Package: npcurePK (via r-universe)

September 17, 2024

Title Mixture Cure Model Estimation with Cure Status Partially Known

Version 1.0-2

Date 2023-05-05

Maintainer Wende Clarence Safari <wende.safari@lshtm.ac.uk>

Depends R (>= 4.2.0)

Description Performs nonparametric estimation in mixture cure models when the cure status is partially known. For details, see Safari et al (2021) <doi:10.1002/bimj.202100156>, Safari et al (2022) <doi:10.1177/09622802221115880> and Safari et al (2023) <doi:10.1007/s10985-023-09591-x>.

License GPL (>= 2)

Encoding UTF-8

Imports DescTools, data.table, parallel, doParallel, foreach, npcure

Suggests knitr, pinp, rmarkdown

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Wende Clarence Safari [aut, cre] (<https://orcid.org/0000-0003-4639-7552>), Ignacio López-de-Ullibarri [aut] (<https://orcid.org/0000-0002-3438-6621>), María Amalia Jácome [aut] (<https://orcid.org/0000-0001-7000-9623>)

Repository CRAN

Date/Publication 2023-05-07 10:40:02 UTC

Contents

`controlpars` *Control Values for the Bootstrap*

Description

This function returns a list of values for the control parameters of the functions that will be used for the bootstrap bandwidth selector.

Usage

```
controlpars(b = 100L, hbound = c(0.1, 3), hl = 30L, hgrid_save = FALSE,
            nnfrac = 0.25, fpilot = NULL, qt = 0.9, ncores = 1L,
            seed = NULL, ...)
```

Arguments

`b`	An integer giving the number of bootstrap resamples, `100` by default.
`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 `10`.
`hgrid_save`	A logical value specifying if the grids of bandwidths must be saved as a component of the list returned by the `prodlim_curepk_boot` function. 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 product-limit estimator, a numeric value specifying 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 lower limit is 0). The default is `0.9` (90th quantile).
`ncores`	The number of cores used in parallel computations.
`seed`	An optional integer passed to `set.seed()` to set the randomization seed.
`...`	Arguments of `fpilot`, if `fpilot` is not `NULL`.

Details

The output of `controlpars` is a list of control parameters required by the functions which use the bootstrap. This is mainly the case of the `prodlimcurePKhboot` function, which compute the bootstrap bandwidth selectors of the estimators of the survival, latency and the probability of cure. Since these functions are indirectly called by `prodlimcurePKhboot` function 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.

`latency_curepk` *Compute Estimator of Latency Function when Cure Status is Partially Known*

Description

This function computes the nonparametric estimator of the latency function when cure status is partially known proposed by Safari *et al* (2023).

