

Package ‘convexHaz’

February 14, 2012

Version 0.2

Date 2008-11-07

Title Nonparametric MLE/LSE of convex hazard

Author Hanna Jankowski, Ivy Wang, Hugh McCague, Jon A. Wellner

Maintainer Hanna Jankowski <hkj@mathstat.yorku.ca>

Depends R (>= 2.4)

Description This package contains functions to compute the nonparametric maximum likelihood estimator (MLE) and the nonparametric least squares estimator (LSE) of a convex hazard function, assuming that the data is IID.

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-01-29 19:36:09

R topics documented:

convexHaz-package	2
convexLSE	3
convexMLE	6
hazard	10
srLSE	11
srMLE	13

Index	17
--------------	-----------

convexHaz-package *Find the nonparametric MLE/LSE of convex hazard*

Description

This package contains functions to compute the nonparametric maximum likelihood estimator (MLE) and the nonparametric least squares estimator (LSE) of a convex hazard function, assuming that the data is IID.

Details

Package: convexHaz
 Version: 0.2
 Date: 2008-11-07
 Depends: R (>= 2.4)
 License: GPL (version 2 or later)
 Built: R 2.7.1; ; 2008-11-07 13:05:46; windows

Index:

convexLSE	Compute the nonparametric LSE of a convex hazard
convexMLE	Compute the nonparametric MLE of a convex hazard
hazard	Create a hazard function (and its integral) given its support and mixing measure.
srLSE	Compute the LSE of a convex hazard with fixed antimode
srMLE	Compute the MLE of a convex hazard with fixed antimode

Author(s)

Hanna Jankowski, Ivy Wang, Hugh McCague, Jon A. Wellner
 Maintainer: Hanna Jankowski <hkj@mathstat.yorku.ca>

References

Groeneboom, Jongbloed and Wellner (2008). The support reduction algorithm for computing nonparametric function estimates in mixture models. *Scan. J. Statist.* **35**, 385–399.

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

convexLSE

*Compute the nonparametric LSE of a convex hazard***Description**

This function computes the nonparametric LSE of a convex hazard function on $[0, TT]$.

Usage

```
convexLSE(x, TT, M = 100, GRIDLESS = 0, tol = 1e-05, tol.SR = 1e-08, type = 1,
max.loop = 100, max.loop.SR = 250, range = c(0, TT), solve.tol = .Machine$double.eps)
```

Arguments

x	vector containing the data.
TT	value specifying TT in the interval $[0, TT]$ over which the LSE is found.
M	size of the grid used in the implementation of the (support reduction) algorithm. The default value is $M=100$.
GRIDLESS	boolean variable. If $GRIDLESS=1$, then a 'gridless' implementation of the algorithm is used. The default is $GRIDLESS=0$.
tol	tolerance value of the bisection algorithm. The default value is $tol=1e-05$.
tol.SR	tolerance value of the support reduction algorithm. The default value is $tol.SR=1e-08$.
type	integer between 1 and 3 indicating the exit criterion function used by the bisection algorithm. The default is $type=1$.
max.loop	maximum number of iterations in the bisection algorithm. The default is $max.loop=100$. To be used for troubleshooting.
max.loop.SR	maximum number of iteration in the support reduction algorithm. The default is $max.loop.SR=250$. To be used for troubleshooting.
range	vector of length two describing the range of antimodes over which the (bisection) algorithm searches. The default is $range=[0, TT]$. The range should always be contained within $[0, TT]$. To be used in troubleshooting.
solve.tol	tolerance value used in <code>solve()</code> invoked by the algorithm. The default is $solve.tol=.Machine$double$. which is also the default of the <code>solve()</code> function. To be used in troubleshooting.

Details

Let HH_n be the empirical cumulative hazard function for the data x . Then the LSE is the convex hazard function which is closest to the hazard function whose integral is given by HH_n , in the least squares sense. See Jankowski and Wellner (2007) for further details.

