Package: sicure (via r-universe)

November 5, 2024

Type Package

Title Single-Index Mixture Cure Models

Version 0.1.0

Description Single-index mixture cure models allow estimating the probability of cure and the latency depending on a vector (or functional) covariate, avoiding the curse of dimensionality. The vector of parameters that defines the model can be estimated by maximum likelihood. A nonparametric estimator for the conditional density of the susceptible population is provided. For more details, see Piñeiro-Lamas (2024) (<https://ruc.udc.es/dspace/handle/2183/37035>).

License GPL (>= 2)

Encoding UTF-8

Imports caTools, doBy, fda, npcure, StatMatch, stats

RoxygenNote 7.2.3

NeedsCompilation no

Author Beatriz Piñeiro-Lamas [aut, cre] (<https://orcid.org/0000-0003-0673-5377>), Ana López-Cheda [aut], Ricardo Cao [aut]

Maintainer Beatriz Piñeiro-Lamas <b.pineiro.lamas@udc.es>

Repository CRAN

Date/Publication 2024-11-04 12:30:03 UTC

Config/pak/sysreqs make libicu-dev

Contents

`cd.sm.uncured` *Smoothed cross-validation conditional density estimator of the susceptible population*

Description

This function implements a smoothed version of the nonparametric cross-validation estimator for the conditional density of the susceptible population proposed by Piñeiro-Lamas (2024) (see Equation (3.5)). Smoothing is done using the `k` nearest neighbors based on Mahalanobis distance. The Mahalanobis distance between two points (X_i, T_i) and (X_j, T_j) is defined as:

$$d_M((X_i,T_i),(X_j,T_j)) = \sqrt{\left(\begin{pmatrix} X_i \\ T_i \end{pmatrix} - \begin{pmatrix} X_j \\ T_j \end{pmatrix}\right)^t \Sigma^{-1} \left(\begin{pmatrix} X_i \\ T_i \end{pmatrix} - \begin{pmatrix} X_j \\ T_j \end{pmatrix}\right)},$$

where Σ is the covariance matrix of the joint distribution of (X, T).

Usage

```
cd.sm.uncured(x, time, delta, logh3, logh4, k = 10)
```

Arguments

`x`	A numeric vector giving the covariate values.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`logh3`	The logarithm of the bandwidth for smoothing the time variable.
`logh4`	The logarithm of the bandwidth for smoothing the covariate.
`k`	The number of nearest neighbors used to smooth.

Value

A vector containing the cross-validation conditional density of the susceptible population for each observation (X_i, T_i), smoothed by considering the `k` nearest neighbors with Mahalanobis distance.

References

Mahalanobis, P. C. (1936). On the generalised distance in statistics. Proceedings of the National Institute of Sciences of India, 2, 49-55.

Piñeiro-Lamas, B. (2024). High dimensional single-index mixture cure models [PhD thesis]. Universidade da Coruña. Available at https://ruc.udc.es/dspace/handle/2183/37035

See Also

cd.uncured, kNN.Mahalanobis

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)
suppressWarnings(cd.sm.uncured(x, t, d, log(0.5), log(0.5), k=10))
```

---

cd.uncured *Cross-validation conditional density of the susceptible population*

Description

This function implements a nonparametric cross-validation estimator for the conditional density of the susceptible population, as proposed by Piñeiro-Lamas (2024) (see Equation (3.5)). A leave-one-out cross-validation approach is considered to ensure that the sample used to construct the estimator and the point at which it is evaluated are independent.

Usage

```
cd.uncured(x, time, delta, logh3, logh4)
```

Arguments

x	A numeric vector giving the covariate values.
time	A numeric vector giving the observed times.
delta	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
logh3	The logarithm of the bandwidth for smoothing the time variable.
logh4	The logarithm of the bandwidth for smoothing the covariate.

Value

A vector containing the cross-validation conditional density of the susceptible population for each observation (X_i, T_i).

References

Piñeiro-Lamas, B. (2024). High dimensional single-index mixture cure models [PhD thesis]. Universidade da Coruña. Available at https://ruc.udc.es/dspace/handle/2183/37035

See Also

cd.sm.uncured

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)
suppressWarnings(cd.uncured(x, t, d, log(0.5), log(0.5)))
```

---

fun.opt *Objective function*

Description

This function computes the negative log-likelihood for a given set of parameters.