Usage

```
latency_curepk(x, t, d, xinu, dataset, x0, h, local = TRUE,
               bootpars = if (!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.
`xinu`	If `dataset` is missing, an integer object giving the values of the cure status indicator. Uncensored and unknown censored observations must be coded as 0, known to be cured censored ones as 1. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the cure status indicator in the data frame.
`dataset`	An optional data frame in which the variables named in `x, t, d` and `xinu` are interpreted. If it is missing, `x, t, d` and `xinu` must be objects of the workspace.
`x0`	A numeric vector of covariate values where the estimates of the latency function will be computed.
`h`	A numeric matrix of bandwidths.

`local` A logical value, TRUE by default, specifying whether `local` or `global` bandwidths are used.

`bootpars` A list of parameters controlling the bootstrap when computing the bootstrap bandwidths of the product-limit estimator. `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. The default is the value returned by the `controlpars` function called without arguments.

Details

This function computes an estimator of the latency function $S_0(t \mid x) = P(Y > t \mid Y < \infty, X = x)$ when the cure status is partially known, introduced in Safari *et al* (2023). It is based on the relationship

$$S(t \mid x) = 1 - p(x) + p(x)S_0(t \mid x)$$

, using the kernel estimator of the cure rate $1 - p(x)$ in Safari *et al* (2022) and the survival function $S(t \mid x)$ in Safari *et al* (2021), with Nadaraya-Watson weights and bandwidth h_1 for the cure rate and h_2 for the survival function. If there are not individuals known to be cured (`xinu=0`), then the kernel estimator of the cure rate in López-Cheda *et al* (2017) is computed.

The latency estimator is computed with the pair of bandwidths in `h`. One bandwidth `h[1, ]` is used for the estimation of $1 - p(x)$ and another bandwidth `h[2, ]` is used for the estimation of $S(t \mid x)$. If the smoothing parameter `h` is not provided, then the bootstrap bandwidth selector in Safari *et al* (2023) is used. The kernel considered is Epanechnikov kernel. The function is available only for one continuous covariate X.

Value

A list of components:

`h` The numeric matrix `(2 x length(x0))` of bandwidths used in the estimation. One bandwidth `h[1, ]` is used for the estimation of $1 - p(x)$ and another bandwidth `h[2, ]` is used for the estimation of $S(t \mid x)$. If `h` argument is missing, the bootstrap bandwidth computed with the control parameters in argument `bootpars`.

`x0` The numeric vector of covariate values where the estimate of the latency function is computed.

`prob_cure` The estimate of the cure probability `1 - p(x0)` with bandwidth `h[1, ]`. It is a vector of the same length as `x0`.

`t` The observed time values, where the latency function is estimated.

`surv` Estimates of the survival function for each one of the covariate values specified by the `x0` argument and the bandwidths in `h[2, ]`. It is a matrix of dimension $n \times length(x0)$ if local bandwidths or bootstrap bandwidths are used, or an array for global bandwidths instead.

`latency` Estimates of the latency for each one of the covariate values specified by the `x0` argument and the bandwidths in `h`. It is a matrix of dimension $n \times length(x0)$ if local bandwidths or bootstrap bandwidths are used, or an array for global bandwidths instead.

References

López-Cheda, A., Jácome, M.A., Cao, R. (2017). Nonparametric latency estimation for mixture cure models. TEST 26:353-376. doi:10.1007/s11749016051 51.

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2021). A product-limit estimator of the conditional survival function when cure status is partially known. Biometrical Journal, 63(5): 984-1005. doi:10.1002/bimj.202000173.

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2022). Nonparametric kernel estimation of the probability of cure in a mixture cure model when the cure status is partially observed. Statistical Methods in Medical Research, 31(11):2164-2188. doi:10.1177/09622802221115880.

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2023). Latency function estimation under the mixture cure model when the cure status is available. Lifetime Data Analysis. doi:10.1007/s10985-02309591x.

See Also

controlpars

Examples

```
library(npcurePK)

## Data-generating function
## n: sample size
## x_cov_range: range of covariate values
## p_knowncure: probability of known cure
data_gen <- function(n, x_cov_range, p_knowncure) {
    ## probability of being susceptible
    p0 <- function(x) exp(2*x)/(1 + exp(2*x))
    ## covariate values
    x <- runif(n, x_cov_range[1], x_cov_range[2])
    ## censoring times
    c <- rexp(n)
    u <- runif(n)
    v <- runif(n)
    data <- data.frame(matrix(0, nrow = n, ncol = 4L,
                              dimnames = list(NULL, c("x", "t", "d", "xinu"))))
    data[, "x"] <- x
    for (i in 1:n) {
        if (u[i] > p0(x[i])) {
            ## Cured individuals (all of them are censored: Yi = infty,
            ## Ti = Ci, delta = 0, nu = 1)
            data[i, "t"] <- c[i]
            if (v[i] < p_knowncure)
                data[i, "xinu"]  <- 1
        } else {
            ## Uncured individual (Yi < infty, Ti = min(Yi, Ci),
            ## delta = 1(Yi < Ci), nu = 0)
            ## Uncensored individual (d = 1): cure status is
            ## observed (xi = 1), i.e., xinu = 0
            ## Censored individual (d = 0): cure status is
```

```
            ## unknown (xi = 0), i.e., xi.nu = 0
            y <- rweibull(1, shape = 0.5 * (x[i] + 4))
            data[i, "t"]  <- ifelse(v[i] < p_knowncure, y, min(y, c[i]))
            if (data[i, "t"] == y) data[i, "d"] <- 1
        }
    }
    return(data)
}

set.seed(123)
data <- data_gen(n = 100, x_cov_range = c(-2, 2), p_knowncure = 0.8)

