lines(br_s[1:J], c_bands$I_p_up[1:J], col = "red")
lines(br_s[1:J], c_bands$I_p_do[1:J], col = "red")
lines(br_s[1:J], c_bands$I_nu[1:J], col = "blue")
lines(br_s[1:J], c_bands$I_nd[1:J], col = "blue")
```

dataset_split

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

```
dataset_split(I, data)
```

Arguments

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

`I` 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 `dataset_split` 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 $\cup_{j=1}^K I_j = \{1, \dots, n\}$, $I_k \cap I_j = \emptyset$ and $|I_j| = |I_k| = I$ for all $j, k \in \{1, \dots, K\}$. Then data frames with $\{1, \dots, n\}/I_j$ individuals are created.

Value

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

See Also

[b_selection](#)

Examples

```
splitted_dataset = dataset_split(26, pbc2)
```

dij	<i>D matrix entries, used for the implementation of the local linear kernel</i>
-----	---

Description

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

Usage

```
dij(b, x, y, K)
```

Arguments

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

`y` A vector of sample data points.

`b` The bandwidth to use (a scalar).

`K` The kernel function to use.

Details

Implements the calculation 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} .

Epan

Epanechnikov kernel

Description

Implements the Epanechnikov kernel function

Usage

Epan(x)

Arguments

x 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 x .

get_alpha	<i>Marker-only hazard rate</i>
-----------	--------------------------------

Description

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

Usage

```
get_alpha(N, Y, b, br_X, K=Epan )
```

Arguments

N	A matrix made by <code>make_N</code> indicating the occurrences of events.
Y	A matrix made by <code>make_Y</code> indicating the exposure.
b	Bandwidth.
br_X	Vector of grid points for the marker values X .
K	Used kernel function.

Details

The function `get_alpha` implements the marker-only hazard estimator

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

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 `br_X`.

See Also

[h_xt](#)

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/( size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, breaks_X=br_X, breaks_s=br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid,
           size_X_grid, int_s, int_X, event_time = 'years', n)

b = 1.7
alpha<-get_alpha(N, Y, b, br_X, K=Epan )

```

get_h_x

Local constant future conditional hazard rate estimator

Description

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

Usage

```
get_h_x(data, marker_name, event_time_name, time_name, event_name, x, b)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
marker_name	The column name of the marker values in the data frame data .
event_time_name	The column name of the event times in the data frame data .
time_name	The column name of the times the marker values were observed in the data frame data .
event_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_x` implements the future local constant conditional hazard estimator

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

across a grid of possible time values t , where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see `get_alpha` for more details.

Value

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

See Also

`get_alpha`, `h_xt`

Examples

```
library(survival)
b = 10
x = 3
Landmark <- 2
pbcT1 <- pbc2[which(pbc2$year < Landmark & pbc2$years > Landmark),]
b=0.9

arg11l<-get_h_x1l(pbcT1, 'albumin', event_time_name='years',
                 time_name='year', event_name='status2', 2, 0.9)
arg11c<-get_h_x(pbcT1, 'albumin', event_time_name='years',
               time_name='year', event_name='status2', 2, 0.9)

#Calculate the local constant and local linear survival functions
br_s = seq(Landmark, 14, length=99)
sfalb21l<- make_sf( (br_s[2]-br_s[1])/4 , arg11l)
sfalb21c<- make_sf( (br_s[2]-br_s[1])/4 , arg11c)

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

#Plot the survival functions:
plot(br_s, sfalb21l, type="l", col=1, lwd=2, ylab="Survival probability", xlab="Marker level")
lines(br_s, sfalb21c, 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)
```

 get_h_xll

Local linear future conditional hazard rate estimator

Description

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

Usage

```
get_h_xll(data, marker_name, event_time_name, time_name, event_name, x, b)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
marker_name	The column name of the marker values in the data frame data .
event_time_name	The column name of the event times in the data frame data .
time_name	The column name of the times the marker values were observed in the data frame data .
event_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_xll](#) implements the local linear future conditional hazard estimator

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

across a grid of possible time values t , where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see [get_alpha](#) for more details.