Usage

```
fun.opt(theta_h, x_cov, time, delta)
```

Arguments

theta_h	A numeric vector containing the initial iterant for the vector of parameters and the initial bandwidths.
x_cov	A matrix or data frame giving the covariate values. Each row represents an individual and each column corresponds to a variable.
time	A numeric vector giving the observed times.
delta	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.

Value

The value of the negative log-likelihood.

`k.epa` *Rescaled Epanechnikov kernel*

Description

This function computes the rescaled Epanechnikov kernel for a given value `s` and bandwidth `g`:

$$K\left(\frac{s}{g}\right) = \frac{3}{4g}\left(1 - \left(\frac{s}{g}\right)^2\right) I\left(\left|\frac{s}{g}\right| \leq 1\right),$$

where I is the indicator function (it takes the value 1 if the condition is true and 0 otherwise).

Usage

```
k.epa(g, s)
```

Arguments

`g`	A numeric value or vector which contains the bandwidth(s).
`s`	A numeric value or vector with the values of the variable in which the kernel will be evaluated.

Value

Rescaled Epanechnikov kernel for the given value `s` and bandwidth `g`.

Examples

```
k.epa(g=5,s=2)
k.epa(g=5,s=c(2,1.5))
k.epa(g=c(5,6),s=c(2,1.5))
```

kNN.Mahalanobis *K Nearest Neighbors with Mahalanobis Distance*

Description

This function computes the k nearest neighbors for a given set of data points, where each observation is a pair of the form (X, T), with X representing a covariate and T the observed time. The distance between each pair of points is computed using the Mahalanobis distance:

$$d_M((X_i, T_i), (X_j, T_j)) = \sqrt{\left(\begin{pmatrix} X_i \\ T_i \end{pmatrix} - \begin{pmatrix} X_j \\ T_j \end{pmatrix}\right)^t \Sigma^{-1} \left(\begin{pmatrix} X_i \\ T_i \end{pmatrix} - \begin{pmatrix} X_j \\ T_j \end{pmatrix}\right)},$$

where Σ is the variance-covariance matrix of the joint distribution of (X, T).

Usage

```
kNN.Mahalanobis(x, time, k)
```

Arguments

x	A numeric vector of length n giving the covariate values.
time	A numeric vector giving the observed times.
k	The number of nearest neighbors to search.

Value

A matrix with n rows and k columns. Each row represents each pair (X_i, T_i). The values in each row give the index of the k nearest neighbors considering Mahalanobis distance.

References

Mahalanobis, P. C. (1936). On the generalised distance in statistics. Proceedings of the National Institute of Sciences of India, 2, 49-55.

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
kNN.Mahalanobis(x=x, time=t, k=5)
```

`loglik.simcm` *Logarithm of the likelihood of a single-index mixture cure model*

Description

This function computes the logarithm of the likelihood of a single-index mixture cure model.

Usage

```
loglik.simcm(x, time, delta, logh1, logh2, logh3, logh4, r, k = 10)
```

Arguments

`x`	A numeric vector giving the covariate values.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`logh1`	The logarithm of the bandwidth used to smooth the covariate in the nonparametric estimation of the probability of cure.
`logh2`	The logarithm of the bandwidth used to smooth the covariate in the nonparametric estimation of the latency.
`logh3`	The logarithm of the bandwidth used to smooth the time variable in the nonparametric estimation of the conditional density of susceptible individuals.
`logh4`	The logarithm of the bandwidth used to smooth the covariate in the nonparametric estimation of the conditional density of susceptible individuals.
`r`	The radius of the moving window used to smooth the nonparametric estimation of the probability of cure.
`k`	The number of nearest neighbors used to smooth the nonparametric estimation of the conditional density of susceptible individuals.

Value

A list with two components:

- The value of the negative log-likelihood.
- The n terms that contribute to the negative log-likelihood.

`probcure.sm` *Smoothed version of the nonparametric estimator of the conditional probability of cure*

Description

This function computes a smoothed version of the nonparametric estimator of the probability of cure proposed by Xu and Peng (2014) and deeply studied by López-Cheda et al. (2017). The smoothing is performed using the `runmean` function, which computes a moving average of the estimated probabilities in a window determined by a radius `r`. The non-smoothed version is implemented in the `probcure` function of the `npcure` package (López-Cheda et al., 2021).

Usage