The function consists of an outer, bisection algorithm loop and an inner, support reduction algorithm loop. The goal is to minimize the least squares criterion function over the space of positive convex hazards. This is done in two steps: the support reduction algorithm minimizes the least squares

criterion function (ϕ) over the space of positive convex hazard with minimum (antimode) at a using the function `srLSE`; the bisection algorithm searches over the possible values of a in $[0, \pi]$. The bisection algorithm ends once the exit criterion function is smaller than `tol`. The exit criterion function may be specified via the option `type`:

```
type=1 sum((cur.phi-min(old.phi))^2)
type=2 sqrt(sum((cur.phi - old.phi)^2))
type=3 sort(cur.phi)[2]+sort(cur.phi)[3]-2*sort(cur.phi)[1]
```

where `cur.phi` is the vector of the least squares criterion function ϕ along different values of antimode a , and `old.phi` is the same but in the previous iteration.

The support reduction algorithm was first described in Groeneboom, Jongbloed and Wellner (2007). It is an iterative optimization procedure which exits when a sufficient tolerance level `tol.SR` is reached. The algorithm uses a gridded implementation (see Jankowski and Wellner (2008) for the details) described by the parameter `M`. The idea is that any convex function may be written as a mixture of elbow functions, and the grid describes the possible support of the mixing distribution. (The documentation for the function `h` provides a little more information; see also Jankowski and Wellner (2007, 2008).) The larger `M` is, the more accurate, but slower, the algorithm. Setting `GRIDLESS=1` invokes an implementation which tries to overcome the grid, and increases the accuracy again. Decreasing `tol` and/or `tol.SR` may also increase the accuracy of the algorithm, but not as much as manipulating the grid size.

Value

A list which includes

<code>lse</code>	list containing the support <code>supp</code> and mixing measure <code>mix</code> for the computed LSE. The function <code>h</code> can translate these into a hazard function.
<code>ls</code>	value of the least squares criterion function at the LSE. See Jankowski and Wellner (2007) for its exact form.
<code>antimode</code>	value of the antimode a at which the LSE of the hazard has its minimum.
<code>conv</code>	boolean indicating if the (bisection) algorithm has converged.
<code>iter</code>	number of iterations of the bisection algorithm.
<code>"convexLSE.out"</code>	

The algorithm also creates a file called `convexLSE.out` in R's current working directory. The file contains three variables: (1) `phi` the value of the smallest least squares criterion function at the antimode found by the support reduction algorithm, (2) `antimode` the value of the antimode, and (3) `conv` a boolean indicating if the support reduction algorithm converged. If `convexLSE.out` already exists, then the algorithm will overwrite it.

Diagnostics

The file `"convexLSE.out"` contains information which may be used to determine how well the algorithm has worked. Plotting `phi` against `antimode` should result in a u-shaped (first decreasing, then increasing) function, with minimum lying at the antimode of the LSE. This is only true though for the values of `phi` (and `antimode`) where the `srLSE` algorithm converged (ie. `conv=TRUE`). Plotting the output in `"convexLSE.out"` can be used to check this (see 'Examples'), and to check where convergence of `srLSE` failed.

Troubleshooting

If the bisection algorithm fails to converge, then `max.loop` may be increased, or changing the range may be used temporarily to diagnose the issue.

Alternatively, the support reduction algorithm may fail to converge. The file "convexLSE.out" shows which of the calls of the support reduction algorithm by the bisection loop converged. If too many of these failed, then changing the values of `max.loop`, `max.loop.SR`, or `solve.tol` may fix this.

At each iteration, the support reduction algorithm does a finite dimensional quadratic optimization using the function `solve()`. Occasionally (quite rarely for the LSE), the matrix passed to `solve()` is computationally singular. Several built-in catches exist in the code to handle this, but if the problem occurs too frequently, the algorithm will abort. In fact, this is by far the most common cause of nonconvergence. The larger the value of `M`, the more likely the problem is, but this also depends on the data set. Setting `solve.tol` to a smaller value may fix the problem.

