Package ‘lpint’

February 20, 2015

Type  Package
Title  Local polynomial estimators of intensity function or its
derivatives
Version 2.0
Date  2014-04-02
Author Feng Chen <feng.chen@unsw.edu.au>
Maintainer Feng Chen <feng.chen@unsw.edu.au>
Description Estimates the intensity function, or its derivative of a given order, of a multiplica-
tive counting process using the local polynomial method
License GPL (>= 2.0)
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2014-04-02 13:24:31

R topics documented:

lpint-package ........................................................................................................... 1
lpint ............................................................................................................................. 3
lplikint ......................................................................................................................... 5

Index 8

lpint-package  Local polynomial estimators of the intensity function of a counting
process and its derivatives

Description

Estimates the intensity function or its derivative of a give a given order using the local polynomial
method with automatic bandwidth selection using a crude rule of thumb plug-in approach.

1
Details
**lpint**

*Martingale estimating equation local polynomial estimator of counting process intensity function and its derivatives*

**Description**

This local polynomial estimator is based on a biased martingale estimating equation.

**Usage**

```r
lpint(jmptimes, jmpsizes = rep(1, length(jmptimes)),
       Y = rep(1, length(jmptimes)), bw = NULL,
       adjust = 1, Tau = max(1, jmptimes), p = nu + 1,
       nu = 0, K = function(x) 3/4 * (1 - x^2) * (x <= 1 & x >= -1),
       n = 101, bw.only=FALSE)
```

**Author(s)**

Feng Chen <feng.chen@unsw.edu.au> Maintainer: Feng Chen <feng.chen@unsw.edu.au>

**References**


Arguments

jmptimes  a numeric vector giving the jump times of the counting process
jmpsizes  a numeric vector giving the jump sizes at each jump time. Need to be of the same length as jmptimes
Y        a numeric vector giving the value of the exposure process (or size of the risk set) at each jump times. Need to be of the same length as jmptimes
bw       a numeric constant specifying the bandwidth used in the estimator. If left unspecified the automatic bandwidth selector will be used to calculate one.
adjust   a positive constant giving the adjust factor to be multiplied to the default bandwidth parameter or the supplied bandwidth
tau      a numeric constant >0 giving the censoring time (when observation of the counting process is terminated)
p        the degree of the local polynomial used in constructing the estimator. Default to 1 plus the degree of the derivative to be estimated
nu       the degree of the derivative of the intensity function to be estimated. Default to 0 for estimation of the intensity itself.
k        the kernel function
n        the number of evenly spaced time points to evaluate the estimator at
bw.only  TRUE or FALSE according as if the rule of thumb bandwidth is the only required output or not

Value

either a list containing
x        the vector of times at which the estimator is evaluated
y        the vector giving the values of the estimator at times given in x
se       the vector giving the standard errors of the estimates given in y
bw       the bandwidth actually used in defining the estimator equal the automatically calculated or supplied multiplied by adjust

or a numeric constant equal to the rule of thumb bandwidth estimate

Author(s)

Feng Chen <feng.chen@unsw.edu.au.>

References


See Also

lplikint
Examples

##simulate a Poisson process on [0,1] with given intensity
```r
t <- function(x) 100 * (1 + 0.5 * cos(2 * pi * x))
censor <- 1
set.seed(2)
N <- rpois(1, 150 * censor)
jtms <- runif(N, 0, censor)
jtms <- jtms[as.logical(mapply(rbinom, n = 1, size = 1, prob = int(jtms) / 150))]
```

##estimate the intensity
```r
intest <- lpint(jtms, Tau = censor)
```

##plot and compare
```r
plot(intest, xlab = "time", ylab = "intensity", type = "l", lty = 1)
curve(int, add = TRUE, lty = 2)
```

## Example estimating the hazard function from right censored data:
## First simulate the (not directly observable) life times and censoring times:
```r
tl <- rweibull(500, 2.5, 3); ct <- rlnorm(500, 1, 0.5)
```

## Now the censored times and censorship indicators delta (the observables):
```r
ot <- pmin(lt, ct); dlt <- as.numeric(lt <= ct)
```

## Estimate the hazard rate based on the censored observations:
```r
jtms <- sort(ot[dlt == 1]);
```

## plot the estimated hazard function:
```r
matplot(haz.est$x, pmax(haz.est$y + outer(haz.est$se, c(-1, 0, 1) * qnorm(0.975)), 0),
        type = "l", lty = c(2, 1, 2),
        xlab = "t", ylab = "h(t)",
        col = 1);
```

