Package ‘curstatCI’

October 12, 2022

Type Package

Title Confidence Intervals for the Current Status Model

Version 0.1.1

Description Computes the maximum likelihood estimator, the smoothed maximum likelihood estimator and pointwise bootstrap confidence intervals for the distribution function under current status data. Groeneboom and Hendrickx (2017) <doi:10.1214/17-EJS1345>.

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports Rcpp

Depends R (>= 2.10)

RoxygenNote 6.0.1.9000

URL https://github.com/kimhendrickx/curstatCI

BugReports https://github.com/kimhendrickx/curstatCI/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Piet Groeneboom [aut], Kim Hendrickx [cre]

Maintainer Kim Hendrickx <kim.hendrickx@uhasselt.be>

Repository CRAN

Date/Publication 2017-10-12 08:05:28 UTC
R topics documented:

ComputeBW .......................... 2
ComputeConfIntervals .................. 3
ComputeMLE .......................... 5
ComputeSMLE .......................... 7
hepatitisA .......................... 8
rubella .......................... 9

Description

The function ComputeBW computes the bandwidth that minimizes the pointwise Mean Squared Error using the subsampling principle in combination with undersmoothing.

Usage

ComputeBW(data, x)

Arguments

data       Dataframe with three variables:

  t  Observation points t sorted in ascending order. All observations need to be positive. The total number of unique observation points equals length(t).
  freq1  Frequency of observation t satisfying $x \leq t$. The total number of observations with censoring indicator $\delta = 1$ equals $\sum(freq1)$.
  freq2  Frequency of observation t. The sample size equals $\sum(freq2)$. If no tied observations are present in the data $\text{length}(t) = \sum(freq2)$.

x        numeric vector containing the points where the confidence intervals are computed.

Value

bw  data-driven bandwidth vector of size $\text{length}(x)$ containing the bandwidth value for each point in x.

References


See Also

vignette("curstatCI")
Examples

library(Rcpp)
library(curstatCI)

# sample size
n <- 1000

# truncated exponential distribution on (0,2)
set.seed(100)
t <- rep(NA, n)
delta <- rep(NA, n)
for(i in (1:n) ){
x<-runif(1)
y<-log(1-(1-exp(-2))*x)
t[i]<-2*runif(1);
if(y<=t[i]){ delta[i]<-1}
else{delta[i]<-0})

A<-cbind(t[order(t)], delta[order(t)], rep(1,n))

# x vector
grid<-seq(0.1,1.9 ,by = 0.1)

# data-driven bandwidth vector
bw <- ComputeBW(data =A, x = grid)
plot(grid, bw)

ComputeConfIntervals

Pointwise Confidence Intervals under Current Status data

Description

The function ComputeConfIntervals computes pointwise confidence intervals for the distribution function under current status data. The confidence intervals are based on the Smoothed Maximum likelihood Estimator and constructed using the nonparametric bootstrap.

Usage

ComputeConfIntervals(data, x, alpha, bw)

Arguments

data

data Dataframe with three variables:

- **t** Observation points \( t \) sorted in ascending order. All observations need to be positive. The total number of unique observation points equals \( \text{length}(t) \).

- **freq1** Frequency of observation \( t \) satisfying \( x \leq t \). The total number of observations with censoring indicator \( \delta = 1 \) equals \( \text{sum}(\text{freq1}) \).
freq2 Frequency of observation t. The sample size equals \( \text{sum}(freq2) \). If no tied observations are present in the data \( \text{length}(t) \) equals \( \text{sum}(freq2) \).

\( x \) numeric vector containing the points where the confidence intervals are computed. This vector needs to be contained within the observation interval: \( t[1] < \min(x) \leq \max(x) < t[n] \).

\( \alpha \) confidence level of pointwise confidence intervals.

\( bw \) numeric vector of size \( \text{length}(x) \). This vector contains the pointwise bandwidth values for each point in the vector \( x \).