```
probcure.sm(x, time, delta, logh1, r)
```

Arguments

`x`	A numeric vector giving the covariate values.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`logh1`	The logarithm of the bandwidth for smoothing the covariate.
`r`	Radius of moving window.

Value

A list with two components:

- A vector containing the cross-validation estimations of the probability of cure.
- The previous vector smoothed with `runmean` with a moving window of $k = 2r + 1$.

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. doi:10.1016/j.csda.2016.08.002.

López-Cheda, A., Jácome, M. A., López-de-Ullibarri, I. (2021). The R Journal, 13(1), 21-41. doi:10.32614/RJ2021027.

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

See Also

`probcure`

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)

# Smoothed nonparametric estimates of cure probability with bandwidth=2
q1 <- probcure.sm(x, t, d, logh1 = log(2), r=2)[[2]]
plot(sort(x), q1[order(x)], type = "l", xlab = "Covariate", ylab = "Cure probability",
     ylim = c(0, 1))
```

`sicure.f` *Estimation of the vector of parameters in a single-index mixture cure model with a functional covariate*

Description

This function provides the estimation of the vector of parameters in a single-index mixture cure model with a functional covariate (see Piñeiro-Lamas, 2024, Section 4.1, pages 83-84). A Functional Principal Components Analysis (FPCA) representation that explains at least the (`propvar`$*100$)% of the variability of the data is considered (for more details, see Ramsay and Silverman, 2005).

Usage

```
sicure.f(x_cov, time, delta, propvar = 0.9, randomsearch = FALSE)
```

Arguments

`x_cov`	A matrix or data frame n x m giving the functional covariate values. Each row represents an individual (a curve); m is the number of observed points in each curve.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`propvar`	Minimum proportion of explained variability with the FPCA representation.
`randomsearch`	A logical value, `FALSE` by default, specifying whether a random search of the initial iterant is considered.

Details

The infinite-dimensional nature of the functional data is reduced via FPCA. This basis representation is then truncated, reducing the dimension to K, where each functional observation is summarized into a vector of scores, $(\xi_1, \xi_2, \ldots, \xi_K)$. After this reduction, the model can be treated similarly to the vector covariate case. For more details on the estimation process and the specific arguments, see `sicure.v` function, which focuses on single-index mixture cure models with a vector of covariates.

Value

A list with the following components:

`par`	A numeric vector of the estimated parameters. The last four correspond to the logarithms of the bandwidths.
`value`	The value of the objective function (negative log-likelihood) at the estimated parameters.
`si`	The estimated single-index variable.

References

Piñeiro-Lamas, B. (2024). High dimensional single-index mixture cure models [PhD thesis]. Universidade da Coruña. Available at https://ruc.udc.es/dspace/handle/2183/37035

Ramsay, J. O., and Silverman, B. W. (2005). Functional Data Analysis, 2nd ed., Springer, New York.

See Also

`sicure.v`, `sicure.vf`

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
delta  <- ifelse(u < p, ifelse(y < c, 1, 0), 0) # Uncensoring indicator
# Number of individuals (rows)
n <- 50
# Numbers of observations per individual (columns)
m <- 100
# Observation times (between 0 and 1)
x <- seq(0, 1, length.out = m)
# Auxiliar function to simulate the other functions by adding some noise
# Shift controls the horizontal displacement of the functions
sim_func <- function(x, shift, sd_noise) {
```

```
  # positive-negative-negative waves
  sin(2*pi*(x + shift))+sin(4*pi*(x + shift))-sin(6*pi*(x + shift))+rnorm(m, 0, sd_noise)
}
# Simulated functions
data_matrix <- matrix(NA, nrow=n, ncol=m)
for (i in 1:n) {
  shift <- runif(1, -0.05, 0.05)
  data_matrix[i, ] <- sim_func(x, shift, sd_noise = 0.03)
}
matplot(x, t(data_matrix), type = "l", lty = 1, ylab='f(x)')

suppressWarnings(sicure.f(data_matrix, t, delta, 0.9))
```

`sicure.v` *Estimation of the vector of parameters in a single-index mixture cure model with a vector of covariates*

Description

This function provides the estimation of the vector of parameters ($\boldsymbol{\theta_0}$) in a single-index mixture cure model with a vector of covariates (see Piñeiro-Lamas, 2024, Section 3.1, pages 37-38).

Usage