Value

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

See Also

[get_alpha](#), [h_xt](#)

Examples

```

library(survival)
library(JM)

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

arg1ll<-get_h_xll(pbcT1, 'albumin', event_time_name = 'years', time_name = 'year',
                 event_name = 'status2', 2, 0.9)
arg1lc<-get_h_x(pbcT1, 'albumin', event_time_name = 'years', time_name = 'year',
                 event_name = 'status2', 2, 0.9)

#Calculate the local constant and local linear survival functions
br_s = seq(Landmark, 14, length=99)
sfalb2ll<- make_sf( (br_s[2]-br_s[1])/4 , arg1ll)
sfalb2lc<- make_sf( (br_s[2]-br_s[1])/4 , arg1lc)

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

#Plot the survival functions:
plot(br_s, sfalb2ll, type="l", col=1, lwd=2, ylab="Survival probability", xlab="Marker level")
lines(br_s, sfalb2lc, 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:
#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),]

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)

arg1<- get_h_x(pbcT1, 'serBilir', event_time_name = 'years',
              time_name = 'year', event_name = 'status2', 1, 10)
br_s1 = seq(Landmark, 14, length=99)
sfbil1<- make_sf( (br_s1[2]-br_s1[1])/5.4 , arg1)
kma1<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

plot(br_s1, sfbil1, 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)

## End(Not run)

```

g_xt

Computation of a key component for wild bootstrap

Description

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

Usage

```
g_xt(br_X, br_s, size_s_grid, int_X, x, t, b, Yi, Y, n)
```

Arguments

br_X	Marker value grid points that will be used in the evaluation.
br_s	Time value grid points that will be used in the evaluation.
size_s_grid	Size of the time grid.
int_X	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.
Yi	A matrix made by <code>make_Yi</code> indicating the exposure.
Y	A matrix made by <code>make_Y</code> 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

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
X = pbc2$serBilir
s = pbc2$year
br_s = seq(0, max(s), max(s)/(size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/(size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2$id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

t = 2
x = 2
b = 10

g_xt(br_X, br_s, size_s_grid, int_X, x, t, b, Yi, Y, n)

```

h_xt

Local constant future conditional hazard rate estimation at a single time point

Description

Calculates the future conditional hazard rate for a marker value x and a time value t .

Usage

```
h_xt(br_X, br_s, int_X, size_s_grid, alpha, x,t, b, Yi,n)
```

Arguments

br_X	Vector of grid points for the marker values X .
br_s	Vector of grid points for the time values s .
int_X	Position of the linear interpolated marker values on the marker grid.
size_s_grid	Size of the time grid.
alpha	Marker-hazard obtained from get_alpha .
x	Numeric value of the last observed marker value.
t	Numeric time value.
b	Bandwidth.
Yi	A matrix made by make_Yi indicating the exposure.
n	Number of individuals.

Details

Function [h_xt](#) implements the future conditional hazard estimator

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

where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see [get_alpha](#) for more details. The future conditional hazard is defined as

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

where T_i is the survival time and X_i the marker of individual i observed in the time frame $[0, T]$. Function [h_xt](#) uses an classic (unmodified) kernel function $K_b()$, e.g. the Epanechnikov kernel.

Value

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

References

[doi:10.1080/03461238.1998.10413997](https://doi.org/10.1080/03461238.1998.10413997)

See Also

[get_alpha](#)

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir

```

```

ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/( size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, breaks_X=br_X, breaks_s=br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

b = 1.7
alpha<-get_alpha(N, Y, b, br_X, K=Epan )

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

x = 2
t = 2
h_hat = h_xt(br_X, br_s, int_X, size_s_grid, alpha, x, t, b, Yi, n)

```

h_xtll	<i>Local linear future conditional hazard rate estimation at a single time point</i>
--------	--

Description

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

Usage

```
h_xtll(br_X, br_s, int_X, size_s_grid, alpha, x,t, b, Yi,n, Y)
```

Arguments

br_X	Vector of grid points for the marker values X .
br_s	Vector of grid points for the time values s .
int_X	Position of the linear interpolated marker values on the marker grid.
size_s_grid	Size of the time grid.
alpha	Marker-hazard obtained from get_alpha .
x	Numeric value of the last observed marker value.
t	Numeric time value.
b	Bandwidth.
Yi	A matrix made by make_Yi indicating the exposure.
n	Number of individuals.
Y	A matrix made by make_Y indicating the exposure.