## Latency estimates for one single covariate value x0 = 0 and using...
x0 <- 0

## ... (a) one single fixed bandwidth h = [1.1, 1]
## h[1,] = 1.1 is used for estimating p(x) at x0 = 0
## h[2,] = 1 is used for estimating S(t|x) at x0 = 0
## The latency estimates are saved in an array (n × 1)
S0_1 <- latency_curepk(x, t, d, xinu, data, x0 = 0,
                       h = matrix(c(1.1, 1), nrow = 2, ncol = 1, byrow = TRUE),
                       local = TRUE)
## Plot predicted latency curve for covariate value x0 = 0 and bandwidths
## h = [1.1, 1]
plot(S0_1$t, S0_1$latency, type = "s", xlab = "Time",
     ylab = "Latency function", ylim = c(0, 1))
## The true latency function is included as reference
lines(S0_1$t, 1 - pweibull(S0_1$t, shape = 0.5 * (x0 + 4)))

## ... (b) two fixed bandwidths h = [1.1, 1] and h = [1.5, 2]
## One estimate of the latency S0(t|x0 = 0) is obtained using h[1, 1] = 1.1
## for estimating p(x) and h[2,1] = 1 for estimating S(t|x)
## Second estimate of the latency S0(t|x0 = 0) is obtained using h[1, 2] = 1.5
## using h[1,2] = 1.5 for estimating p(x) and h[2,2] = 2 for estimating S(t|x)
## The estimates are saved in an array (n × 2)
S0_2 <- latency_curepk(x, t, d, xinu, data, x0 = c(0, 0),
                       h = matrix(c(1.1, 1, 1.5, 2), nrow = 2, ncol = 2,
                                  byrow = FALSE), local = TRUE)
## Plot predicted latency curve for covariate value x0 = 0 and bandwidths
## h = [1.1, 1] and and h = [1.5, 2]
plot(S0_2$t, S0_2$latency[, 1], type = "s", xlab = "Time",
     ylab = "Latency function", ylim = c(0, 1))
lines(S0_2$t, S0_2$latency[, 2], type = "s", lwd = 2)
## The true latency function is included as reference
lines(S0_2$t, 1 - pweibull(S0_2$t, shape = 0.5 * (x0 + 4)))


    ## ... (c) with the bootstrap bandwidth selector (the default when the
    ## bandwidth argument h is not provided).
    ## The bootstrap bandwidth is searched with parallel computation
    ## (ncores = 2) in a grid of 9 bandwidths (hl = 9) between 0.2 and 2 times
    ## the standardized interquartile range of the covariate values
    ## (hbound = c(0.1, 2)). The latency estimates are saved in an array of
```

```
## dimension (n, 1)
library(doParallel)
(S0_3 <- latency_curepk(x, t, d, xinu, data, x0 = 0,
                        bootpars = controlpars(b = 50, hl = 9,
                        hbound = c(0.1, 2), ncores = 2)))
plot(S0_3$t, S0_3$latency[, 1], type = "s", xlab = "Time",
     ylab = "Latency function", ylim = c(0, 1))
## The true latency function is included as reference
lines(S0_3$t, 1 - pweibull(S0_3$t, shape = 0.5 * (x0 + 4)))
```

`npcurePK-internal` *Internal npcurePK Functions*

Description

Internal functions of the package.

Details

Internal functions not to be called by the user.

`prob_curepk` *Compute Estimator of Cure Probability when Cure Status is Partially Known*

Description

This function computes the nonparametric estimator of the cure probability when cure status is partially known proposed by Safari *et al* (2022).

Usage

```
prob_curepk(x, t, d, xinu, dataset, x0, h, local = TRUE,
            bootpars = if (!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.
xinu	If dataset is missing, an integer object giving the values of the cure status indicator. Uncensored and unknown censored observations must be coded as 0, known to be cured censored ones as 1. If dataset is a data frame, it is interpreted as the name of the variable corresponding to the cure status indicator in the data frame.
dataset	An optional data frame in which the variables named in x, t, d and xinu are interpreted. If it is missing, x, t, d and xinu must be objects of the workspace.
x0	A numeric vector of covariate values where the estimates of the cure probability will be computed.
h	A numeric vector of bandwidths.
local	A logical value, TRUE by default, specifying whether local or global bandwidths are used.
bootpars	A list of parameters controlling the bootstrap when computing the bootstrap bandwidths of the cure probability estimator. 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. The default is the value returned by the controlpars function called without arguments.