Author(s)

Hanna Jankowski: <hkj@mathstat.yorku.ca>, Ivy Wang, Hugh McCague, Jon A. Wellner

References

Groeneboom, Jongbloed and Wellner (2008). The support reduction algorithm for computing non-parametric function estimates in mixture models. *Scan. J. Statist.* **35**, 385–399.

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

See Also

[convexMLE](#) [srLSE](#) [srMLE](#) [h](#)

Examples

```
# Generate sample data:
set.seed(1111, kind="default")
x <- rweibull(50, 3)

# Set the value of TT:
TT <- 1

# Find the lse
lse <- convexLSE(x, TT=TT)

# And the lse has minimum at
lse$antimode

# Check the bisection algorithm
diag <- read.table("convexLSE.out", header=TRUE)
diag
```

```

plot(diag$antimode, diag$phi, pch=16, col="blue", cex=1, main="diagnostics", ylab="phi", xlab="antimode")

# plot the true hazard function and the lse
h.true <- function(x) 3*x^2
tt <- c(0,sort(lse$lse$supp$tau), sort(lse$lse$supp$eta), TT)
yy <- sapply(tt, h, supp=lse$lse$supp, hpar=lse$lse$mix)

plot(h.true, xlim=c(0,TT), lwd=2, col="blue", ylab="hazard", xlab="time", main="LSE of convex hazard")
lines(tt,yy, lwd=2, col="red")
legend("topleft", c("true","estimated"),col=c("blue", "red"), lwd=2, inset = .05)

```

convexMLE

Compute the nonparametric MLE of a convex hazard

Description

This function computes the nonparametric MLE of a convex hazard function.

Usage

```

convexMLE(x, M = 100, GRIDLESS = 0, ini.LSE = 0, tol = 1e-05, tol.SR = 1e-08, type = 1,
max.loop = 100, max.inner.loop.SR = 250, max.outer.loop.SR = 50, range = c(0, max(x)),
solve.tol = .Machine$double.eps)

```

Arguments

x	vector containing the data.
M	size of the grid used in the implementation of the (support reduction) algorithm. The default value is M=100.
GRIDLESS	boolean variable. If GRIDLESS=1, then a 'gridless' implementation of the algorithm is used. The default value is GRIDLESS=0.
ini.LSE	boolean variable. If ini.LSE=1 then the initial value for each call of the support reduction algorithm for the MLE is taken from the least squares estimator. The default value is ini.LSE=0.
tol	tolerance value of the bisection algorithm. The default value is tol=1e-05.
tol.SR	tolerance value of the support reduction algorithm. The default value is tol.SR=1e-08.
type	integer between 1 and 3 indicating the exit criterion function used by the bisection algorithm. The default is type=1.
max.loop	maximum number of iterations in the bisection algorithm. The default value is max.loop=100. To be used for troubleshooting.

<code>max.inner.loop.SR</code>	maximum number of iterations in the inner loop of the support reduction algorithm. The default value is <code>max.inner.loop.SR=250</code> . To be used for troubleshooting.
<code>max.outer.loop.SR</code>	maximum number of iterations in the outer loop of the support reduction algorithm. The default value is <code>max.outer.loop.SR=50</code> . To be used for troubleshooting.
<code>range</code>	vector of length two describing the range of antimodes over which the (bisection) algorithm searches. The default is <code>range=c(0,max(x))</code> . The range should always be contained in $[0, \max(x)]$. To be used in troubleshooting.
<code>solve.tol</code>	tolerance value used in <code>solve()</code> invoked by the algorithm. The default is <code>solve.tol=.Machine\$double</code> , which is also the default of the <code>solve()</code> function. To be used in troubleshooting.