Details

Function `h_xtll` implements the future conditional hazard estimator

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

where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see `get_alpha` for more details. The future conditional hazard is defined as

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

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

The function `h_xtll`, in the place of $K_b(\cdot)$ 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 $c_1 = (c_{11}, \dots, c_{1d})^T$, $D = (d_{ij})_{(d+1) \times (d+1)}$ 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).

Value

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

References

[doi:10.1080/03461238.1998.10413997](https://doi.org/10.1080/03461238.1998.10413997)

See Also

`get_alpha`, `dij`

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/(size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/(size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

```

```

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, breaks_X=br_X, breaks_s=br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

b = 1.7
alpha<-get_alpha(N, Y, b, br_X, K=Epan )

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s, size_s_grid, size_X_grid, int_s, int_X, 'years', n)

x = 2
t = 2
h_hat = h_xtll(br_X, br_s, int_X, size_s_grid, alpha, x, t, b, Yi, n, Y)

```

h_xt_vec

*Hqm estimator on the marker grid***Description**

Computes the hqm estimator on the marker grid.

Usage

```
h_xt_vec(br_X, br_s, size_s_grid, alpha, t, b, Yi, int_X, n)
```

Arguments

br_X	Marker value grid points that will be used in the evaluation.
br_s	Time value grid points that will be used in the evaluation.
size_s_grid	Size of the time grid.
alpha	Marker-hazard obtained from get_alpha .
t	Numeric value of the time the function should be evaluated.
b	Bandwidth.
Yi	A matrix made by make_Yi indicating the exposure.
int_X	Position of the linear interpolated marker values on the marker grid.
n	Number of individuals.

Details

The function implements the future conditional hazard estimator

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

for every x on the marker grid where X is the marker, Z is the exposure and $\alpha(z)$ is the marker-only hazard, see [get_alpha](#) for more details.

Value

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

Examples

```
# Longitudinal data example

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/( size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, br_X, br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s,
           size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)

b = 1.7
alpha <- get_alpha(N, Y, b, br_X, K=Epan )

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s,
             size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)

t = 2

h_xt_vec(br_X, br_s, size_s_grid, alpha, t, b, Yi, int_X, 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
arg1 <- get_h_x(pbcT1, 'albumin', event_time_name = 'years', time_name = 'year',
              event_name = 'status2', 2, b)

br_s2 = seq(Landmark, 14, length=99)
sfalb2 <- make_sf( (br_s2[2]-br_s2[1])/1.35 , arg1)
```

```

kma2<- survfit(Surv(years , status2) ~ 1, data = pbcT1)

#Plot the survival functions:
plot(br_s2, sfalb2, 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)

```

Kernels

Classical (unmodified) kernel and related functionals

Description

Implements the classical kernel function and related functionals

Usage

```

K_b(b,x,y, K)
xK_b(b,x,y, K)
K_b_mat(b,x,y, K)

```

Arguments

x	A vector of design points where the kernel will be evaluated.
y	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 `xK_b` 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_mat` is the vectorized version of `K_b`. It uses as inputs the vectors (X_1, \dots, X_n) and (Y_1, \dots, 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 xK_b and matrix outputs for K_b_mat.

lin_interpolate	<i>Linear interpolation</i>
-----------------	-----------------------------

Description

Implements a linear interpolation between observed marker values.

Usage

```
lin_interpolate(t, i, data_id, data_marker, data_time)
```

Arguments

t	A vector of time values where the function should be evaluated.
i	A vector of ids of individuals for whom the marker values should be interpolated.
data_id	The vector of ids from a data frame of time dependent variables.
data_marker	The vector of marker values from a data frame of time dependent variables.
data_time	The vector of time values from a data frame of time dependent variables.

Details

Given time points t_1, \dots, t_K and marker values m_1, \dots, m_J at different time points t_1^m, \dots, t_J^m , the function calculates a linear interpolation f with $f(t_i^m) = m_i$ at the time points t_1, \dots, t_K for all indicated individuals. Returned are then $(f(t_1), \dots, f(t_K))$. 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 $(f(t_1), \dots, f(t_K))$ as described above for every individual in the vector i .

Examples

