Package 'npcure'

October 13, 2022

Version 0.1-5

Date 2020-02-28

Title Nonparametric Estimation in Mixture Cure Models

Author Ignacio López-de-Ullibarri [aut, cre],
Ana López-Cheda [aut],
Maria Amalia Jácome [aut]

Maintainer Ignacio López-de-Ullibarri <ignacio.lopezdeullibarri@udc.es>

Depends R (>= 3.5.0)

Suggests KMsurv

Description Performs nonparametric estimation in mixture cure models, and significance tests for the cure probability. For details, see López-Cheda et al. (2017a) <doi:10.1016/j.csda.2016.08.002> and López-Cheda et al. (2017b) <doi:10.1007/s11749-016-0515-1>.

License GPL (>= 2)

Imports permute, stats, utils, zoo

NeedsCompilation yes

Encoding UTF-8

Repository CRAN

Date/Publication 2020-02-29 10:10:02 UTC

R topics documented:

---

npcure-package *Nonparametric Estimation in Mixture Cure Models*

Description

Performs nonparametric estimation in mixture cure models, and significance tests for the cure probability. For details, see López-Cheda et al. (2017a) <doi:10.1016/j.csda.2016.08.002> and López-Cheda et al. (2017b) <doi:10.1007/s11749-016-0515-1>.

Details

Index of help topics:

beran	Compute Beran's Estimator of the Conditional Survival
berancv	Compute the Cross-Validation Bandwidth for Beran's Estimator of the Conditional Survival
controlpars	Control Values for the Bootstrap or Cross-validation
hpilot	Compute the Pilot Bandwidth for the Nonparametric Estimators of Cure Probability and Latency
latency	Compute Nonparametric Estimator of the Conditional Latency
latencyhboot	Compute the Bootstrap Bandwidth for the Nonparametric Estimator of the Latency
npcure-package	Nonparametric Estimation in Mixture Cure Models
print.npcure	Print Method for Objects of Class 'npcure'
probcure	Compute Nonparametric Estimator of the Conditional Probability of Cure
probcurehboot	Compute the Bootstrap Bandwidth for the Nonparametric Estimator of the Cure Probability
summary.npcure	Summary Method for Objects of Class 'npcure'
testcov	Covariate Significance Test of the Cure Probability
testmz	Test of Maller-Zhou

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

López-Cheda, A., Cao, R., Jácome, M. A., Van Keilegom, I. (2017). Nonparametric incidence estimation and bootstrap bandwidth selection in mixture cure models. Computational Statistics & Data Analysis 105: 144–165. https://doi.org/10.1016/j.csda.2016.08.002.

López-Cheda, A., Jácome, M. A., Cao, R. (2017). Nonparametric latency estimation for mixture cure models. Test, 26: 353–376. https://doi.org/10.1007/s11749-016-0515-1.

`beran` *Compute Beran's Estimator of the Conditional Survival*

Description

This function computes the Beran nonparametric estimator of the conditional survival function.

Usage

```
beran(x, t, d, dataset, x0, h, local = TRUE, testimate = NULL,
conflevel = 0L, cvbootpars = if (conflevel == 0 && !missing(h)) NULL
else controlpars())
```

Arguments

`x`	If `dataset` is missing, a numeric object giving the covariate values. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
`t`	If `dataset` is missing, a numeric object giving the observed times. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
`d`	If `dataset` is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator.
`dataset`	An optional data frame in which the variables named in `x`, `t` and `d` are interpreted. If it is missing, `x`, `t` and `d` must be objects of the workspace.
`x0`	A numeric vector of covariate values where the survival estimates will be computed.
`h`	A numeric vector of bandwidths. If it is missing the default is to use the cross-validation bandwidth computed by the `berancv` function.
`local`	A logical value, `TRUE` by default, specifying whether local or global bandwidths are used.
`testimate`	A numeric vector specifying the times at which the survival is estimated. By default it is `NULL`, and then the survival is estimated at the times given by `t`.

`conflevel` A value controlling whether bootstrap confidence intervals (CI) of the survival are to be computed. With the default value, 0L, the CIs are not computed. If a numeric value between 0 and 1 is passed, it specifies the confidence level of the CIs.

`cvbootpars` A list of parameters controlling the bootstrap when computing the CIs of the survival: `B`, the number of bootstrap resamples, and `nnfrac`, the fraction of the sample size that determines the order of the nearest neighbor used for choosing a pilot bandwidth. If `h` is missing the list of parameters is extended to be the same used for computing the cross-validation bandwidth (see the help of `berancv` for details). The default is the value returned by the `controlpars` function called without arguments. In case the CIs are not computed and `h` is not missing the default is `NULL`.

Details

This function computes the kernel type product-limit estimator of the conditional survival function $S(t|x) = P(Y > t|X = x)$ under censoring, using the Nadaraya-Watson weights. The kernel used is the Epanechnikov. If the smoothing parameter h is not provided, then the cross-validation bandwidth selector in Geerdens et al. (2018) is used. The function is available only for one continuous covariate X.

Value

An object of S3 class 'npcure'. Formally, a list of components:

`type` The constant string "survival".

`local` The value of the `local` argument.

`h` The value of the `h` argument, unless this is missing, in which case its value is that of the cross-validation bandwidth.

`x0` The value of the `x0` argument.

`testim` The numeric vector of time values where the survival function is estimated.

`S` A list whose components are the estimates of the survival function for each one of the covariate values, i.e., those specified by the `x0` argument. The survival estimates are given at the times determined by the `testimate` argument.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Beran, R. (1981). Nonparametric regression with randomly censored survival data. Technical report, University of California, Berkeley.

Geerdens, C., Acar, E. F., Janssen, P. (2018). Conditional copula models for right-censored clustered event time data. Biostatistics, 19(2): 247-262. https://doi.org/10.1093/biostatistics/kxx034.

See Also

controlpars, berancv

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Survival estimates for covariate values 0, 0.5 using...
## ... (a) global bandwidths 0.3, 0.5, 1.
## By default, the estimates are computed at the observed times
x0 <- c(0, .5)
S1 <- beran(x, t, d, data, x0 = x0, h = c(.3, .5, 1), local = FALSE)

## Plot predicted survival curves for covariate value 0.5
plot(S1$testim, S1$S$h0.3$x0.5, type = "s", xlab = "Time", ylab =
"Survival", ylim = c(0, 1))
lines(S1$testim, S1$S$h0.5$x0.5, type = "s", lty = 2)
lines(S1$testim, S1$S$h1$x0.5, type = "s", lty = 3)
## The true survival curve is plotted for reference
p0 <- exp(2*x0[2])/(1 + exp(2*x0[2]))
lines(S1$testim, 1 - p0 + p0*pweibull(S1$testim, shape = .5*(x0[2] + 4),
lower.tail = FALSE), col = 2)
legend("topright", c("Estimate, h = 0.3", "Estimate, h = 0.5",
"Estimate, h = 1", "True"), lty = c(1:3, 1), col = c(rep(1, 3), 2))

## As before, but with estimates computed at fixed times 0.1, 0.2,...,1
S2 <- beran(x, t, d, data, x0 = x0, h = c(.3, .5, 1), local = FALSE,
testimate = .1*(1:10))