Details

Mixture cure model writes the conditional survival function $S(t \mid x) = P(Y > t \mid X = x)$ as

$$S(t \mid x) = 1 - p(x) + p(x)S_0(t \mid x)$$

where $1 - p(x) = P(Y = \infty \mid X = x)$ is the probability of cure.

This function computes the kernel estimator of the probability of cure $1 - p(x)$ in Safari *et al* (2022). It is based on the previous relationship and the generalized product-limit estimator of the conditional survival function $S(t \mid x)$ in Safari *et al* (2021), using the Nadaraya-Watson weights, when the cure status is partially known. If there are not individuals known to be cured (xinu=0), then the nonparametric estimator of the cure rate in López-Cheda *et al* (2017) is computed.

The Epanechnikov kernel is used. If the smoothing parameter h is not provided, then the bootstrap bandwidth selector in Safari *et al* (2022) is used. The function is available only for one continuous covariate X.

Value

A list of components:

h	The numeric vector of bandwidths used in the estimation. If h argument is missing, the bootstrap bandwidth computed with the control parameters in argument bootpars.

x0	The numeric vector of covariate values where the estimate of the cure probability is computed.
prob_cure	The estimate of the cure probability 1-p(x0) with bandwidth h. It is a vector of the same length as x0.

References

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

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 and Data Analysis 105:144-165. doi:10.1016/j.csda.2016.08.002.

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2021). A product-limit estimator of the conditional survival function when cure status is partially known. Biometrical Journal, 63(5): 984-1005. doi:10.1002/bimj.202000173.

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2022). Nonparametric kernel estimation of the probability of cure in a mixture cure model when the cure status is partially observed. Statistical Methods in Medical Research, 31(11):2164-2188. doi:10.1177/09622802221115880.

See Also

controlpars

Examples

```
library(npcurePK)

## Data-generating function
## n: sample size
## x_cov_range: range of covariate values
## p_knowncure: probability of known cure
data_gen <- function(n, x_cov_range, p_knowncure) {
    ## probability of being susceptible
    p0 <- function(x) exp(2*x)/(1 + exp(2*x))
    ## covariate values
    x <- runif(n, x_cov_range[1], x_cov_range[2])
    ## censoring times
    c <- rexp(n)
    u <- runif(n)
    v <- runif(n)
    data <- data.frame(matrix(0, nrow = n, ncol = 4L,
                              dimnames = list(NULL, c("x", "t", "d", "xinu"))))
    data[, "x"] <- x
    for (i in 1:n) {
        if (u[i] > p0(x[i])) {
            ## Cured individuals (all of them are censored: Yi = infty,
            ## Ti = Ci, delta = 0, nu = 1)
            data[i, "t"] <- c[i]
            if (v[i] < p_knowncure)
                data[i, "xinu"]  <- 1
```

```
        } else {
            ## Uncured individual (Yi < infty, Ti = min(Yi, Ci),
            ## delta = 1(Yi < Ci), nu = 0)
            ## Uncensored individual (d = 1): cure status is
            ## observed (xi = 1), i.e., xinu = 0
            ## Censored individual (d = 0): cure status is
            ## unknown (xi = 0), i.e., xi.nu = 0
            y <- rweibull(1, shape = 0.5 * (x[i] + 4))
            data[i, "t"]  <- ifelse(v[i] < p_knowncure, y, min(y, c[i]))
            if (data[i, "t"] == y) data[i, "d"] <- 1
        }
    }
    return(data)
}

set.seed(123)
data <- data_gen(n = 100, x_cov_range = c(-2, 2), p_knowncure = 0.8)

## Cure rate estimates for one single covariate value x0 = 0 and using ...
## ... (a) one single fixed bandwidth h = 0.5
p1 <- prob_curepk(x, t, d, xinu, data, x0 = 0,
                  h = 0.5, local = TRUE)

## ... (b) a vector of bandwidths h = c(0.25, 0.5, 0.75, 1)
p2 <- prob_curepk(x, t, d, xinu, data, x0 = c(0, 0, 0, 0),
                  h = c(0.25, 0.5, 0.75, 1), local = TRUE)

## ... (c) a bootstrap bandwidth (the default when the bandwidths
## argument h is not provided).
## The bootstrap bandwidth is searched in a grid of 10 bandwidths (hl = 10)
## between 0.2 and 2 times the standardized interquartile range of the
## covariate values (hbound = c(0.1, 3)).
(p3 <- prob_curepk(x, t, d, xinu, data, x0 = 0))
## Equivalently

    (p3 <- prob_curepk(x, t, d, xinu, data, x0 = 0,
                       bootpars = controlpars(hl = 10, hbound = c(0.1, 3))))


## Cure rate estimates for a vector of 20 covariate values and using ...
x0 = seq(from = min(data$x), to = max(data$x), length.out = 15)
## ... (a) one single fixed bandwidth h = 0.5
p4 <- prob_curepk(x, t, d, xinu, data, x0 = x0, h = 0.5, local = FALSE)
## Plot predicted cure probabilities for covariate values x0 and bandwidths
## h = 0.5
plot(p4$x0, p4$prob_cure, xlab = "Covariate X", type = "l",
     ylab = "Probability of cure", ylim = c(0, 1))
## The true cure rate is included as reference
lines(p4$x0, 1 - exp(2*x0)/(1 + exp(2*x0)), lwd = 2)

## ... (b) a vector of bandwidths h = c(0.5, 0.75, 1)
p5 <- prob_curepk(x, t, d, xinu, data, x0 = x0, h = c(0.5, 0.75, 1),
                  local = FALSE)
```

```
## Plot predicted cure probabilities for covariate values x0 and bandwidths
## h = 0.5
plot(p5$x0, p5$prob_cure[1, ], xlab = "Covariate X", type = "l",
     ylab = "Probability of cure", ylim = c(0, 1))
## The estimates with bandwidth h = 0.75 and h = 1 are added
lines(p5$x0, p5$prob_cure[2, ])
lines(p5$x0, p5$prob_cure[3, ])
## The true cure rate is included as reference
lines(p5$x0, 1 - exp(2*x0)/(1 + exp(2*x0)), lwd = 2)


    ## ... (c) the bootstrap bandwidth
    (p6 <- prob_curepk(x, t, d, xinu, data, x0 = x0,
                       bootpars = controlpars(b = 50, ncores = 2, seed = 123)))
    ## Plot predicted cure probabilities for covariate values x0 and bootstrap
    ## bandwidths
    plot(p6$x0, p6$prob_cure, xlab = "Covariate X", type = "l",
         ylab = "Probability of cure", ylim = c(0, 1))
    ## The true cure rate is included as reference
    lines(p6$x0, 1 - exp(2*x0)/(1 + exp(2*x0)), lwd = 2)
```