```
size_s_grid <- 100
X = pbc2$serBilir
s = pbc2$year
br_s = seq(0, max(s), max(s)/(size_s_grid-1))
pbc2_id = to_id(pbc2)

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)
```


llK_b

*Local linear kernel***Description**

Implements the local linear kernel function.

Usage

```
llK_b(b, x, y, K)
```

Arguments

x A vector of design points where the kernel will be evaluated.
y 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 $c_1 = (c_{11}, \dots, c_{1d})^T$, $D = (d_{ij})_{(d+1) \times (d+1)}$ 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).

Value

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

References

[doi:10.1080/03461238.1998.10413997](https://doi.org/10.1080/03461238.1998.10413997)

llweights	<i>Local linear weight functions</i>
-----------	--------------------------------------

Description

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

Usage

```
sn.0(xin, xout, h, kfun)
sn.1(xin, xout, h, kfun)
sn.2(xin, xout, h, kfun)
```

Arguments

xin	Sample values.
xout	Grid points where the estimator will be evaluated.
h	Bandwidth parameter.
kfun	Kernel function.

Details

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

Value

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

make_N, make_Ni, make_Y, make_Yi	<i>Occurance and Exposure on grids</i>
----------------------------------	--

Description

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

Usage

```
make_N(data, data.id, breaks_X, breaks_s, ss, XX, delta)
make_Ni(breaks_s, size_s_grid, ss, delta, n)
make_Y(data, data.id, X_lin, breaks_X, breaks_s,
        size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
make_Yi(data, data.id, X_lin, breaks_X, breaks_s,
        size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
```

Arguments

data	A data frame of time dependent data points. Missing values are allowed.
data.id	An id data frame obtained from <code>to_id</code> .
breaks_X	Marker value grid points where the function will be evaluated.
breaks_s	Time value grid points where the function will be evaluated.
ss	Vector with event times.
XX	Vector of last observed marker values.
delta	0-1 vector of whether events happened.
size_s_grid	Size of the time grid.
size_X_grid	Size of the marker grid.
n	Number of individuals.
X_lin	Linear interpolation of observed marker values evaluated on the marker grid.
int_s	Position of the observed time values on the time grid.
int_X	Position of the linear interpolated marker values on the marker grid.
event_time	String of the column name with the event times.

Details

Implements matrices for the computation of integrals with occurrences 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_N` and `make_Y` return a matrix on the time grid and marker grid for occurrence and exposure, respectively, while `make_Ni` and `make_Yi` return a matrix on the time grid for every individual again for occurrence and exposure, respectively.

See Also

`h_xt`, `g_xt`, `get_alpha`

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/(size_s_grid-1))

```

```

br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, br_X, br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s,
            size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s,
              size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
Ni <- make_Ni(br_s, size_s_grid, ss, delta, n)

```

make_sf

Survival function from a hazard

Description

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

Usage

```
make_sf(step_size_s_grid, haz)
```

Arguments

`step_size_s_grid` Numeric value indicating the distance between two grid continuous grid points.

`haz` Vector of hazard values. Hazard rate must have been calculated on a time grid.

Details

The function `make_sf` calculates the survival function

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

where h is the hazard rate. Here, a discretisation 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_sf(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, transplantation, 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)
```

```
prep_boot
```

```
Precomputation for wild bootstrap
```

Description

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

Usage