## ... (b) local bandwidths 0.3, 0.5.
## Note that the length of the covariate vector x0 and the bandwidth h
## must be the same.
S3 <- beran(x, t, d, data, x0 = x0, h = c(.3, .5), local = TRUE)

## ... (c) the cross-validation (CV) bandwidth selector (the default
## when the bandwidth argument is not provided).
## The CV bandwidth is searched in a grid of 150 bandwidths (hl = 150)
## between 0.2 and 2 times the standardized interquartile range
## of the covariate values (hbound = c(.2, 2)).
## 95% confidence intervals are also given.
S4 <- beran(x, t, d, data, x0 = x0, conflevel = .95, cvbootpars =
controlpars(hl = 150, hbound = c(.2, 2)))
```

```
## Plot of predicted survival curve and confidence intervals for
## covariate value 0.5
plot(S4$testim, S4$S$x0.5, type = "s", xlab = "Time", ylab = "Survival",
ylim = c(0, 1))
lines(S4$testim, S4$conf$x0.5$lower, type = "s", lty = 2)
lines(S4$testim, S4$conf$x0.5$upper, type = "s", lty = 2)
lines(S4$testim, 1 - p0 + p0 * pweibull(S4$testim, shape = .5*(x0[2] +
4), lower.tail = FALSE), col = 2)
legend("topright", c("Estimate with CV bandwidth", "95% CI limits",
"True"), lty = c(1, 2, 1), col = c(1, 1, 2))


## Example with the dataset 'bmt' in the 'KMsurv' package
## to study the survival of patients aged 25 and 40.
data("bmt", package = "KMsurv")
x0 <- c(25, 40)
S <- beran(z1, t2, d3, bmt, x0 = x0, conflevel = .95)
## Plot of predicted survival curves and confidence intervals
plot(S$testim, S$S$x25, type = "s", xlab = "Time", ylab = "Survival",
ylim = c(0, 1))
lines(S$testim, S$conf$x25$lower, type = "s", lty = 2)
lines(S$testim, S$conf$x25$upper, type = "s", lty = 2)
lines(S$testim, S$S$x40, type = "s", lty = 1, col = 2)
lines(S$testim, S$conf$x40$lower, type = "s", lty = 2, col = 2)
lines(S$testim, S$conf$x40$upper, type = "s", lty = 2, col = 2)
legend("topright", c("Age 25: Estimate", "Age 25: 95% CI limits",
"Age 40: Estimate", "Age 40: 95% CI limits"), lty = 1:2,
col = c(1, 1, 2, 2))
```

`berancv` *Compute the Cross-Validation Bandwidth for Beran's Estimator of the Conditional Survival*

Description

This function computes the cross-validation bandwidth for Beran's estimator of the conditional survival function.

Usage

```
berancv(x, t, d, dataset, x0, cvpars = controlpars())
```

Arguments

`x`	If `dataset` is missing, a numeric object giving the covariate values. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.

t	If dataset is missing, a numeric object giving the observed times. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
d	If dataset is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator in the data frame.
dataset	An optional data frame in which the variables named in x, t and indicator are interpreted. If it is missing, x, t and indicator must be objects of the workspace.
x0	A numeric vector of covariate values where the local cross-validation bandwidth will be computed.
cvpars	A list of parameters controlling the process of bandwidth selection. The default is the value returned by the controlpars function called without arguments. See the help for controlpars for details.

Details

The cross-validation (CV) bandwidth is taken as the largest local minimizer of the leave-one-out cross-validated criterion in Geerdens et al. (2018). Let $F^{(-i)}(t|x_i)$, $i = 1, \ldots, n$ be the Beran estimator obtained using the data points (x_j, t_j, d_j), $j = 1, \ldots, i-1, i+1, \ldots, n$. For the CV criterion, the differences $I(t_i \leq t_j) - F^{(-i)}(t_j|x_i)$ are computed only for the so-called 'useful pairs' of observed times (t_i, t_j). A pair (T_i, T_j) is useful if the value of the indicator $I(T_i \leq T_j)$ gives an unambiguous correct value for the indicator $I(Y_i \leq Y_j)$ which contains the corresponding true (possibly unknown) event times, see Geerdens et al. (2018) for details. Gannoun et al. (2007) apply a similar criterion to perform bandwidth selection for the Beran estimator, but they consider only the pairs of true (uncensored) event times. Note that the inclusion of useful pairs of observed times would be especially advantageous if the censoring rate is high.

Value

An object of S3 class 'npcure'. Formally, a list of components:

type	The constant character string c("Cross-validation bandwidth", "survival").
x0	Grid of covariate values.
h	Selected local cross-validation bandwidths.
hgrid	Grid of bandwidths used (optional).

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Gannoun, A., Saracco, J., Yu, K. (2007). Comparison of kernel estimators of conditional distribution function and quantile regression under censoring. Statistical Modeling, 7: 329-344. https://doi.org/10.1177/1471082X0700700404.

Geerdens, C., Acar, E. F., Janssen, P. (2018). Conditional copula models for right-censored clustered event time data. Biostatistics, 19(2): 247-262. https://doi.org/10.1093/biostatistics/kxx034.

See Also

beran, controlpars, hpilot

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Computation of cross-validation (CV) local bandwidth for Beran's
## estimator of survival for covariate values 0, 1, ...
#### ... with the default control parameters (passed through 'cvpars')
x0 <- c(0, 1)
hcv <- berancv(x, t, d, data, x0 = x0)

#### ... changing the default 'cvpars' by calling the 'controlpars()'
#### function:
#### (a) the CV local bandwidth is searched in a grid of 150 bandwidths
#### ('hl = 150') between 0.2 and 4 times the standardized interquartile
#### range of the covariate values of x ('hbound = c(.2, 4')
#### (b) all the grid bandwidths are saved ('hsave = TRUE')
hcv <- berancv(x, t, d, data, x0 = x0, cvpars = controlpars(hbound =
c(.2, 4), hl = 150, hsave = TRUE))

## Survival estimates for covariate values 0, 1, with CV local bandwidth
S1 <- beran(x, t, d, data, x0 = x0,  h = hcv$h)
## Plot predicted survival curves for covariate values 0, 1
plot (S1$testim, S1$S$x0, type = "s", xlab = "Time", ylab = "Survival",
ylim = c(0, 1))
lines(S1$testim, S1$S$x1, type = "s", lty = 2)
## The survival curves are displayed for reference
p0 <- exp(2*x0)/(1 + exp(2*x0))
lines(S1$testim, 1 - p0[1] + p0[1]*pweibull(S1$testim, shape = .5*(x0[1]
+ 4), lower.tail = FALSE), col = 2)
lines(S1$testim, 1 - p0[2] + p0[2]*pweibull(S1$testim, shape = .5*(x0[2]
+ 4), lower.tail = FALSE), lty = 2, col = 2)
legend("topright", c("Estimate, x = 0", "True, x = 0",
"Estimate, x = 1", "True, x = 1"), lty = c(1, 1, 2, 2), col = 1:2)
```

```
## Example with the dataset 'bmt' of the 'KMsurv' package to study the
## survival of patients aged 25 and 40.
data("bmt", package = "KMsurv")
x0 <- c(25, 40)
hcv <- berancv(z1, t2, d3, bmt, x0 = x0, cvpars = controlpars(hbound =
c(.2, 4), hl = 150, hsave = TRUE))
S <- beran(z1, t2, d3, bmt, x0 = x0, h = hcv$h, conflevel = .95)
## Plot of predicted survival curves and confidence intervals
plot(S$testim, S$S$x25, type = "s", xlab = "Time", ylab = "Survival",
ylim = c(0, 1))
lines(S$testim, S$conf$x25$lower, type = "s", lty = 2)
lines(S$testim, S$conf$x25$upper, type = "s", lty = 2)
lines(S$testim, S$S$x40, type = "s", lty = 1, col = 2)
lines(S$testim, S$conf$x40$lower, type = "s", lty = 2, col = 2)
lines(S$testim, S$conf$x40$upper, type = "s", lty = 2, col = 2)
legend("topright", c("Age 25: Estimate", "Age 25: 95% CI limits",
"Age 40: Estimate", "Age 40: 95% CI limits"), lty = 1:2,
col = c(1, 1, 2, 2))
```

