Package 'extremefit'

April 10, 2025

Type Package

Title Estimation of Extreme Conditional Quantiles and Probabilities

Version 1.0.3

Date 2025-04-06

Depends R (>= 2.10)

Suggests survival, R.rsp

Maintainer Kevin Jaunatre <kevin.jaunatre@hotmail.fr>

Description Extreme value theory, nonparametric kernel estimation, tail conditional probabilities, extreme conditional quantile, adaptive estimation, quantile regression, survival probabilities.

License GPL-2

RoxygenNote 7.3.2

VignetteBuilder R.rsp

NeedsCompilation no

Author Gilles Durrieu [aut],
Ion Grama [aut],
Kevin Jaunatre [aut, cre],
Quang-Khoai Pham [aut],
Jean-Marie Tricot [aut]

Repository CRAN

Date/Publication 2025-04-10 19:20:22 UTC

Contents

`bandwidth.CV` *Choice of the bandwidth by cross validation.*

Description

Choose a bandwidth by minimizing the cross validation function.

Usage

```
bandwidth.CV(
  X,
  t,
  Tgrid,
  hgrid,
  pcv = 0.99,
  kernel = TruncGauss.kernel,
  kpar = NULL,
```

```
  CritVal = 3.6,
  plot = FALSE
)
```

Arguments

X	a vector of the observed values.
t	a vector of time covariates which should have the same length as X.
Tgrid	a sequence of times used to perform the cross validation (can be any sequence in the interval [min(t) , max(t)]).
hgrid	a sequence of values from which the bandwidth is selected.
pcv	a probability value which determines the level of quantiles used to perform the cross validation, with default 0.99.
kernel	a kernel function used to compute the weights in the time domain, with default the truncated gaussian kernel.
kpar	a value for the kernel function parameter, with no default value.
CritVal	a critical value associated to the kernel function computed from the function CriticalValue, with default 3.6 corresponding to the truncated Gaussian kernel.
plot	If TRUE, the cross validation function is plotted.

Details

The sequence $hgrid$ must be geometric. (see bandwidth.grid to generate a geometric grid of bandwidths).

The value pcv should be scalar (vector values are not admitted).

Value

hgrid	the sequence of bandwidth given in input.
CV	the values of the cross validation function for hgrid.
h.cv	the bandwidth that minimizes the cross-validation function.

Author(s)

Durrieu, G., Grama, I., Jaunatre, K., Pham, Q. and Tricot, J.- M

References

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

See Also

bandwidth.grid , CriticalValue

Examples

```
#Generate the data
theta <- function(t){
   0.5+0.25*sin(2*pi*t)
 }
n <- 5000
t <- 1:n/n
Theta <- theta(t)
Data <- NULL
for(i in 1:n){
   Data[i] <- rparetomix(1, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75)
 }

#compute the cross validation bandwidth
Tgrid <- seq(0, 1, 0.02) #define a grid to perform the cross validation
hgrid <- bandwidth.grid(0.1, 0.3, 20) #define a grid of bandwidths
## Not run:  #For computation time purpose
  Hcv <- bandwidth.CV(Data, t, Tgrid, hgrid, pcv = 0.99, plot = TRUE)
  #The computing time can be long
  Hcv

## End(Not run)
```

`bandwidth.grid` *Bandwidth Grid*

Description

Create either a geometric or an uniform grid of bandwidths between two values

Usage

```
bandwidth.grid(hmin, hmax, length = 20, type = "geometric")
```

Arguments

`hmin`	the minimum value of the grid.
`hmax`	the maximum value of the grid.
`length`	the length of the grid.
`type`	the type of grid, either $geometric$ or $uniform$.

Details

The geometric grid is defined by:

$$grid(l) = hmin * exp((log(hmax) - log(hmin))/(length - 1))^l, l = 0, ..., (length - 1)$$

Value

Return a geometric or uniform grid of size $length$ between $hmin$ and $hmax$.

Examples

```
hmin <- 0.05
hmax <- 0.2
length <- 20
(h.geometric <- bandwidth.grid(hmin, hmax, length, type = "geometric"))
(h.uniform <- bandwidth.grid(hmin, hmax, length, type = "uniform"))
```

Biweight.kernel *Biweight kernel function*

Description

Biweight kernel function.

Usage

```
Biweight.kernel(x)
```

Arguments

x	a vector.

Details

Biweight kernel:

$$K(x) = 15/16(1 - x^2)^2(abs(x) <= 1)$$

We recommend a critical value of 7 for this kernel function.

Examples

```
plot(function(x) Biweight.kernel(x),-2, 2,
main = " Biweight kernel ")
```

bootCI *Pointwise confidence intervals by bootstrap*

Description

Pointwise quantiles and survival probabilities confidence intervals using bootstrap.

Usage

```
bootCI(
  X,
  weights = rep(1, length(X)),
  probs = 1:(length(X) - 1)/length(X),
  xgrid = sort(X),
  B = 100,
  alpha = 0.05,
  type = "quantile",
  CritVal = 10,
  initprop = 1/10,
  gridlen = 100,
  r1 = 1/4,
  r2 = 1/20,
  plot = F
)
```

Arguments

X	a numeric vector of data values.
weights	a numeric vector of weights.
probs	used if type = "quantile", a numeric vector of probabilities with values in $[0, 1]$.
xgrid	used if type = "survival", a numeric vector with values in the domain of X.
B	an integer giving the number of bootstrap iterations.
alpha	the type 1 error of the bootstrap (1-$alpha$)-confidence interval.
type	type is either "quantile" or "survival".
CritVal	a critical value associated to the kernel function given by CriticalValue. The default value is 10 corresponding to the rectangular kernel.
gridlen, initprop, r1, r2	parameters used in the function hill.adapt (see hill.adapt).
plot	If TRUE, the bootstrap confidence interval is plotted.