```
prep_boot(g_xt, alpha, Ni, Yi, size_s_grid, br_X, br_s, t, b, int_X, x, n)
```

Arguments

<code>g_xt</code>	A vector obtained by <code>g_xt</code> .
<code>alpha</code>	A vector of the marker only hazard on the marker grid obtained by <code>get_alpha</code> .
<code>Ni</code>	A matrix made by <code>make_Ni</code> indicating the occurrence.
<code>Yi</code>	A matrix made by <code>make_Yi</code> indicating the exposure.
<code>size_s_grid</code>	Size of the time grid.
<code>br_X</code>	Vector of grid points for the marker values.
<code>br_s</code>	Time value grid points that will be used in the evaluation.
<code>t</code>	Numeric value of the time the function should be evaluated.
<code>b</code>	Bandwidth.
<code>int_X</code>	Position of the linear interpolated marker values on the marker grid.
<code>x</code>	Numeric value of the last observed marker value.
<code>n</code>	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 $V \sim N(0, 1)$,

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

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_xt` and the last one is the value of Γ .

See Also

[Conf_bands](#)

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/(size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/(size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, br_X, br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s,
```

```

        size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)

b = 1.7
alpha<-get_alpha(N, Y, b, br_X, K=Epan )

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s,
              size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
Ni  <- make_Ni(br_s, size_s_grid, ss, delta, n)

t = 2
x = 2

g = g_xt(br_X, br_s, size_s_grid, int_X, x, t, b, Yi, Y, n)

Boot_all = prep_boot(g, alpha, Ni, Yi, size_s_grid, br_X, br_s, t, b, int_X, x, n)
Boot_all

```

```
prep_cv
```

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

```

prep_cv(data, data.id, marker_name, event_time_name = 'years',
        time_name = 'year', event_name = 'status2', n, I, b)

```

Arguments

<code>data</code>	A data frame of time dependent data points. Missing values are allowed.
<code>data.id</code>	An id data frame obtained from to_id .
<code>marker_name</code>	The column name of the marker values in the data frame data .
<code>event_time_name</code>	The column name of the event times in the data frame data .
<code>time_name</code>	The column name of the times the marker values were observed in the data frame data .
<code>event_name</code>	The column name of the events in the data frame data .
<code>n</code>	Number of individuals.
<code>I</code>	Number of observations leave out for a K cross validation.
<code>b</code>	Bandwidth.

Details

The function splits the data set via `dataset_split` and calculates for every splitted data set the hqm estimator

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

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` 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

[b_selection](#)

Examples

```

pbc2_id = to_id(pbc2)
n = max(as.numeric(pbc2$id))
b = 1.5
I = 26
h_xt_mat_list = prep_cv(pbc2, pbc2_id, 'serBilir', 'years', 'year', 'status2', n, I, b)

```

 Q1

Bandwidth selection score Q1

Description

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

Usage

```
Q1(h_xt_mat, int_X, size_X_grid, n, Yi)
```

Arguments

<code>h_xt_mat</code>	A matrix of the estimator for the future conditional hazard rate for all values x and t .
<code>int_X</code>	Vector of the position of the observed marker values in the grid for marker values.
<code>size_X_grid</code>	Numeric value indicating the number of grid points for marker values.
<code>n</code>	Number of individuals.
<code>Yi</code>	A matrix made by <code>make_Yi</code> indicating the exposure.

Details

The function implements

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

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

Value

A value of the score Q1.

See Also

[b_selection](#)

Examples

```

pbc2_id = to_id(pbc2)
size_s_grid <- size_X_grid <- 100
n = max(as.numeric(pbc2$id))
s = pbc2$year
X = pbc2$serBilir
XX = pbc2_id$serBilir
ss <- pbc2_id$years
delta <- pbc2_id$status2
br_s = seq(0, max(s), max(s)/( size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)

int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

N <- make_N(pbc2, pbc2_id, br_X, br_s, ss, XX, delta)
Y <- make_Y(pbc2, pbc2_id, X_lin, br_X, br_s,
            size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)

b = 1.7
alpha <- get_alpha(N, Y, b, br_X, K=Epan )

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s,
              size_s_grid, size_X_grid, int_s, int_X, event_time = 'years', n)
Ni <- make_Ni(br_s, size_s_grid, ss, delta, n)

t = 2

h_xt_mat = t(sapply(br_s[1:99],
                    function(si){h_xt_vec(br_X, br_s, size_s_grid, alpha, t, b, Yi, int_X, n)}))

Q = Q1(h_xt_mat, int_X, size_X_grid, n, Yi)