`controlpars` *Control Values for the Bootstrap or Cross-validation*

Description

This function returns a list of values for the control parameters of the functions of the package that use the bootstrap or cross-validation.

Usage

```
controlpars(B = 999L, hbound = c(0.1, 3), hl = 100L, hsave =
FALSE, nnfrac = 0.25, fpilot = NULL, qt = 0.75, hsmooth = 1L, ...)
```

Arguments

`B`	An integer giving the number of bootstrap resamples.
`hbound`	A numeric vector of length 2 specifying the minimum (default, 0.1) and maximum (default, 3), respectively, of the initial grid of bandwidths as a multiple of the standardized interquartile range of the covariate values.
`hl`	A numeric value giving the length of the initial grid of bandwidths. The default is 100.
`hsave`	A logical value specifying if the grids of bandwidths must be saved as a component of the list returned by the `berancv`, `latencyboot` and `probcurehboot` functions. The default is `FALSE`.
`nnfrac`	A numeric value giving the fraction of the sample size that determines the order of the nearest neighbor used when choosing the pilot bandwidth. The default is 0.25.

fpilot	A function name or NULL. If NULL, the default, the hpilot function is used for computing a pilot bandwidth in case that one is needed. If not NULL, it must be the name of a user-defined function (given as a function name or as a character string). This function must necessarily have an argument x0, playing the same role than in hpilot, and must return a value of the same length than x0. If fpilot has more arguments, they are passed through the ... argument (see below).
qt	In bandwidth selection for the latency estimator (see latencyhboot), a numeric value specifiying the order of a quantile of the observed times. It determines the right boundary of the integration interval in the computation of the ISE (the left boundary is 0). The default is 0.75 (third quartile).
hsmooth	An integer. Its value controls whether the bandwidths selected by the latencyhboot and probcurehboot and berancv function should be smoothed, and, if so, the degree of smoothing. The smoothing consists in computing a centered moving average of the unsmoothed vector of bandwidths returned by default by latencyhboot and probcurehboot. The value of hsmooth is the number of terms used to compute the average. The default is 1L, which means that no smoothing is done.
...	Arguments of fpilot, if fpilot is not NULL.

Details

The output of controlpars is a list of control parameters required by the package functions which use the bootstrap or cross-validation. This is mainly the case of the berancv function, which computes a cross-validation bandwidth for Beran's estimator of survival, and of the latencyhboot and probcurehboot functions, which compute the bootstrap bandwidth selectors of the estimators of the latency and the probability of cure, respectively. Since these functions are indirectly called by, respectively, the beran, latency and probcure functions when their h argument is missing, the output of controlpars is also the expected (and default) way of passing to them the parameters for bandwidth selection.

Additionally, controlpars is used by beran, latency and probcure to set the number of bootstrap resamples and the value of nnfrac (see above) when confidence intervals are computed. The testcov function also uses it for setting the number of bootstrap resamples.

Value

A list whose components are the arguments of the function, their defaults being replaced with the values the function was called with.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

See Also

beran,berancv, hpilot, latency, latencyhboot, probcure, probcurehboot,testcov

`hpilot` *Compute the Pilot Bandwidth for the Nonparametric Estimators of Cure Probability and Latency*

Description

This function computes local pilot bandwidths for the nonparametric estimators of the probability of cure and the latency function.

Usage

```
hpilot(x, x0, nnfrac = 0.25)
```

Arguments

`x`	A numeric vector of observed covariate values.
`x0`	A numeric vector specifying a grid of covariate values.
`nnfrac`	A numeric value giving the fraction of the sample size that determines the order of the nearest neighbor. This is taken as `floor(length(x)*nnfrac)`. The default is 0.25.

Details

The function computes a data-driven local pilot bandwidth, required for the bootstrap bandwidth selector of the nonparametric estimators of the cure rate and latency functions. Simulations in López-Cheda et al. (2017) show that the choice of pilot bandwidth has small effect on the bootstrap bandwidth. This pilot bandwidth only depends on the sample size and the distribution of the covariate x (see López-Cheda, 2018):

$$g(x0) = 0.5(d_k^+(x0) + d_k^-(x0))(100/n)^{1/9}$$

where $d_k^+(x0)$ and $d_k^-(x0)$ are the distances from $x0$ to the k-th nearest neighbor on the right and the left, respectively, and k is a suitable integer depending on the sample size n. If there are not at least k neighbors on the right or on the left, we use $d_k^+(x0) = d_k^-(x0)$. The default value of k is $n/4$. The order $n^{-1/9}$ satisfies the conditions in Theorem 1 of Li and Datta (2001) and coincides with the order obtained by Cao and González-Manteiga (1993) for the uncensored case.

Value

A numeric vector of local pilot bandwidths corresponding to each one of the values of the grid of covariate values given by `x0`.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Cao R., González-Manteiga W. (1993). Bootstrap methods in regression smoothing. Journal of Nonparametric Statistics, 2: 379-388. https://doi.org/10.1080/10485259308832566.

Li, G., Datta, S. (2001). A bootstrap approach to nonparametric regression for right censored data. Annals of the Institute of Statistical Mathematics, 53(4): 708-729. https://doi.org/10.1023/A:1014644700806.

López-Cheda A. (2018). *Nonparametric Inference in Mixture Cure Models*. PhD dissertation, Universidade da Coruña. Spain.

López-Cheda, A., Cao, R., Jácome, M. A., Van Keilegom, I. (2017). Nonparametric incidence estimation and bootstrap bandwidth selection in mixture cure models. Computational Statistics & Data Analysis, 105: 144–165. https://doi.org/10.1016/j.csda.2016.08.002.

López-Cheda, A., Jácome, M. A., Cao, R. (2017). Nonparametric latency estimation for mixture cure models. TEST, 26: 353–376. https://doi.org/10.1007/s11749-016-0515-1.