Details

In the current status model, the variable of interest \( X \) with distribution function \( F \) is not observed directly. A censoring variable \( T \) is observed instead together with the indicator \( \Delta = (X \leq T) \). ComputeConfIntervals computes the pointwise 1-\( \alpha \) bootstrap confidence intervals around the SMLE of \( F \) based on a sample of size \( n \leftarrow \text{sum}(data.freq2) \).

The bandwidth parameter vector that minimizes the pointwise Mean Squared Error using the subsampling principle in combination with undersmoothing is returned by the function ComputeBW.

The default method for constructing the confidence intervals in [Groeneboom & Hendrickx (2017)] is based on estimating the asymptotic variance of the SMLE. When the bandwidth is small for some point in \( x \), the variance estimate of the SMLE at this point might not exist. If this happens the Non-Studentized confidence interval is returned for this particular point in \( x \).

Value

List with 5 variables:

- **MLE** Maximum Likelihood Estimator. This is a matrix of dimension \((m+1)x2\) where \( m \) is the number of jump points of the MLE. The first column consists of the point zero and the jump locations of the MLE. The second column contains the value zero and the values of the MLE at the jump points.

- **SMLE** Smoothed Maximum Likelihood Estimator. This is a vector of size \( \text{length}(x) \) containing the values of the SMLE for each point in the vector \( x \).

- **CI** pointwise confidence interval. This is a matrix of dimension \( \text{length}(x)x2 \). The first resp. second column contains the lower resp. upper values of the confidence intervals for each point in \( x \).

- **Studentized** points in \( x \) for which Studentized nonparametric bootstrap confidence intervals are computed.

- **NonStudentized** points in \( x \) for which classical nonparametric bootstrap confidence intervals are computed.

References


See Also

vignette("curstatCI")
Examples

```r
library(Rcpp)
library(curstatCI)

# sample size
n <- 1000

# Uniform data U(0,2)
set.seed(2)
y <- runif(n,0,2)
t <- runif(n,0,2)
delta <- as.numeric(y <= t)

A<-cbind(t[order(t)], delta[order(t)], rep(1,n))

# x vector
grid<-seq(0.1,1.9 ,by = 0.1)

# data-driven bandwidth vector
bw <- ComputeBW(data =A, x = grid)

# pointwise confidence intervals at grid points:
out<-ComputeConfIntervals(data = A,x =grid,alpha = 0.05, bw = bw)

left <- out$CI[,1]
right <- out$CI[,2]

plot(grid, out$SMLE,type ="l", ylim=c(0,1), ylab="",xlab="",las=1)
points(grid, left, col = 4)
points(grid, right, col = 4)
segments(grid,left, grid, right)
```

---

**ComputeMLE**

<table>
<thead>
<tr>
<th>Maximum Likelihood Estimator</th>
</tr>
</thead>
</table>

**Description**

The function ComputeMLE computes the Maximum Likelihood Estimator of the distribution function under current status data.

**Usage**

`ComputeMLE(data)`

**Arguments**

data: Dataframe with three variables:
ComputeMLE

**t** Observation points \( t \) sorted in ascending order. All observations need to be positive. The total number of unique observation points equals \( \text{length}(t) \).

**freq1** Frequency of observation \( t \) satisfying \( x \leq t \). The total number of observations with censoring indicator \( \delta = 1 \) equals \( \text{sum}(\text{freq1}) \).

**freq2** Frequency of observation \( t \). The sample size equals \( \text{sum}(\text{freq2}) \). If no tied observations are present in the data \( \text{length}(t) \) equals \( \text{sum}(\text{freq2}) \).

**Details**

In the current status model, the variable of interest \( X \) with distribution function \( F \) is not observed directly. A censoring variable \( T \) is observed instead together with the indicator \( \Delta = (X \leq T) \). ComputeMLE computes the MLE of \( F \) based on a sample of size \( n \leftarrow \text{sum}(\text{data}[,\text{freq2}]) \).