Details

This function gives an iterative procedure to find the nonparametric MLE of a convex hazard function under IID sampling. The likelihood in this case is actually a modified likelihood, which excludes the term $h(\max(x))$. The full likelihood can be made arbitrarily large by increasing $h(\max(x))$, and therefore this term is omitted. The estimated MLE returned by the algorithm is thus the function on $[0, \max(x))$, with the assumption that the MLE is equal to infinity beyond $\max(x)$.

The function consists of an outer, bisection algorithm loop and an inner, support reduction algorithm loop which calls upon `srMLE`. The goal is to maximize the likelihood over the space of positive convex hazards. This is done in two steps: the support reduction algorithm maximizes the log likelihood (or minimizes its negative, `negllh`) over the space of positive convex hazard with minimum (antimode) at a ; the bisection algorithm searches over the possible values of a in $[0, \max(x)]$.

The bisection algorithm ends once the exit criterion function is smaller than `tol`. The exit criterion function may be specified via the option `type`:

```
type=1 sum((cur.phi-min(old.phi))^2)
```

```
type=2 sqrt(sum((cur.phi - old.phi)^2))
```

```
type=3 sort(cur.phi)[2]+sort(cur.phi)[3]-2*sort(cur.phi)[1]
```

where `cur.phi` is the vector of the least squares criterion function ϕ along different values of antimode a , and `old.phi` is the same but in the previous iteration.

The support reduction algorithm was first described in Groeneboom, Jongbloed and Wellner (2007). It is an iterative optimization procedure which exits when a sufficient tolerance level `tol.SR` is reached. The algorithm uses a gridded implementation (see Jankowski and Wellner (2008) for the details) described by the parameter `M`. The idea is that any convex function may be written as a mixture of elbow functions, and the grid describes the possible support of the mixing distribution. (The documentation for the function `h` provides a little more information; see also Jankowski and Wellner (2007, 2008).) The larger `M` is the more accurate, and slower, the algorithm. Setting `GRIDLESS=1` will also increase the accuracy, but this often results in non-convergence of the support reduction algorithm (see 'Troubleshooting').

Value

<code>mle</code>	list containing the support <code>supp</code> and mixing measure <code>mix</code> (further lists) for the computed MLE. The function <code>h</code> can translate these into a hazard function.
<code>llh</code>	value of the (modified) log likelihood at the MLE.
<code>antimode</code>	value of the antimode <code>a</code> at which the MLE of the hazard has its minimum.
<code>conv</code>	boolean indicating if the (bisection) algorithm has converged.
<code>iter</code>	number of iterations of the bisection algorithm.
<code>"convexMLE.out"</code>	

The algorithm also creates a file called `convexMLE.out` in R's current working directory. The file contains three variables: (1) `negllh` the value of the smallest least squares criterion function at the antimode found by the support reduction algorithm, (2) `antimode` the value of the antimode, and (3) `conv` a boolean indicating if the support reduction algorithm converged. If `convexMLE.out` already exists, then the algorithm will overwrite it.

Diagnostics

The file `"convexMLE.out"` contains information which may be used to determine how well the algorithm has worked. Plotting `negllh` against `antimode` should result in a U-shaped (first decreasing, then increasing) function, with minimum lying at the antimode of the MLE. This is only true though for the values of `negllh` (and `antimode`) where the `srMLE` algorithm converged (ie. `conv=TRUE`). Plotting the output in `"convexMLE.out"` can be used to check this (see 'Examples'), and to check where convergence of `srMLE` failed.

Troubleshooting

If the bisection algorithm fails to converge, then `max.loop` may be increased. Also, changing the range may be used temporarily to diagnose the issue.

Alternatively, the support reduction algorithm may fail to converge. The file `"convexMLE.out"` shows which of the calls of the support reduction algorithm by the bisection loop converged. If too many of these failed, then changing the values of `max.inner.loop.SR`, `max.outer.loop.SR`, `solve.tol` or `ini.LSE` may fix this. Increasing `max.inner.loop.SR` or `max.outer.loop.SR` gives the algorithms more time to converge, though it is rare that this is the cause of the problem. Setting `ini.LSE` changes the starting point of the support reduction algorithm, and may even increase its speed.