See Also

controlpars, latencyhboot, probcurehboot

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Computing pilot bandwidths for covariate values -1, -0.8, ..., 1
## by taking the 5-th nearest neighbor
hpilot(data$x, x0 = seq(-1, 1, by = .2), nnfrac = .05)
```

---

latency *Compute Nonparametric Estimator of the Conditional Latency*

Description

This function computes the nonparametric estimator of the conditional latency function proposed by López-Cheda et al. (2017).

Usage

```
latency(x, t, d, dataset, x0, h, local = TRUE, testimate = NULL,
conflevel = 0L, bootpars = if (conflevel == 0 && !missing(h)) NULL else
controlpars())
```

Arguments

x	If dataset is missing, a numeric object giving the covariate values. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
t	If dataset is missing, a numeric object giving the observed times. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
d	If dataset is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. Ifdataset is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator.
dataset	An optional data frame in which the variables named in x, t and d are interpreted. If it is missing, x, t and d must be objects of the workspace.
x0	A numeric vector of covariate values where the latency estimates will be computed.
h	A numeric vector of bandwidths. If it is missing the default is to use the local bootstrap bandwidth computed by the latencyhboot function.
local	A logical value, TRUE by default, specifying whether local or global bandwidths are used.
testimate	A numeric vector specifying the times at which the latency is estimated. By default it is NULL, and then the latency is estimated at the times given by t.
conflevel	A value controlling whether bootstrap confidence intervals (CI) of the latency are to be computed. With the default value, 0L, the CIs are not computed. If a numeric value between 0 and 1 is passed, it specifies the confidence level of the CIs.
bootpars	A list of parameters controlling the bootstrap when computing the CIs of the latency: B, the number of bootstrap resamples, and nnfrac, the fraction of the sample size that determines the order of the nearest neighbor used for choosing a pilot bandwidth. If h is missing the list of parameters is extended to be the same used for computing the bootstrap bandwidth (see the help of latencyhboot for details). The default is the value returned by the controlpars function called without arguments. In case the CIs are not computed and h is not missing the default is NULL.

Details

The function computes the nonparametric estimator of the conditional latency $S_0(t|X = x_0) = P(Y > t|Y < \infty, X = x_0)$ proposed by López-Cheda et al. (2017). It is only available for a continuous covariate X.

Value

An object of S3 class 'npcure'. Formally, a list of components:

`type`	The constant string "latency".
`local`	The value of the `local` argument.
`h`	The value of the `h` argument, unless this is missing, in which case its value is that of the cross-validation bandwidth.
`x0`	The value of the `x0` argument.
`testim`	The numeric vector of time values where the latency function is estimated.
`S`	A list whose components are the estimates of the latency function for each one of the covariate values, i.e., those specified by the `x0` argument. The latency estimates are given at the times determined by the `testimate` argument.
`conf`	A list of components `lower` and `upper` giving the lower and the upper limits of the confidence intervals, respectively.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

López-Cheda, A., Jácome, M. A., Cao, R. (2017). Nonparametric latency estimation for mixture cure models. Test, 26: 353–376. https://doi.org/10.1007/s11749-016-0515-1.

See Also

controlpars, latencyhboot

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Latency estimates for covariate value 0.5...
x0 <- .5

## ... (a) with global bandwidths 0.5, 1, 2.
## By default, estimates are computed at the time values of 't'
S1 <- latency(x, t, d, data, x0 = x0, h = c(.5, 1, 2), local = FALSE)
plot(S1$testim, S1$S$h0.5$x0.5, type = "s", xlab = "Time", ylab =
```

```
"Latency", ylim = c(0, 1))
lines(S1$testim, S1$S$h1$x0.5, type = "s", lty = 2)
lines(S1$testim, S1$S$h2$x0.5, type = "s", lty = 3)
## The true latency curve is plotted as reference
lines(S1$testim, pweibull(S1$testim, shape = .5*(x0 + 4), lower.tail =
FALSE), col = 2)
legend("topright", c(paste("Estimate, ", c("h = 0.5", "h = 1",
"h = 2")), "True"), lty = c(1:3, 1), col = c(rep(1, 3), 2))

## As before, but with estimates computed at times 0.1, 0.2,..., 1
S2 <- latency(x, t, d, data, x0 = x0, h = c(.5, 1, 2), local = FALSE,
testimate = .1*(1:10))

## ... (b) with local bandwidth 2.
S3 <- latency(x, t, d, data, x0 = x0, h = 2, local = TRUE)
#### Note that with only one covariate value the results with
#### 'local = FALSE' and 'local = TRUE' coincide, but the output formats
#### differ slightly. Compare with
S3 <- latency(x, t, d, data, x0 = x0, h = 2, local = FALSE)

## ... (c) with local bootstrap bandwidth
b  <- latencyhboot(x, t, d, data, x0 = x0)
S4 <- latency(x, t, d, data, x0 = x0, h = b$h)

## ... (d) when the bandwidth is not specified, the bootstrap bandwidth
#### selector given by the 'latencyhboot' function is used by default.
#### The computation of 95% confidence intervals based on 1999 bootstrap
#### resamples is also illustrated
S5 <- latency(x, t, d, data, x0 = x0, conflevel = .95, bootpars =
controlpars(B = 1999))
plot(S5$testim, S5$S$x0, type = "s", xlab = "Time", ylab = "Latency",
ylim = c(0, 1))
lines(S5$testim, S5$conf$x0$lower, type = "s", lty = 2)
lines(S5$testim, S5$conf$x0$upper, type = "s", lty = 2)
lines(S5$testim, pweibull(S5$testim, shape = .5*(x0 + 4), lower.tail =
FALSE), col = 2)
legend("topright", c("Estimate", "95% CI limits", "True"), lty = c(1,
2, 1), col = c(1, 1, 2))


## Example with the dataset 'bmt' of the 'KMsurv' package
## to study the survival of the uncured patients aged 25 and 40
data("bmt", package = "KMsurv")
x0 <- c(25, 40)
S <- latency(z1, t2, d3, bmt, x0 = x0, conflevel = .95)
## Plot of predicted latency curves and confidence intervals
plot(S$testim, S$S$x25, type = "s", xlab = "Time (days)",
ylab = "Latency", ylim = c(0,1))
lines(S$testim, S$conf$x25$lower, type = "s", lty = 2)
lines(S$testim, S$conf$x25$upper, type = "s", lty = 2)
lines(S$testim, S$S$x40, type = "s", lty = 1, col = 2)
lines(S$testim, S$conf$x40$lower, type = "s", lty = 2, col = 2)
lines(S$testim, S$conf$x40$upper, type = "s", lty = 2, col = 2)
```

```
legend("topright", c("Age 25: Estimate", "Age 25: 95% CI limits",
"Age 40: Estimate","Age 40: 95% CI limits"), lty = 1:2,
col = c(1, 1, 2, 2))
```

`latencyhboot` *Compute the Bootstrap Bandwidth for the Nonparametric Estimator of the Latency*

Description

This function computes the bootstrap bandwidth for the nonparametric estimator of the conditional latency function.