**Value**

Dataframe with two variables:

- \( x \) jump locations of the MLE
- \( \text{mle} \) MLE evaluated at the jump locations

**References**


**See Also**

ComputeConfIntervals

**Examples**

```r
library(Rcpp)
library(curstatCI)

# sample size
n <- 1000

# Uniform data U(0,2)
set.seed(2)
y <- runif(n,0,2)
t <- runif(n,0,2)
delta <- as.numeric(y <= t)
A<-cbind(t[order(t)], delta[order(t)], rep(1,n))
mle <-ComputeMLE(A)
plot(mle$x, mle$mle,type =‘s’, ylim=c(0,1), main= "",ylab="",xlab="",las=1)
```
Description

The function ComputeSMLE computes the Smoothed Maximum Likelihood Estimator of the distribution function under current status data.

Usage

ComputeSMLE(data, x, bw)

Arguments

data  Dataframe with three variables:
t  Observation points t sorted in ascending order. All observations need to be positive. The total number of unique observation points equals length(t).
freq1  Frequency of observation t satisfying x ≤ t. The total number of observations with censoring indicator δ = 1 equals sum(freq1).
freq2  Frequency of observation t. The sample size equals sum(freq2). If no tied observations are present in the data length(t) equals sum(freq2).
x  numeric vector containing the points where the confidence intervals are computed.
bw  numeric vector of size length(x). This vector contains the pointwise bandwidth values for each point in the vector x.

Details

In the current status model, the variable of interest X with distribution function F is not observed directly. A censoring variable T is observed instead together with the indicator Δ = (X ≤ T). ComputeSMLE computes the SMLE of F based on a sample of size n <= sum(data$freq2). The bandwidth parameter vector that minimizes the pointwise Mean Squared Error using the subsampling principle in combination with undersmoothing is returned by the function ComputeBW.

Value

SMLE(x) Smoothed Maximum Likelihood Estimator. This is a vector of size length(x) containing the values of the SMLE for each point in the vector x.

References


See Also

ComputeConfIntervals
### Examples

```r
library(Rcpp)
library(curstatCI)

# sample size
n <- 1000

# Uniform data U(0,2)
set.seed(2)
y <- runif(n,0,2)
t <- runif(n,0,2)
delta <- as.numeric(y <= t)

A<-cbind(t[order(t)], delta[order(t)], rep(1,n))
ggrid <-seq(0,2 ,by = 0.01)

# bandwidth vector
h<-rep(2*n^-0.2,length(grid))

smle <-ComputeSMLE(A,grid,h)
plot(grid, smle,type =',ylab=',xlab=','xlas=1)
```

---

**hepatitisA**

**Hepatitis A data**

---

### Description

A dataset on the prevalence of hepatitis A in individuals from Bulgaria with age ranging from 1 to 86 years. The data consists of a cross-sectional survey conducted in 1964.

### Usage

hepatitisA

### Format

A data frame with 83 rows and three variables:

- **t** Age of the individual
- **freq1** Number of individuals of age t that are seropositive for Hepatitis A
- **freq2** Total number of individuals of age t

### References

Rubella data

Description

A dataset on the prevalence of rubella in 230 Austrian males older than three months for whom the exact date of birth was known. Each individual was tested at the Institute of Virology, Vienna during the period 1–25 March 1988 for immunization against Rubella.

Usage

rubella

Format

A data frame with 225 rows and three variables:

- t  Age of the individual at the time of testing for immunization
- freq1  Number of individuals of age t that are immune for Rubella
- freq2  Total number of individuals of age t

References

Index

* datasets
  hepatitisA, 8
  rubella, 9

ComputeBW, 2, 4, 7
ComputeConfIntervals, 3, 6, 7
ComputeMLE, 5
ComputeSMLE, 7

hepatitisA, 8
rubella, 9