```
sicure.v(x_cov, time, delta, randomsearch = FALSE)
```

Arguments

`x_cov`	A matrix or data frame giving the covariate values. Each row represents an individual and each column corresponds to a variable.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`randomsearch`	A logical value, `FALSE` by default, specifying whether a random search of the initial iterant is considered.

Details

The vector of parameters, $\boldsymbol{\theta_0}$, is estimated by maximum likelihood, with the likelihood function also depending on four bandwidths. Since the parameters cannot be obtained directly from the likelihood, the estimation is performed using numerical optimization with the Nelder-Mead method. To begin the optimization, an initial iterant is required. To select a starting point for $\boldsymbol{\theta_0}$, a logistic regression model is fitted using the uncensoring indicator `delta` as the response variable and the covariates as predictors. Regarding the initial bandwidths, for h_3 the rule of thumb bandwidth selector used to estimate the density of the time variable is considered. For h_1, h_2 and h_4, the rule

of thumb bandwidth used to estimate the density of $\boldsymbol{\theta_{0,ini}}'\boldsymbol{X}$, where $\boldsymbol{\theta_{0,ini}}$ is the initial estimate of $\boldsymbol{\theta_0}$ and $\boldsymbol{X}$ is the vector of covariates. If `randomsearch = TRUE`, nine additional starting points are generated by adding a Gaussian noise (mean 0, standard deviation 0.25) to the logistic regression-based iterant. Each of these ten starting points is used to estimate the vector of parameters of the single-index mixture cure model, and the one that gives place to the lowest negative log-likelihood value is selected.

Value

A list with the following components:

`par`	A numeric vector of the estimated parameters. The last four correspond to the logarithms of the bandwidths.
`value`	The value of the objective function (negative log-likelihood) at the estimated parameters.
`si`	The estimated single-index variable.

References

Piñeiro-Lamas, B. (2024). High dimensional single-index mixture cure models [PhD thesis]. Universidade da Coruña. Available at https://ruc.udc.es/dspace/handle/2183/37035

See Also

`sicure.f`, `sicure.vf`

Examples

```
# Some artificial data
set.seed(123)
n <- 50
mix1a<-rnorm(n,mean=0,sd=1); mix1b<-rnorm(n,mean=0.25,sd=sqrt(2)); alf1<-rbinom(n,1,0.2)
mix2a<-rnorm(n,mean=0,sd=1); mix2b<-rnorm(n,mean=0.25,sd=sqrt(2)); alf2<-rbinom(n,1,0.2)
mix1<-alf1*mix1a+(1-alf1)*mix1b; mix2<-alf2*mix2a+(1-alf2)*mix2b
x_cov<-array(c(mix1,mix2),dim=c(n,2)) # Matrix of covariate values
theta<-c(1,1.2)
Z<-colSums(theta*t(x_cov))
y<-Z+rnorm(n,sd=sqrt(abs(Z))) # True lifetimes
# Probability of being susceptible
p_fun <- function(x){ 0.55 * exp(1.5*x+1.5)/(1+exp(1.5*x+1.5)) + 0.001 }
for (i in 1:n){
   w <- runif(1)
   if (w > p_fun(Z[i])) y[i] <- Inf
}
c<-rexp(n,rate=0.98) # Censoring values
t<-pmin(y,c) # Observed times
d = 1 * (y<=c) # Uncensoring indicator

suppressWarnings(sicure.v(x_cov, t, d))
```

`sicure.vf` *Estimation of the vector of parameters in a single-index mixture cure model with a vector and a functional covariate*

Description

This function provides the estimation of the vector of parameters in a single-index mixture cure model with a vector and a functional covariate (see Piñeiro-Lamas, 2024, Section 5.1, page 99). A Functional Principal Components Analysis (FPCA) representation that explains at least the (`propvar`$*100$)% of the variability of the functional data is considered (for more details, see Ramsay and Silverman, 2005).

Usage