Usage

```
latencyhboot(x, t, d, dataset, x0, bootpars = controlpars())
```

Arguments

`x`	If `dataset` is missing, a numeric object giving the covariate values. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
`t`	If `dataset` is missing, a numeric object giving the observed times. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
`d`	If `dataset` is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator in the data frame.
`dataset`	An optional data frame in which the variables named in `x`, `t` and `indicator` are interpreted. If it is missing, `x`, `t` and `indicator` must be objects of the workspace.
`x0`	A numeric vector of covariate values where the local bootstrap bandwidth will be computed.
`bootpars`	A list of parameters controlling the process of bandwidth selection. The default is the value returned by the `controlpars` function called without arguments.

Details

The function computes the bootstrap bandwidth selector for the nonparametric estimator of the conditional latency function at the covariate values given by `x0`. The bootstrap bandwidth is the minimizer of a bootstrap version of the Mean Integrated Squared Error (MISE) of the latency estimator, which is approximated by Monte Carlo by simulating a large number of bootstrap resamples, `B`. For each value of `x0`, the bootstrap MISE is the bootstrap expectation of the integrated difference between the value of the latency estimator computed with the bootstrap sample in a grid of bandwidths

and its value computed with the original sample and a pilot bandwidth. The bootstrap resamples are generated by using the simple weighted bootstrap resampling method, fixing the covariate. This method is equivalent to the simple weighted bootstrap of Li and Datta (2001). All the parameters typically involved in the bootstrap bandwidth selection process (number of bootstrap resamples, grid of bandwidths, pilot bandwidth, and right boundary of the integration interval for computing the MISE) are typically set through the `controlpars` function, whose output is passed to the `bootpars` argument. Also, the bootstrap bandwidths can be smoothed, and, if so, the smoothed bandwidths are returned as a separate component of the output. See the help of `controlpars` for details.

Value

An object of S3 class 'npcure'. Formally, a list of components:

`type`	The constant character string c("Bootstrap bandwidth", "latency").
`x0`	Grid of covariate values.
`h`	Selected local bootstrap bandwidths.
`hsmooth`	Smoothed selected local bootstrap bandwidths (optional)
`hgrid`	Grid of bandwidths used (optional).

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Li, G., Datta, S. (2001). A bootstrap approach to nonparametric regression for right censored data. Annals of the Institute of Statistical Mathematics, 53: 708–729. https://doi.org/10.1023/A:1014644700806.

López-Cheda, A., Jácome, M. A., Cao, R. (2017). Nonparametric latency estimation for mixture cure models. TEST, 26: 353–376. https://doi.org/10.1007/s11749-016-0515-1.

See Also

`controlpars`, `latency`

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
```

```
data <- data.frame(x = x, t = t, d = d)

## A vector of covariate values
vecx0 <- seq(-1.5, 1.5, by = .1)

## Computation of bootstrap local bandwidths at the values of 'vecx0'...
#### ... with the default control parameters
hb1 <- latencyhboot(x, t, d, data, x0 = vecx0)

#### ... changing the default 'bootpars' with 'controlpars()':
#### (a) 'B = 1999' (1999 bootstrap resamples are generated),
#### (b) 'hbound = c(.2, 4)' and 'hl = 50' (a grid of 50 bandwidths
#### between 0.2 and 4 times the standardized interquartile range of
#### the covariate values is built), and
#### (c) 'hsave = TRUE' (the grid bandwidths are saved), and

hb2 <- latencyhboot(x, t, d, data, x0 = vecx0, bootpars =
controlpars(B = 1999, hbound = c(.2, 4), hl = 50, hsave = TRUE))

## Estimates of the conditional latency at the covariate value x0 = 0
## with the selected bootstrap bandwidths
S1 <- latency(x, t, d, data, x0 = 0, h = hb1$h[hb1$x0 == 0])
S2 <- latency(x, t, d, data, x0 = 0, h = hb2$h[hb2$x0 == 0])

## A plot comparing the estimates with bootstrap bandwidths obtained
## with default and non-default 'bootpars'
plot(S1$testim, S1$S$x0, type = "s", xlab = "Time", ylab = "Latency",
ylim = c(0, 1))
lines(S2$testim, S2$S$x0, type = "s", lty = 2)
lines(S1$testim, pweibull(S1$testim, shape = .5*(0 + 4), lower.tail =
FALSE), col = 2)
legend("topright", c("Estimate with 'hb1'", "Estimate with 'hb2'",
"True"), lty = c(1, 2, 1), col = c(1, 1, 2))

## Example with the dataset 'bmt' of the 'KMsurv' package
## to study the survival of the uncured patients aged 25 and 40
data("bmt", package = "KMsurv")
x0 <- c(25, 40)
hb <- latencyhboot(z1, t2, d3, bmt, x0 = x0, bootpars = controlpars(B =
1999, hbound = c(.2, 4), hl = 150, hsave = TRUE))
S0 <- latency(z1, t2, d3, bmt, x0 = x0, hb$h, conflevel = .95)
## Plot of predicted latency curves and confidence bands
plot(S0$testim, S0$S$x25, type = "s", xlab = "Time (days)",
ylab = "Survival", ylim = c(0,1))
lines(S0$testim, S0$conf$x25$lower, type = "s", lty = 2)
lines(S0$testim, S0$conf$x25$upper, type = "s", lty = 2)
lines(S0$testim, S0$S$x40, type = "s", lty = 1, col = 2)
lines(S0$testim, S0$conf$x40$lower, type = "s", lty = 2, col = 2)
lines(S0$testim, S0$conf$x40$upper, type = "s", lty = 2, col = 2)
legend("topright", c("Age 25: Estimate", "Age 25: 95% CI limits",
"Age 40: Estimate", "Age 40: 95% CI limits"), lty = 1:2,
col = c(1, 1, 2, 2))
```

print.npcure *Print Method for Objects of Class 'npcure'*

Description

This function implements a print method for 'npcure' objects.

Usage

```
## S3 method for class 'npcure'
print(x, how, head = FALSE, ...)
```

Arguments

x	An object of class 'npcure'.
how	A character string with values "wide" or "long". If missing, the function itself chooses a convenient default.
head	A logical value that controls whether the function's output must be abbreviated (TRUE) or not (FALSE, the default).
...	Further optional arguments. Excepting for n, which controls how many lines are printed when head = TRUE, these are the arguments for the default method (i.e., print.default) of the print generic function.

Value

A formatted output.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

See Also

summary.npcure

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)
```

```
## Calling 'print()' with an object of class 'npcure' created by
## 'latency()'
S1 <- latency(x, t, d, data, x0 = c(0, .5), h = c(1, 1.5))

## In this case (latency estimation with local bandwidths and without
## confidence bands), the 'wide' format is used by default
S1
print(S1, how = "wide")
print(S1, how = "long")

## How to control the number of significant digits of the output, and
## how to abbreviate the output
print(S1, digits = 5, head = TRUE, n = 4)

## Calling 'print()' with a 'npcure' object created by 'probcure()'
q1 <- probcure(x, t, d, data, x0 = c(0, .5), h = c(.5, 1, 1.5), local =
FALSE, conflevel = .95)

## Only the 'long' format is available when confidence bands are
## computed
q1
print(q1, how = "long")
print(q1, how = "wide")
```

`probcure` *Compute Nonparametric Estimator of the Conditional Probability of Cure*

Description

This function computes the nonparametric estimator of the conditional probability of cure proposed by Xu and Peng (2014).