## add the truth:
```r
haz <- function(x) dweibull(x, 2.5, 3) / pweibull(x, 2.5, 3, lower.tail = FALSE)
curve(haz, add = TRUE, col = 2)
```

### Description
This local polynomial estimator is based on the (localized) partial likelihood

### Usage
```r
lplikint(jmptimes, jmpsizes = rep(1, length(jmptimes)),
         Y = rep(1, length(jmptimes)),
         K = function(x) 3/4 * (1 - x^2) * (x <= 1 & x >= -1),
```
\texttt{bw, adjust = 1, nu = 0, p = 1, Tau = 1, n = 101,}
\texttt{tseq = seq(from = 0, to = Tau, length = n), tol = 1e-05,}
\texttt{maxit = 100, us = 10, gd = 5)}

**Arguments**

- \texttt{jmptimes}: a numeric vector giving the jump times of the counting process
- \texttt{jmpsizes}: a numeric vector giving the jump sizes at each jump time. Need to be of the same length as \texttt{jmptimes}
- \texttt{y}: a numeric vector giving the value of the exposure process (or size of the risk set) at each jump times. Need to be of the same length as \texttt{jmptimes}
- \texttt{K}: the kernel function
- \texttt{bw}: a numeric constant specifying the bandwidth used in the estimator. If left unspecified the automatic bandwidth selector will be used to calculate one.
- \texttt{adjust}: a positive constant giving the adjust factor to be multiplied to the default bandwidth parameter or the supplied bandwidth
- \texttt{nu}: the degree of the derivative of the intensity function to be estimated. Default to 0 for estimation of the intensity itself.
- \texttt{p}: the degree of the local polynomial used in constructing the estimator. Default to 1 plus the degree of the derivative to be estimated
- \texttt{tau}: a numeric constant >0 giving the censoring time (when observation of the counting process is terminated)
- \texttt{n}: the number of evenly spaced time points to evaluate the estimator at. Not used when \texttt{tseq} is provided.
- \texttt{tseq}: the time sequence at which to evaluate the estimator
- \texttt{tol}: the parameter error tolerance used to stop the iterations in optimizing the local likelihood
- \texttt{maxit}: maximum number of iterations allowed in the optimization used in a single estimation point
- \texttt{us}: a numeric constants used together with \texttt{gd} to grid search for a decent start value in solving the local score equation. The starting value is 0 except on its first dimension, which was chosen so that the starting value is the minimizer of the $L^1$ norm of the score function among the values: average intensity $X us^{(-gd:gd)}$
- \texttt{gd}: a numeric constant used together with \texttt{us} to search for a decent start value in solving the local score equation

**Details**

The estimator is based on solving the local score equation using the Newton-Raphson method and extract the appropriate dimension.
Value

- a list containing
  - `x` the vector of times at which the estimator is evaluated
  - `y` the vector giving the values of the estimator at times given in `x`
  - `se` the vector giving the standard errors of the estimates given in `y`
  - `bw` the bandwidth actually used in defining the estimator equal the automatically calculated or supplied multiplied by `adjust`
  - `fun` the intensity (or derivative) estimator as a function of the estimation point, which can be called to evaluate the estimator at points not included in `tseq`

Author(s)

Feng Chen <feng.chen@unsw.edu.au.>

References


See Also

- `lpint`

Examples

```r
##simulate a Poisson process on [0,1] with given intensity
int <- function(x)100*(1+0.5*cos(2*pi*x))
censor <- 1
set.seed(2)
N <- rpois(1,150*censor);
jtms <- runif(N,0,censor);
jtms <- jtms[as.logical(mapply(rbinom,N,1,size=1,prob=int(jtms)/150))];

##estimate the intensity
intest <- lplikint(jtms,bw=0.15,Tau=censor)
#plot and compare
plot(intest,xlab="time",ylab="intensity",type="l",lty=1)
curve(int,add=TRUE,lty=2)
```
Index

*Topic package
  lpint-package, 1

*Topic smooth
  lpint, 3
  lplikint, 5

*Topic survival
  lpint, 3
  lplikint, 5

lpint, 3, 7
lpint-package, 1
lplikint, 4, 5