At each iteration, the support reduction algorithm does a finite dimensional quadratic optimization using the function `solve()`. Occasionally, the matrix passed to `solve()` is computationally singular. Several built-in catches exist in the code to handle this, but if the problem occurs too frequently, the algorithm will abort. In fact, this is by far the most common cause of nonconvergence. The larger the value of `M`, the more likely the problem is, but this also depends on the data set. Also, choosing `GRIDLESS=1` highly increases the probability that this type of problem will occur. Setting `solve.tol` to a smaller value may fix the problem.

Author(s)

Hanna Jankowski: <hkj@mathstat.yorku.ca>, Ivy Wang, Hugh McCague, Jon A. Wellner

References

Groeneboom, Jongbloed and Wellner (2008). The support reduction algorithm for computing nonparametric function estimates in mixture models. *Scan. J. Statist.* **35**, 385–399.

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

See Also

[convexLSE hazard srLSE srMLE](#)

Examples

```
# Generate sample data:
set.seed(1111, kind="default")
x <- rbeta(100, 0.5, 0.5)

# Find the mle (this takes a few minutes)
mle <- convexMLE(x, M=1000)

# And the mle has minimum at
mle$antimode

# Check the bisection algorithm
diag <- read.table("convexMLE.out", header=TRUE)
diag

plot(diag$antimode, diag$negllh, pch=16, col="blue", main="diagnostics", ylab="phi", xlab="antimode")

# plot the true hazard function and the lse
h.true <- function(x) dbeta(x, 0.5, 0.5)/(1-pbeta(x, 0.5, 0.5))
tt <- c(0,sort(mle$mle$supp$tau), sort(mle$mle$supp$eta), max(x))
yy <- sapply(tt, h, supp=mle$mle$supp, hpar=mle$mle$mix)

plot(h.true, xlim=c(0, 0.95), lwd=2, col="blue", ylab="hazard", xlab="time", main="MLE of convex hazard")
lines(tt,yy, lwd=2, col="red")
legend("topleft", c("true","estimated"),col=c("blue", "red"), lwd=2, inset = .05)
```

hazard	<i>Create a hazard function (and its integral) given its support and mixing measure.</i>
--------	--

Description

Create a hazard function (and its integral) given its support and mixing measure.

Usage

`h(tt, supp, hpar)`

`H(tt, supp, hpar)`

Arguments

<code>tt</code>	real positive number.
<code>supp</code>	list containing the values in the support: e.g. <code>list(constant = 0, tau = c(0.5), eta = c(1,2))</code> .
<code>hpar</code>	list containing the values in the mixing measure: e.g. <code>list(alpha = numeric(), nu = c(3), mu = c(2,0.5))</code> .

Details

The functions `convexMLE` and `convexLSE`, as well as `srMLE` and `srLSE` return the MLE/LSE in terms of the support and mixture of the hazard function. The function `h` translates these into a hazard function, and the function `H` gives the cumulative hazard function (integral of `h`).

Any convex hazard function can be written as a mixture of three types of basis functions: a constant function $e(t)=1$, decreasing elbow functions $e(t)=\max(\tau-t, 0)$, and increasing elbow functions $e(t)=\max(t-\eta, 0)$. If the minimum of the hazard is located at a point a , then the possible values of τ lie in the set $[0, a]$ and the possible values of η must be greater than a . We call the point a the antimode. `supp` describes the list of basis functions to use: if `constant` is 0 then there is no constant function, and `tau` and `eta` list the values of τ and η used, respectively. `hpar` describes the mixing weights associated to these functions, with `alpha` giving the weight of the constant, and `nu` and `mu` giving the weights of τ and η , respectively. Note that the weights need not total to one, and that for a discrete mixture the hazard rate will be a piecewise constant function.