Usage

```
probcure(x, t, d, dataset, x0, h, local = TRUE, conflevel = 0L,
bootpars = if (conflevel == 0 && !missing(h)) NULL else controlpars())
```

Arguments

`x`	If `dataset` is missing, a numeric object giving the covariate values. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
`t`	If `dataset` is missing, a numeric object giving the observed times. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.

d	If dataset is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator in the data frame.
dataset	An optional data frame in which the variables named in x, t and d are interpreted. If it is missing, x, t and d must be objects of the workspace.
x0	A numeric vector of covariate values where the estimates of cure probability will be computed.
h	A numeric vector of bandwidths. If it is missing the default is to use the local bootstrap bandwidth computed by the probcurehboot function.
local	A logical value, TRUE by default, specifying whether local or global bandwidths are used.
conflevel	A value controlling whether bootstrap confidence intervals (CI) of the cure probability are to be computed. With the default value, 0L, the CIs are not computed. If a numeric value between 0 and 1 is passed, it specifies the confidence level of the CIs.
bootpars	A list of parameters controlling the bootstrap when computing either the CIs of the cure probability or the bootstrap bandwidth (if h is missing): B, the number of bootstrap resamples, and nnfrac, the fraction of the sample size that determines the order of the nearest neighbor used for choosing a pilot bandwidth. The default is the value returned by the controlpars function called without arguments. If the CIs are not computed and h is not missing the default is NULL.

Details

The function computes the nonparametric estimator of the conditional cure probability $q(X = x_0) \equiv 1 - p(X = x_0) = P(Y = \infty | X = x_0)$ proposed by Xu and Peng (2014), and also studied by López-Cheda et al (2017). It is only available for a continuous covariate X.

Value

An object of S3 class 'npcure'. Formally, a list of components:

estimate	The constant string "cure".
local	The value of the local argument.
h	The value of the h argument, unless this is missing, in which case its value is that of the bootstrap bandwidth.
x0	The value of the x0 argument.
q	A list with the estimates of the probability of cure.
conf	A list of components lower and upper giving the lower and the upper limits of the confidence intervals, respectively.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

López-Cheda, A., Cao, R., Jácome, M. A., Van Keilegom, I. (2017). Nonparametric incidence estimation and bootstrap bandwidth selection in mixture cure models. Computational Statistics & Data Analysis, 105: 144–165. https://doi.org/10.1016/j.csda.2016.08.002.

Xu, J., Peng, Y. (2014). Nonparametric cure rate estimation with covariates. The Canadian Journal of Statistics 42: 1-17. https://doi.org/10.1002/cjs.11197.

See Also

controlpars, probcurehboot

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = 0.5 * (x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t  <- ifelse(u < p, pmin(y, c), c) ## Observed times
d  <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Covariate values where cure probability is estimated
x0 <- seq(-1.5, 1.5, by = 0.1)

## Nonparametric estimates of cure probability at 'x0'...

## ... (a) with global bandwidths 1, 1.5, 2
q1 <- probcure(x, t, d, data, x0 = x0, h = c(1, 1.5, 2), local = FALSE)

#### Plot predicted cure probabilities at 'x0' for each bandwidth in 'h'
#### (the true cure probability is displayed for reference)
plot(q1$x0, q1$q$h1, type = "l", xlab = "Covariate", ylab =
"Cure probability", ylim = c(0, 1))
lines(q1$x0, q1$q$h1.5, lty = 2)
lines(q1$x0, q1$q$h2, lty = 3)
lines(q1$x0, 1 - exp(2*q1$x0)/(1 + exp(2*q1$x0)), col = 2)
legend("topright", c(paste("Estimate, ", c("h = 1", "h = 1.5",
"h = 2")), "True"), lty = c(1, 2, 3, 1), col = c(1, 1, 1, 2))

## ... (b) with local bandwidths (default)
#### (the vectors passed to 'x0' and 'h' must have the same length)
q2 <- probcure(x, t, d, data, x0 = x0, h = seq(1, 2.5, along = x0))

## ... (c) with local bootstrap bandwidths (based on 1999 booostrap
#### resamples). Besides, 95% confidence intervals are computed  and
#### smoothed (with a 15-th order moving average)
set.seed(1) ## Not needed, just for reproducibility.
```

```
hb <- probcurehboot(x, t, d, data, x0 = x0, bootpars = controlpars(B =
1999, hsmooth = 15))
q3 <- probcure(x, t, d, data, x0 = x0, h = hb$hsmooth, conflevel = .95,
bootpars = controlpars(B = 1999))


## ... (d) If the bandwidth is not specified, the local bootstrap
#### bandwidth is used (same results as in (c))
set.seed(1) ## Not needed, just for reproducibility.
q4 <- probcure(x, t, d, data, x0 = x0, conflevel = .95, bootpars =
controlpars(B = 1999, hsmooth = 15))

#### Plot of the estimated cure probabilities evaluated at 'x0'
#### (true cure rate displayed as reference)
plot (q4$x0, q4$q, type = "l", ylim = c(0, 1), xlab = "Covariate X",
ylab = "Cure probability")
lines(q4$x0, q4$conf$lower, lty = 2)
lines(q4$x0, q4$conf$upper, lty = 2)
lines(q4$x0, 1-exp(2 * q4$x0)/(1 + exp(2 * q4$x0)), col = 2)
legend("topright", c("Estimate", "95% CI limits", "True"),
lty = c(1, 2, 1), col = c(1, 1, 2))

## Example with the dataset 'bmt' in the 'KMsurv' package
## to study the probability of cure as a function of the age (z1).
data("bmt", package = "KMsurv")
x0 <- seq(quantile(bmt$z1, .05), quantile(bmt$z1, .95), length.out = 100)
q.age <- probcure(z1, t2, d3, bmt, x0 = x0, conflevel = .95, bootpars =
controlpars(B = 1999, hsmooth = 10))

## Plot of estimated cure probability and confidence intervals
par(mar = c(5, 4, 4, 5) + .1)
plot(q.age$x0, q.age$q, type = "l", ylim = c(0, 1), xlab =
"Patient age (years)", ylab = "Cure probability")
lines(q.age$x0, q.age$conf$lower, lty = 2)
lines(q.age$x0, q.age$conf$upper, lty = 2)
## The estimated density of age (z1) is added for reference
par(new = TRUE)
d.age <- density(bmt$z1)
plot(d.age, xaxt = "n", yaxt = "n", xlab = "", ylab = "", col = 2,
main = "", zero.line = FALSE)
mtext("Density", side = 4, col = 2, line = 3)
axis(4, ylim = c(0, max(d.age$y)), col = 2, col.axis = 2)
legend("topright", c("Estimate", "95% CI limits", "Covariate density"),
lty = c(1, 2, 1), col = c(1, 1, 2))
```

---

probcurehboot *Compute the Bootstrap Bandwidth for the Nonparametric Estimator of the Cure Probability*

Description

This function computes the bootstrap bandwidth for the nonparametric estimator of the conditional probability of cure.

Usage