```

R_K

*Bandwidth selection score R***Description**

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

Usage

```
R_K(h_xt_mat_list, int_X, size_X_grid, Yi, Ni, n)
```

Arguments

h_xt_mat_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.
int_X	Vector of the position of the observed marker values in the grid for marker values.
size_X_grid	Numeric value indicating the number of grid points for marker values.
Yi	A matrix made by make_Yi indicating the exposure.
Ni	A matrix made by make_Ni indicating the occurrence.
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

[b_selection](#)

Examples

```

pbc2_id = to_id(pbc2)
n = max(as.numeric(pbc2$id))
b = 1.5
I = 104
h_xt_mat_list = prep_cv(pbc2, pbc2_id, 'serBilir', 'years', 'year', 'status2', n, I, b)

size_s_grid <- size_X_grid <- 100
s = pbc2$year
X = pbc2$serBilir
br_s = seq(0, max(s), max(s)/( size_s_grid-1))
br_X = seq(min(X), max(X), (max(X)-min(X))/( size_X_grid-1))

ss <- pbc2_id$years
delta <- pbc2_id$status2

X_lin = lin_interpolate(br_s, pbc2_id$id, pbc2$id, X, s)
int_X <- findInterval(X_lin, br_X)
int_s = rep(1:length(br_s), n)

Yi <- make_Yi(pbc2, pbc2_id, X_lin, br_X, br_s,
              size_s_grid, size_X_grid, int_s, int_X, 'years', n)
Ni <- make_Ni(br_s, size_s_grid, ss, delta, n)

R = R_K(h_xt_mat_list, int_X, size_X_grid, Yi, Ni, n)
R

```

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_set)
```

Arguments

data_set 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 `pb2.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_set.id = to_id(pb2)
```

Index

auc.hqm, [2](#), [2](#), [4](#)

b_selection, [4](#), [5](#), [6](#), [9](#), [33–35](#)
b_selection_prep_g, [5](#), [5](#), [6](#)
bs.hqm, [3](#), [3](#)

Conf_bands, [7](#), [7](#), [31](#)

data, [4](#), [5](#), [7](#), [12](#), [14](#), [32](#)
dataset_split, [5](#), [8](#), [9](#), [33](#)
dij, [9](#), [20](#)

Epan, [10](#)

g_xt, [8](#), [16](#), [27](#), [30](#)
get_alpha, [11](#), [11](#), [13](#), [14](#), [18–21](#), [27](#), [30](#), [33](#)
get_h_x, [2](#), [3](#), [12](#), [13](#)
get_h_xll, [14](#), [14](#)

h_xt, [11](#), [13](#), [14](#), [17](#), [18](#), [26](#), [27](#), [31](#)
h_xt_vec, [21](#)
h_xtll, [19](#), [20](#)

I, [9](#)

K_b (Kernels), [23](#)
K_b_mat (Kernels), [23](#)
Kernels, [23](#)

lin_interpolate, [24](#)
llK_b, [25](#)
llweights, [26](#)

make_N, [11](#), [27](#)
make_N (make_N, make_Ni, make_Y, make_Yi), [26](#)
make_N, make_Ni, make_Y, make_Yi, [26](#)
make_Ni, [27](#), [30](#), [35](#)
make_Ni (make_N, make_Ni, make_Y, make_Yi), [26](#)
make_sf, [28](#), [28](#)

make_Y, [11](#), [16](#), [19](#), [27](#)
make_Y (make_N, make_Ni, make_Y, make_Yi), [26](#)
make_Yi, [6](#), [16](#), [18](#), [19](#), [21](#), [27](#), [30](#), [33](#), [35](#)
make_Yi (make_N, make_Ni, make_Y, make_Yi), [26](#)

pb2, [29](#)
prep_boot, [8](#), [30](#)
prep_cv, [5](#), [32](#)

Q1, [5](#), [33](#)

R_K, [5](#), [35](#)

sn.0 (llweights), [26](#)
sn.1 (llweights), [26](#)
sn.2 (llweights), [26](#)

to_id, [27](#), [32](#), [36](#), [36](#)

xK_b (Kernels), [23](#)