`prodlim_curepk` *Compute Product-Limit Estimator of Conditional Survival Function when Cure Status is Partially Known*

Description

This function computes the nonparametric estimator of the conditional survival function when cure status is partially known proposed by Safari *et al* (2021).

Usage

```
prodlim_curepk(x, t, d, xinu, dataset, x0, h, local = TRUE,
               bootpars = if (!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.

`xinu` If `dataset` is missing, an integer object giving the values of the cure status indicator. Uncensored and unknown censored observations must be coded as 0, known to be cured censored ones as 1. If `dataset` is a data frame, it is interpreted as the name of the variable corresponding to the cure status indicator in the data frame.

`dataset` An optional data frame in which the variables named in `x, t, d` and `xinu` are interpreted. If it is missing, `x, t, d` and `xinu` must be objects of the workspace.

`x0` A numeric vector of covariate values where the estimates of the conditional survival function will be computed.

`h` A numeric vector of bandwidths.

`local` A logical value, TRUE by default, specifying whether `local` or `global` bandwidths are used.

`bootpars` A list of parameters controlling the bootstrap when computing the bootstrap bandwidths of the product-limit estimator. `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. The default is the value returned by the `controlpars` function called without arguments.

Details

Mixture cure model writes the conditional survival function $S(t \mid x) = P(Y > t \mid X = x)$ as $S(t \mid x) = 1 - p(x) + p(x)S_0(t \mid x)$. This function computes the generalized product-limit estimator of the conditional survival function $S(t \mid x)$, using the Nadaraya-Watson weights, when the cure status is partially known, introduced in Safari *et al* (2021). If there are not individuals known to be cured (`xinu=0`), then the usual generalized product-limit estimator in Beran (1981) is computed.

The Epanechnikov kernel is used. If the smoothing parameter `h` is not provided, then the bootstrap bandwidth selector in Safari *et al* (2021) is used. The function is available only for one continuous covariate X.

Value

A list of components:

`h` The numeric vector of bandwidths used in the estimation. If `h` argument is missing, the bootstrap bandwidth computed with the control parameters in argument `bootpars`.

`x0` The numeric vector of covariate values where the estimate of the conditional survival function is computed.

`t` The observed time values, where the conditional survival function is estimated.

`surv` Estimates of the survival function for each one of the covariate values specified by the `x0` argument and the bandwidths in `h`. It is a matrix of dimension $n \times length(x0)$ if local bandwidths or bootstrap bandwidths are used, or an array of dimension $n \times length(x0) \times length(h)$ for global bandwidths instead.

References

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

Safari, W. C., López-de-Ullibarri I., Jácome, M. A. (2021). A product-limit estimator of the conditional survival function when cure status is partially known. Biometrical Journal, 63(5): 984-1005. doi:10.1002/bimj.202000173.

See Also

controlpars

Examples