```
probcurehboot(x, t, d, dataset, x0, bootpars = controlpars())
```

Arguments

`x`	If `dataset` is missing, a numeric object giving the covariate values. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
`t`	If `dataset` is missing, a numeric object giving the observed times. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
`d`	If `dataset` is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator in the data frame.
`dataset`	An optional data frame in which the variables named in `x`, `t` and `indicator` are interpreted. If it is missing, `x`, `t` and `indicator` must be objects of the workspace.
`x0`	A numeric vector of covariate values where the local bootstrap bandwidth will be computed.
`bootpars`	A list of parameters controlling the process of bandwidth selection. The default is the value returned by the `controlpars` function called without arguments.

Details

The function computes the bootstrap bandwidth selector for the nonparametric estimator of the cure probability at the covariate values given by `x0`. The bootstrap bandwidth is the minimizer of a bootstrap version of the Mean Squared Error (MSE) of the cure rate estimator, which is approximated by Monte Carlo by simulating a large number, `B`, of bootstrap resamples. The bootstrap MSE is the bootstrap expectation of the difference between the value of the cure rate estimator computed with the bootstrap sample in a grid of bandwidths and its value computed with the original sample and a pilot bandwidth. The bootstrap resamples are generated by using the simple weighted bootstrap resampling method, fixing the covariate. This method is equivalent to the simple weighted bootstrap of Li and Datta (2001). All the parameters involved in the bootstrap bandwidth selection process (number of bootstrap resamples, grid of bandwidths, and pilot bandwidth) are typically set through the `controlpars` function, whose output is passed to the `bootpars` argument. See the help of `controlpars` for details.

Given the local nature of bootstrap bandwidth selection, estimates obtained from sets of bootstrap bandwidths may sometimes look wiggly. To counter this behavior, the selected vector of bootstrap bandwidths can be smoothed by computing a moving average (its order being set by `controlpars`). Then, the smoothed bandwidths are contained in the `hsmooth` component of the returned value.

Value

An object of S3 class 'npcure'. Formally, a list of components:

type	The constant character string c("Bootstrap bandwidth", "cure").
x0	Grid of covariate values.
h	Selected local bootstrap bandwidths.
hsmooth	Smoothed selected local bootstrap bandwidths (optional)
hgrid	Grid of bandwidths used (optional).

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Li, G., Datta, S. (2001). A bootstrap approach to nonparametric regression for right censored data. Annals of the Institute of Statistical Mathematics, 53: 708-729. https://doi.org/10.1023/A:1014644700806.

López-Cheda, A., Cao, R., Jácome, M. A., Van Keilegom, I. (2017). Nonparametric incidence estimation and bootstrap bandwidth selection in mixture cure models. Computational Statistics & Data Analysis, 105: 144–165. https://doi.org/10.1016/j.csda.2016.08.002.

See Also

controlpars, probcure

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## A vector of covariate values
vecx0 <- seq(-1.5, 1.5, by = .1)

## Computation of bootstrap local bandwidth at the values of 'vecx0'...
#### ... with the default control parameters
set.seed(1) ## Not needed, just for reproducibility.
hb1 <- probcurehboot(x, t, d, data, x0 = vecx0)

#### ... changing the default 'bootpars' through 'controlpars()', with
```

```
#### arguments:
#### (a) 'B = 1999' (1999 bootstrap resamples are generated),
#### (b) 'hbound = c(.2, 4)' and 'hl = 50' (a grid of 50 bandwidths
#### between 0.2 and 4 times the standardized interquartilic range of
#### the covariate values is built),
#### (c) 'hsave = TRUE' (the grid bandwidths are saved), and
#### (d) 'hsmooth = 7' (the bootstrap bandwidths are smoothed by a
#### moving average of 7-th order)
set.seed(1) ## Not needed, just for reproducibility.
hb2 <- probcurehboot(x, t, d, data, x0 = vecx0, bootpars =
controlpars(B = 1999, hbound = c(.2, 4), hl = 50, hsave = TRUE, hsmooth
= 7))

## Estimates of the conditional probability of cure at the covariate
## values of 'vecx0' with the selected bootstrap bandwidths
q1 <- probcure(x, t, d, data, x0 = vecx0, h = hb1$h)
q2 <- probcure(x, t, d, data, x0 = vecx0, h = hb2$h)
q2sm <- probcure(x, t, d, data, x0 = vecx0, h = hb2$hsmooth)

## A plot comparing the estimates obtained with the bootstrap bandwidths
plot(q1$x0, q1$q, type = "l", xlab = "Covariate", ylab =
"Cure probability", ylim = c(0,1))
lines(q2$x0, q2$q, type = "l", lty = 2)
lines(q2sm$x0, q2sm$q, type = "l", lty = 3)
lines(q1$x0, 1 - exp(2*q1$x0)/(1 + exp(2*q1$x0)), col = 2)
legend("topright", c("Estimate with 'hb1'", "Estimate with 'hb2'",
"Estimate with 'hb2' smoothed", "True"), lty = c(1, 2, 3, 1), col = c(1,
1, 1, 2))


## Example with the dataset 'bmt' of the 'KMsurv' package
## to study the probability of cure as a function of the age (z1).
data("bmt", package = "KMsurv")
x0 <- seq(quantile(bmt$z1, .05), quantile(bmt$z1, .95), length.out =
100)
## This might take a while
hb <- probcurehboot(z1, t2, d3, bmt, x0 = x0, bootpars =
controlpars(B = 1999, hbound = c(.2, 2), hl = 50, hsave = TRUE, hsmooth
= 10))
q.age <- probcure(z1, t2, d3, bmt, x0 = x0, h = hb$h)
q.age.smooth <- probcure(z1, t2, d3, bmt, x0 = x0, h = hb$hsmooth)

## Plot of estimated cure probability
plot(q.age$x0, q.age$q, type = "l", ylim = c(0, 1), xlab =
"Patient age (years)", ylab = "Cure probability")
lines(q.age.smooth$x0, q.age.smooth$q, col = 2)
legend("topright", c("Estimate with h bootstrap",
"Estimate with smoothed h bootstrap"), lty = 1, col = 1:2)
```

`summary.npcure` *Summary Method for Objects of Class 'npcure'*

Description

This function prints a summary of a 'npcure' object.

Usage

```
## S3 method for class 'npcure'
summary(object, ...)
```

Arguments

`object`	An object of class 'npcure'.
`...`	Further optional arguments for the default method (i.e., `summary.default`) of the `summary` generic function.