For example, if `supp=list(constant=1, tau=c(0.5), eta=numeric())` and `hpar=list(constant=3, nu=c(2), mu=numeric())`, then the hazard function is $h(t)=3*1+2*\max(0.5-t, 0)$.

Value

value of the hazard or cumulative hazard function at (time) t .

Author(s)

Hanna Jankowski: <hkj@mathstat.yorku.ca>

References

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

See Also

[convexLSE](#) [convexMLE](#) [srLSE](#) [srMLE](#)

Examples

```
# Generate sample data:
set.seed(1111, kind="default")
x <- rweibull(50, 3)

# Find the LSE with antimode a=0 over the range [0,1]:
TT <- 1
lse <- srLSE(x, a=0, TT=TT)

# create simpler function to evaluate hazard and cumulative hazard
h.lse <- function(t){ return(h(t, lse$lse$supp, lse$lse$mix))}
H.lse <- function(t){ return(H(t, lse$lse$supp, lse$lse$mix))}

# hazard function at t=0
h.lse(0)

# cumulative hazard function at t=1
H.lse(1)

# plot the hazard function h.lse
tt <- c(0,sort(lse$lse$supp$tau), sort(lse$lse$supp$eta), TT) # where h.lse changes slope
yy <- sapply(tt, h.lse) # values of h.lse at tt
plot(tt, yy, xlim=c(0,TT), type="l", lwd=2, col="red", ylab="hazard", xlab="time", main="LSE (a=0)")
```

srLSE

Compute the LSE of a convex hazard with fixed antimode

Description

This function computes the LSE of a convex hazard function on $[0, TT]$ with antimode a .

Usage

```
srLSE(x, a, TT, M = 100, GRIDLESS = 0, tol = 1e-08, max.loop = 250, print = 0,
solve.tol = .Machine$double.eps)
```

Arguments

x	vector containing the data.
a	value of the antimode (the location of the minimum of the hazard rate).
TT	value specifying TT in the interval [0, TT] over which the LSE is found.
M	size of the grid used in the implementation of the algorithm. The default value is M=100.
GRIDLESS	boolean variable. If GRIDLESS=1, then a 'gridless' implementation of the algorithm is used. The default value is GRIDLESS=0.
tol	tolerance value of the algorithm. The default is tol=1e-08.
max.loop	maximum number of iterations in the algorithm. The default is max.loop=250. To be used for troubleshooting.
print	boolean variable. If print=1, then the results of each iteration are printed. The default value is print=0. To be used for troubleshooting.
solve.tol	tolerance value used in solve() invoked by the algorithm. To be used for troubleshooting. The default value is the same as for the solve() function.

Details

Let HH_n be the empirical cumulative hazard function for the data x . Then the LSE is the convex hazard function, with minimum at a , which is closest to the hazard function whose integral is given by HH_n , in the least squares sense. See Jankowski and Wellner (2007) for further details.

The function implements the support reduction algorithm Groeneboom, Jongbloed and Wellner (2007).

Increasing M , decreasing tol , and setting $GRIDLESS=1$ will each increase the accuracy of the algorithm. However, these will also increase the computing time, especially increasing M .

Value

A list which includes

lse	list containing the support <code>supp</code> and mixing measure <code>mix</code> (further lists) for the computed LSE. The function <code>h</code> can translate these into a hazard function.
ls	value of the least squares criterion function at the LSE.
iter	number of iterations of the algorithm.
conv	boolean indicating if the algorithm has converged.

Troubleshooting

If the algorithm fails to converge, then there are several options available. Setting `print=1` can reveal the reason behind the failure. If the maximal number of iterations was reached, then increasing `max.loop` may solve the problem. If a linear solve problem occurred, then one can try to decrease the tolerance of the `solve()` function, `solve.tol`.