```
library(npcurePK)

## Data-generating function
## n: sample size
## x_cov_range: range of covariate values
## p_knowncure: probability of known cure
data_gen <- function(n, x_cov_range, p_knowncure) {
  ## probability of being susceptible
  p0 <- function(x) exp(2*x)/(1 + exp(2*x))
  ## covariate values
  x <- runif(n, x_cov_range[1], x_cov_range[2])
  ## censoring times
  c <- rexp(n)
  u <- runif(n)
  v <- runif(n)
  data <- data.frame(matrix(0, nrow = n, ncol = 4L,
                            dimnames = list(NULL, c("x", "t", "d", "xinu"))))
  data[, "x"] <- x
  for (i in 1:n) {
    if (u[i] > p0(x[i])) {
      ## Cured individuals (all of them are censored: Yi = infty,
      ## Ti = Ci, delta = 0, nu = 1)
      data[i, "t"] <- c[i]
      if (v[i] < p_knowncure)
        data[i, "xinu"]  <- 1
    } else {
      ## Uncured individual (Yi < infty, Ti = min(Yi, Ci),
      ## delta = 1(Yi < Ci), nu = 0)
      ## Uncensored individual (d = 1): cure status is
      ## observed (xi = 1), i.e., xinu = 0
      ## Censored individual (d = 0): cure status is
      ## unknown (xi = 0), i.e., xi.nu = 0
      y <- rweibull(1, shape = 0.5 * (x[i] + 4))
      data[i, "t"]  <- ifelse(v[i] < p_knowncure, y, min(y, c[i]))
      if (data[i, "t"] == y) data[i, "d"] <- 1
    }
  }
  return(data)
```

```
}

set.seed(123)
data <- data_gen(n = 100, x_cov_range = c(-2, 2), p_knowncure = 0.8)

## Covariate values where the conditional survival function is estimated
x0 <- c(0, 0.5)

## Survival estimates for covariate values x0 = c(0, 0.5)
## ... (a) with 3 global bandwidths (0.5, 1, 1.25)
## The survival function S(t|x) is estimated for each value of x0 with the three
## bandwidths (local == FALSE).
## The estimates are saved in an array (n x length(x0) x length(h))
S1 <- prodlim_curepk(x, t, d, xinu, data, x0 = x0, h = c(0.5, 1, 1.25), local = FALSE)

## Plot predicted survival curve for covariate value x0 = 0.5 and bandwidth
## h = 0.5
x0 <- 0.5
plot(S1$t, S1$surv[, 2, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
## The true survival curve is included as reference
lines(S1$t, 1 - exp(2*x0)/(1 + exp(2*x0)) + exp(2*x0)/(1 + exp(2*x0))*
            (1 - pweibull(S1$t, shape = 0.5 * (x0 + 4))), lwd = 2)

## Plot predicted survival curve for covariate value x0 = 0.5 and all
## bandwidths
plot(S1$t, S1$surv[, 2, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
lines(S1$t, S1$surv[, 2, 2], type = "s", lwd = 2)
lines(S1$t, S1$surv[, 2, 3],  type = "s", lwd = 3)
# The true survival curve is included as reference
lines(S1$t, 1 - exp(2*x0)/(1 + exp(2*x0)) + exp(2*x0)/(1 + exp(2*x0))*
            (1 - pweibull(S1$t, shape = 0.5 * (x0 + 4))), lwd = 2)