```
sicure.vf(x_cov_v, x_cov_f, time, delta, propvar = 0.9, randomsearch = FALSE)
```

Arguments

`x_cov_v`	A matrix or data frame giving the vector covariate values. Each row represents an individual and each column corresponds to a variable.
`x_cov_f`	A matrix or data frame n x m giving the functional covariate values. Each row represents an individual (a curve); m is the number of observed points in each curve.
`time`	A numeric vector giving the observed times.
`delta`	A numeric vector giving the values of the uncensoring indicator, where 1 indicates that the event of interest has been observed and 0 indicates that the observation is censored.
`propvar`	Minimum proportion of explained variability with the FPCA representation.
`randomsearch`	A logical value, `FALSE` by default, specifying whether a random search of the initial iterant is considered.

Details

The infinite-dimensional nature of the functional data is reduced via FPCA. This basis representation is then truncated, reducing the dimension to K, where each functional observation is summarized into a vector of scores, $(\xi_1, \xi_2, \ldots, \xi_K)$. Once this reduction is performed, if the vector covariate has dimension d, a combined joint vector variable can be constructed by considering both the vector covariate and the functional scores, resulting in a total dimension of $d + K$. This joint variable can then be analyzed within the framework of a single-index mixture cure model with a vector of covariates. For more details on the estimation process and the specific arguments, see `sicure.v` function, which focuses on single-index mixture cure models with a vector of covariates.

Value

A list with the following components:

`par`	A numeric vector of the estimated parameters. The last four correspond to the logarithms of the bandwidths.
`value`	The value of the objective function (negative log-likelihood) at the estimated parameters.
`si`	The estimated single-index variable.

References

Piñeiro-Lamas, B. (2024). High dimensional single-index mixture cure models [PhD thesis]. Universidade da Coruña. Available at https://ruc.udc.es/dspace/handle/2183/37035

Ramsay, J. O., and Silverman, B. W. (2005). Functional Data Analysis, 2nd ed., Springer, New York.

See Also

`sicure.v`, `sicure.f`

Examples

```
# Some artificial data
set.seed(123)
n <- 50
mix1a<-rnorm(n,mean=0,sd=1); mix1b<-rnorm(n,mean=0.25,sd=sqrt(2)); alf1<-rbinom(n,1,0.2)
mix2a<-rnorm(n,mean=0,sd=1); mix2b<-rnorm(n,mean=0.25,sd=sqrt(2)); alf2<-rbinom(n,1,0.2)
mix1<-alf1*mix1a+(1-alf1)*mix1b; mix2<-alf2*mix2a+(1-alf2)*mix2b
x_cov_v<-array(c(mix1,mix2),dim=c(n,2)) # Matrix of covariate values
theta<-c(1,1.2)
Z<-colSums(theta*t(x_cov_v))
y<-Z+rnorm(n,sd=sqrt(abs(Z))) # True lifetimes
# Probability of being susceptible
p_fun <- function(x){ 0.55 * exp(1.5*x+1.5)/(1+exp(1.5*x+1.5)) + 0.001 }
for (i in 1:n){
   w <- runif(1)
   if (w > p_fun(Z[i])) y[i] <- Inf
}
c<-rexp(n,rate=0.98) # Censoring values
t<-pmin(y,c) # Observed times
d = 1 * (y<=c) # Uncensoring indicator
# Functional covariate:
# Number of individuals (rows)
n <- 50
# Numbers of observations per individual (columns)
m <- 100
# Observation times (between 0 and 1)
x <- seq(0, 1, length.out = m)
# Auxiliar function to simulate the other functions by adding some noise
# Shift controls the horizontal displacement of the functions
```

```
sim_func <- function(x, shift, sd_noise) {
  # positive-negative-negative waves
  sin(2*pi*(x + shift))+sin(4*pi*(x + shift))-sin(6*pi*(x + shift))+rnorm(m, 0, sd_noise)
}
# Simulated functions
data_matrix <- matrix(NA, nrow=n, ncol=m)
for (i in 1:n) {
  shift <- runif(1, -0.05, 0.05)
  data_matrix[i, ] <- sim_func(x, shift, sd_noise = 0.03)
}
matplot(x, t(data_matrix), type = "l", lty = 1, ylab='f(x)')

suppressWarnings(sicure.vf(x_cov_v, data_matrix, t, d, 0.9))
```

Index