Author(s)

Hanna Jankowski: <hkj@mathstat.yorku.ca>, Ivy Wang, Hugh McCague, Jon A. Wellner

References

Groeneboom, Jongbloed and Wellner (2008). The support reduction algorithm for computing non-parametric function estimates in mixture models. *Scan. J. Statist.* **35**, 385–399.

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

See Also

[convexMLE](#) [convexLSE](#) [hazard](#) [srMLE](#)

Examples

```
# Generate sample data:
set.seed(1111, kind="default")
x <- rweibull(50, 3)

# Find the LSE with antimode a=0 over the range [0,1]:
TT <- 1
lse <- srLSE(x, a=0, TT=TT)

# plot the true hazard function and the lse
h.true <- function(x) 3*x^2
tt <- c(0,sort(lse$lse$supp$tau), sort(lse$lse$supp$eta), TT)
yy <- sapply(tt, h, supp=lse$lse$supp, hpar=lse$lse$mix)

plot(h.true, xlim=c(0,TT), lwd=2, col="blue", ylab="hazard", xlab="time", main="LSE (a=0)")
lines(tt,yy, lwd=2, col="red")
legend("topleft", c("true","estimated"),col=c("blue", "red"), lwd=2, inset = .05)
```

 srMLE

Compute the MLE of a convex hazard with fixed antimode

Description

This function computes the MLE of a convex hazard function with antimode a .

Usage

```
srMLE(x, a, M = 100, GRIDLESS = 0, ini.LSE = 0, tol = 1e-08, max.inner.loop = 250,
max.outer.loop = 50, print = 0, solve.tol = .Machine$double.eps)
```

Arguments

<code>x</code>	vector containing the data.
<code>a</code>	value of the antimode (location of the minimum of the hazard rate).
<code>M</code>	size of the grid used in the implementation of the algorithm. The default value is <code>M=100</code> .
<code>GRIDLESS</code>	boolean variable. If <code>GRIDLESS=1</code> , then a 'gridless' implementation of the algorithm is used. The default value is <code>GRIDLESS=0</code> .
<code>ini.LSE</code>	boolean variable. If <code>ini.LSE=1</code> then the initial value for the support reduction algorithm for the MLE is taken from the least squares estimator. The default value is <code>ini.LSE=0</code> .
<code>tol</code>	tolerance value of the algorithm. The default value is <code>tol=1e-08</code> .
<code>max.inner.loop</code>	maximum number of iterations in the inner loop of the support reduction algorithm. The default value is <code>max.inner.loop=250</code> . To be used for troubleshooting.
<code>max.outer.loop</code>	maximum number of iterations in the outer loop of the support reduction algorithm. The default value is <code>max.outer.loop=50</code> . To be used for troubleshooting.
<code>print</code>	boolean variable. If <code>print=1</code> , then the results of each iteration are printed. The default value is <code>print=0</code> . To be used for troubleshooting.
<code>solve.tol</code>	tolerance value used in <code>solve()</code> invoked by the algorithm. The default value is the same as that of the <code>solve()</code> function. To be used for troubleshooting.

Details

This function implements the support reduction algorithm to find the nonparametric MLE of a convex hazard function with antimode at `a` under IID sampling. The likelihood in this case is actually a modified likelihood, which excludes the term $h(\max(x))$. The full likelihood can be made arbitrarily large by increasing $h(\max(x))$, and therefore this term is omitted. The estimated MLE returned by the algorithm is thus the function on $[0, \max(x))$, and we assume that the MLE is equal to infinity beyond $\max(x)$.

Increasing `M` and/or decreasing `tol` will increase the accuracy of the algorithm, but increase the computation time. Setting `GRIDLESS=1` will also increase the accuracy, but this may cause the algorithm not to converge (see 'Troubleshooting').