## ... (b) with local bandwidths h = (3, 1)
## The survival function S(t|x) is estimated for each value of x0 with the
## corresponding assigned bandwidth (local == TRUE).
## Note that the length of the vector x0 and the bandwidth h must be the same.
## The estimates are saved in a matrix of dimension (n, length(x0))
x0 <- c(0, 0.5)
h  <- c(3, 1)
S3 <- prodlim_curepk(x, t, d, xinu, data, x0 = x0, h = h, local = TRUE)
## Plot predicted survival curve for covariate value x = 0 and its bandwidth
## (h = 3)
plot(S3$t, S3$surv[, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
## The true survival curve is included as reference
x0 <- 0
lines(S3$t, 1 - exp(2*x0)/(1 + exp(2*x0)) + exp(2*x0)/(1 + exp(2*x0))*
            (1 - pweibull(S3$t, shape = 0.5 * (x0 + 4))), lwd = 2)

## ... (c) with the bootstrap bandwidth selector in x0 = 0 (the default
## when the bandwidth argument h is not provided).
```

```
## The bootstrap bandwidth is searched in a grid of 10 bandwidths (hl = 10)
## between 0.2 and 2 times the standardized interquartile range of the
## covariate values (hbound = c(0.1, 2)).
x0 <- 0
(S4 <- prodlim_curepk(x, t, d, xinu, data, x0 = x0))
## Equivalently
(S4 <- prodlim_curepk(x, t, d, xinu, data, x0 = x0,
                      bootpars = controlpars(hl = 10, hbound = c(0.1, 2))))
## Plot predicted survival curve for covariate value x = 0 and the bootstrap
## bandwidth
plot(S4$t, S4$surv[, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
## The true survival curve is included as reference
lines(S4$t, 1 - exp(2*x0)/(1 + exp(2*x0)) + exp(2*x0)/(1 + exp(2*x0))*
            (1 - pweibull(S4$t, shape = 0.5 * (x0 + 4))), lwd = 2)

## ... (d) with parallel computation (The bootstrap bandwidth is searched with
## b = 100 bootstrap resamples and 2 cores)
library(doParallel)
(S5 <- prodlim_curepk(x, t, d, xinu, data, x0 = x0,
                      bootpars = controlpars(b = 100, ncores = 2)))
## Plot predicted survival curve for covariate value x = 0 and the bootstrap
## bandwidth
plot(S5$t, S5$surv[, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
## The true survival curve is included as reference
lines(S5$t, 1 - exp(2*x0)/(1 + exp(2*x0)) + exp(2*x0)/(1 + exp(2*x0))*
            (1 - pweibull(S5$t, shape = 0.5 * (x0 + 4))), lwd = 2)
```

`sarcoma` *Sarcoma Dataset*

Description

Sarcoma is a rare type of cancer that represents 1% of all adult solid malignancies (Choy, 2014). If a tumor can be surgically removed to render the patient with sarcoma free of detectable disease, 5 years is the survival time at which sarcoma oncologists assume long-term remissions. `sarcoma` dataset contains the observed survival time of 232 patients until death from sarcoma, and covariates such as the age at diagnosis. Patients tumor free for more than 5 years were assumed to be long-term survivors (known to be cured, `xinu = 1`).

Usage

```
sarcoma
data(sarcoma, package = "npcurePK")
```

Format

A data frame with 232 rows and 4 variables:

`x` Age (years) of patients at diagnosis.

`t` Observed time until death from sarcoma.

`d` Censoring status (0 = censored, 1 = death from sarcoma).

`xinu` Cure status (0 = dead or unknown, 1 = tumor free for more than 5 years).

Source

Provided by the authors to serve as an example.

References

Choy, E. (2014). Sarcoma after 5 years of progression-free survival: Lessons from the French sarcoma group. Cancer, 120(19), 2942-2943.

Examples

```
## Survival estimates of patients aged 40 and 90 years old
## computed with bootstrap bandwidths
## (seed is used for bootstrap resampling)
(S1 <- prodlim_curepk(x, t, d, xinu, sarcoma, x0 = c(40, 90),
                      bootpars = controlpars(b = 100, ncores = 2, seed = 123)))
plot(S1$t, S1$surv[, 1], type = "s", xlab = "Time",
     ylab = "Survival probability", ylim = c(0, 1))
lines(S1$t, S1$surv[, 2], type = "s")
```

Index