Value

A compact summary showing the components of the object.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

See Also

`print.npcure`

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Calling 'summary()' with an object of class 'npcure' created by
## 'latency()'
S1 <- latency(x, t, d, data, x0 = c(0, .5), h = c(1, 1.5))
summary(S1)

## If needed, the number of significant digits of the output can be set
summary(S1, digits = 5)

## Calling 'summary()' with an object created by 'probcure()'
q1 <- probcure(x, t, d, data, x0 = c(0, .5), h = c(.5, 1, 1.5), local =
FALSE, conflevel = .95)
```

```
summary(q1)
```

---

testcov *Covariate Significance Test of the Cure Probability*

Description

This function carries out a significance test of covariate effect on the probability of cure.

Usage

```
testcov(x, t, d, dataset, bootpars = controlpars())
```

Arguments

x	If dataset is missing, an object giving the covariate values, whose type can be numeric, integer, factor or character. Ifdataset is a data frame, it is interpreted as the name of the variable corresponding to the covariate in the data frame.
t	If dataset is missing, a numeric object giving the observed times. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
d	If dataset is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator.
dataset	An optional data frame in which the variables named in x, t and d are interpreted. If it is missing, x, t and d must be objects of the workspace.
bootpars	A list of parameters controlling the test. Currently, the only accepted component is B, the number of bootstrap resamples. The default is the value returned by the controlpars function called without arguments.

Details

The function computes a statistic, based on the method by Delgado and González-Manteiga (2004), to test whether a covariate X has an effect on the cure probability (H_1: cure probability = q(x)) or not (H_0: cure probability = q). Since the cure rate can be written as the regression function $E(\nu|X = x) = q(x)$, where ν is the cure indicator, the procedure is carried out as a significance test in nonparametric regression. The challenge of the test is that the cure indicator ν is only partially observed due to censoring: for the uncensored observations $\nu = 0$, but it is unknown if a censored individual will be eventually cured or not (ν unknown). The approach consists in expressing the cure rate as a regression function with another response, not observable but estimable, say η. The estimated values of η depend on suitable estimates of the conditional distribution of the censoring variable and $\tau(x)$, an unknown time beyond which a subject could be considered as cured; see López-Cheda (2018). For the computation of the values of η, the censoring distribution is estimated unconditionally using the Kaplan-Meier product-limit estimator. The time τ is estimated by the largest uncensored time.

The test statistic is a weighted mean of the difference between the observations of η_i and the values of the conditional mean of η under the null hypothesis:

$$T_{n(x)} = 1/n \sum (\eta_i - \bar{\eta}) I(x_i \leq x)$$

.

For a qualitative covariate there is no natural way to order the values x_i. In principle, this makes impossible to compute the indicator function in the test statistic. The problem is solved by considering all the possible combinations of the covariate values, and by computing the test statistic for each 'ordered' combination. $T_{n(x)}$ is taken as the largest value of these test statistics.

The distribution of the test statistic under the null hypothesis is approximated by bootstrap, using independent naive resampling.

Value

An object of S3 class 'npcure'. Formally, a list of components:

`type`	The constant character string c("test", "Covariate").
`x`	The name of the covariate.
`CM`	The result of the Cramer-von Mises test: a list with components `statistic`, the test statistic, and `pvalue`, the p-value.
`KS`	The result of the Kolmogorov-Smirnov test: a list with components `statistic`, the test statistic, and `pvalue`, the p-value.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Delgado M. A., González-Manteiga W. (2001). Significance testing in nonparametric regression based on the bootstrap. Annals of Statistics, 29: 1469-1507.

López-Cheda A. (2018). *Nonparametric Inference in Mixture Cure Models*. PhD dissertation, Universidade da Coruña. Spain.

See Also

`controlpars`

Examples

```
## Some artificial data
set.seed(123)
n <- 50
x <- runif(n, -2, 2) ## Covariate values
y <- rweibull(n, shape = .5*(x + 4)) ## True lifetimes
c <- rexp(n) ## Censoring values
p <- exp(2*x)/(1 + exp(2*x)) ## Probability of being susceptible
u <- runif(n)
```

```
t <- ifelse(u < p, pmin(y, c), c) ## Observed times
d <- ifelse(u < p, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(x = x, t = t, d = d)

## Test of the significance of the covariate 'x'
testcov(x, t, d, data)

## Test carried out with 1999 bootstrap resamples (the default is 999)
testcov(x, t, d, data, bootpars = controlpars(B = 1999))


## How to apply the test repeatedly when there is more than one
## covariate...
## ... 'y' is another continuous covariate of the data frame 'data'
data$y <- runif(n, -1, 1)
namecovar <- c("x", "y")
## ... testcov is called from a 'for' loop
for (i in 1:length(namecovar)) {
   result <- testcov(data[, namecovar[i]], data$t, data$d)
   print(result)
}

## In the previous example, testcov() was called without using the
## argument 'dataset'. To use it, the 'for' must be avoided
testcov(x, t, d, data)
testcov(y, t, d, data)

## Non-numeric covariates can also be tested...
## ... 'z' is a nominal covariate of the data frame 'data'
data$z <- rep(factor(letters[1:5]), each = n/5)
testcov(z, t, d, data)

## Example with the dataset 'bmt' in the 'KMsurv' package
## to study the effect on the probability of cure of...
## ... (a) a continuous covariate (z1 = age of the patient)
data("bmt", package = "KMsurv")
set.seed(1) ## Not needed, just for reproducibility.
testcov(z1, t2, d3, bmt, bootpars = controlpars(B = 4999))

## ... (b) a qualitative covariate (z3 = patient gender)
set.seed(1) ## Not needed, just for reproducibility.
testcov(z3, t2, d3, bmt, bootpars = controlpars(B = 4999))
```

`testmz` *Test of Maller-Zhou*

Description

This function carries out the nonparametric test of Maller and Zhou (1992).

Usage

```
testmz(t, d, dataset)
```

Arguments

t	If dataset is missing, a numeric object giving the the observed times. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the observed times in the data frame.
d	If dataset is missing, an integer object giving the values of the uncensoring indicator. Censored observations must be coded as 0, uncensored ones as 1. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the uncensoring indicator.
dataset	An optional data frame in which the variables named in x, t and d are interpreted. If it is missing, x, t and d must be objects of the workspace.

Details

The function implements Maller and Zhou's (1992) method to test the null hypothesis $H_0 : \tau_{F_0} > \tau_G$ vs. $H_1 : \tau_{F_0} \leq \tau_G$, where τ_{F_0} and τ_G are the supports of, respectively, the distribution function of the survival time of the uncured and the distribution function of the censoring time.

Value

An object of S3 class 'npcure'. Formally, a list of components:

type	The constant character string c("test", "Maller-Zhou").
pvalue	The p-value of the test.
aux	A list of components: statistic, the test statistic, n the sample size, delta, the difference between the largest observed time t_n and the largest uncensored time $t_n^\star$, and interval, a vector giving the range between $\max(0, t_n^\star - \texttt{delta})$ and $t_n^\star$.

Author(s)

Ignacio López-de-Ullibarri [aut, cre], Ana López-Cheda [aut], Maria Amalia Jácome [aut]

References

Maller R. A., Zhou S. (1992). Estimating the proportion of immunes in a censored sample. Biometrika, 79: 731-739. https://doi.org/10.1093/biomet/79.4.731.

See Also

latency, probcure

Examples

```
## Some artificial data
set.seed(123)
n <- 50
y <- qweibull(runif(n)*pweibull(2, shape = 2), shape = 2) ## True lifetimes
c <- qexp(runif(n)*pexp(2.5)) ## Censoring values
u <- runif(n)
## Probability of being susceptible is constantly equal to .5
t <- ifelse(u < .5, pmin(y, c), c) ## Observed times
d <- ifelse(u < .5, ifelse(y < c, 1, 0), 0) ## Uncensoring indicator
data <- data.frame(t = t, d = d)

## Maller-Zhou test
testmz(t, d, data)
```

Index