Value

A list which includes

<code>mle</code>	list containing the support <code>supp</code> and mixture <code>mix</code> (further lists) for the computed MLE. The function <code>h</code> can translate these into a hazard function.
<code>llh</code>	value of the (modified) log likelihood at the MLE.
<code>conv</code>	boolean indicating if the algorithm has converged.
<code>iter</code>	number of iterations (of the outer loop) of the algorithm.

Troubleshooting

If the algorithm fails to converge, then there are several options available. Setting `print=1` in the options may reveal to cause of the problem.

The support reduction algorithm consists of inner and outer loops, if the algorithm reaches the maximum in either one of these, then increasing `max.outer.loop` or `max.inner.loop` may fix the problem, though this is unlikely.

At each iteration, the support reduction algorithm performs a finite dimensional quadratic optimization using the function `solve()`. Occasionally, the matrix passed to `solve()` is computationally singular. Several built-in catches exist in the code to handle this, but if the problem occurs too frequently, the algorithm will abort. In fact, this is by far the most common cause of nonconvergence. The larger the value of `M`, the more likely the problem is, but this also depends on the data set. The problem is also likely to occur if `GRIDLESS=1`. Setting `solve.tol` to a smaller value may fix the problem.

In addition, changing the starting point of the algorithm may help, which is done by setting `sr.LSE=1`. This may also increase the speed of the algorithm.

Author(s)

Hanna Jankowski: <hkj@mathstat.yorku.ca>, Ivy Wang, Hugh McCague, Jon A. Wellner

References

Groeneboom, Jongbloed and Wellner (2008). The support reduction algorithm for computing nonparametric function estimates in mixture models. *Scan. J. Statist.* **35**, 385–399.

Jankowski and Wellner (2007). Nonparametric estimation of a convex bathtub-shaped hazard function. *Technical Report 521*, Department of Statistics, University of Washington.

Jankowski and Wellner (2008). Computation of nonparametric convex hazard estimators via profile methods. *Technical Report 542*, Department of Statistics, University of Washington.

See Also

[convexLSE](#) [convexMLE](#) [hazard](#) [srLSE](#)

Examples

```
# Generate sample data:
set.seed(3333, kind="default")
x <- rweibull(50, 3)

# Find the LSE with antimode a=0 over the range [0,1]:
TT <- 1
mle <- srMLE(x, a=0)

# plot the true hazard function and the lse
h.true <- function(x) 3*x^2
tt <- c(0, sort(mle$mle$supp$tau), sort(mle$mle$supp$eta), max(x))
yy <- sapply(tt, h, supp=mle$mle$supp, hpar=mle$mle$mix)
```

```
plot(h.true, xlim=c(0,max(x)), lwd=2, col="blue", ylab="hazard", xlab="time", main="MLE (a=0)")  
lines(tt,yy, lwd=2, col="red")  
legend("topleft", c("true","estimated"),col=c("blue", "red"), lwd=2, inset = .05)
```

Index

*Topic **iteration**

- convexHaz-package, 2
- convexLSE, 3
- convexMLE, 6
- srLSE, 11
- srMLE, 13

*Topic **nonparametric**

- convexHaz-package, 2
- convexLSE, 3
- convexMLE, 6
- hazard, 10
- srLSE, 11
- srMLE, 13

*Topic **optimize**

- convexHaz-package, 2
- convexLSE, 3
- convexMLE, 6
- srLSE, 11
- srMLE, 13

*Topic **package**

- convexHaz-package, 2

*Topic **survival**

- convexHaz-package, 2
- convexLSE, 3
- convexMLE, 6
- hazard, 10
- srLSE, 11
- srMLE, 13

convexHaz (convexHaz-package), 2

convexHaz-package, 2

convexLSE, 3, 9–11, 13, 15

convexMLE, 5, 6, 10, 11, 13, 15

H (hazard), 10

h, 4, 5, 8, 12, 14

h (hazard), 10

hazard, 9, 10, 13, 15

srLSE, 4, 5, 9, 10, 11, 11, 15

srMLE, 5, 7–11, 13, 13