Details

Generate B samples of X with replacement to estimate the quantiles of orders $probs$ or the survival probability corresponding to $xgrid$. Determine the bootstrap pointwise (1-$alpha$)-confidence interval for the quantiles or the survival probabilities.

Value

LowBound	the lower bound of the bootstrap (1-$alpha$)-confidence interval.
UppBound	the upper bound of the bootstrap (1-$alpha$)-confidence interval of level.

See Also

hill.adapt,CriticalValue,predict.hill.adapt

Examples

```
X <- abs(rcauchy(400))
hh <- hill.adapt(X)
probs <- probgrid(0.1, 0.999999, length = 100)
B <- 200
## Not run:  #For computing time purpose
  bootCI(X, weights = rep(1, length(X)), probs = probs, B = B, plot = TRUE)
  xgrid <- sort(sample(X, 100))
 bootCI(X, weights = rep(1, length(X)), xgrid = xgrid, type = "survival", B = B, plot = TRUE)

## End(Not run)
```

---

bootCI.ts *Pointwise confidence intervals by bootstrap*

Description

Pointwise quantiles and survival probabilities confidence intervals using bootstrap.

Usage

```
bootCI.ts(
  X,
  t,
  Tgrid,
  h,
  kernel = TruncGauss.kernel,
  kpar = NULL,
  prob = 0.99,
  threshold = quantile(X, 0.99),
  B = 100,
  alpha = 0.05,
  type = "quantile",
  CritVal = 3.6,
  initprop = 1/10,
  gridlen = 100,
  r1 = 1/4,
```

```
  r2 = 1/20,
  plot = F
)
```

Arguments

X	a vector of the observed values.
t	a vector of time covariates which should have the same length as X.
Tgrid	a sequence of times used to perform the cross validation (can be any sequence in the interval [min(t) , max(t)]).
h	a bandwidth value (vector values are not admitted).
kernel	a kernel function used to compute the weights in the time domain, with default the truncated gaussian kernel.
kpar	a value for the kernel function parameter, with no default value.
prob	used if type = "quantile", a scalar value in $[0, 1]$ which determines the quantile order (vector values are not admitted).
threshold	used if type = "survival", a scalar value in the domain of X.
B	an integer giving the number of bootstrap iterations.
alpha	the type 1 error of the bootstrap (1-$alpha$)-confidence interval.
type	type is either "quantile" or "survival".
CritVal	a critical value associated to the kernel function given by CriticalValue. The default value is 3.6 corresponding to the truncated Gaussian kernel.
gridlen, initprop, r1, r2	parameters used in the function hill.adapt (see hill.adapt).
plot	If TRUE, the bootstrap confidence interval is plotted.

Details

For each point in $Tgrid$, generate B samples of X with replacement to estimate the quantile of order $prob$ or the survival probability beyond $threshold$. Determine the bootstrap pointwise (1-$alpha$)-confidence interval for the quantiles or the survival probabilities.

The kernel implemented in this packages are : Biweight kernel, Epanechnikov kernel, Rectangular kernel, Triangular kernel and the truncated Gaussian kernel.

Value

LowBound	the lower bound of the bootstrap (1-$alpha$)-confidence interval.
UppBound	the upper bound of the bootstrap (1-$alpha$)-confidence interval of level.

Warning

The executing time of the function can be time consuming if the B parameter or the sample size are high (B=100 and the sample size = 5000 for example) .

See Also

hill.ts,predict.hill.ts, Biweight.kernel, Epa.kernel, Rectangular.kernel, Triang.kernel, TruncGauss.kernel

Examples

```
theta <- function(t){
   0.5+0.25*sin(2*pi*t)
 }
n <- 5000
t <- 1:n/n
Theta <- theta(t)
set.seed(123)
Data <- NULL
for(i in 1:n){
   Data[i] <- rparetomix(1, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75)
 }
Tgrid <- seq(1, length(Data)-1, length = 20)/n
h <- 0.1
## Not run:  #For computing time purpose
  bootCI.ts(Data, t, Tgrid, h, kernel = TruncGauss.kernel, kpar = c(sigma = 1),
            CritVal = 3.6, threshold = 2, type = "survival", B = 100, plot = TRUE)
  true.p <- NULL
  for(i in 1:n){
     true.p[i] <- 1-pparetomix(2, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75)
   }
  lines(t, true.p, col = "red")
  bootCI.ts(Data, t, Tgrid, h, kernel = TruncGauss.kernel, kpar = c(sigma = 1),
 prob = 0.999, type = "quantile", B = 100, plot = TRUE)
  true.quantile <- NULL
  for(i in 1:n){
     true.quantile[i] <- qparetomix(0.999, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75)
   }
  lines(t, log(true.quantile), col = "red")

## End(Not run)
```

Burr Distribution *Burr distribution*

Description

Density, distribution function, quantile function and random generation for the Burr distribution with a and k two parameters.

Usage

```
rburr(n, a, k)

dburr(x, a, k)

pburr(q, a, k)

qburr(p, a, k)
```

Arguments

n	a number of observations. If length(n) > 1, the length is taken to be the number required.
a	a parameter of the burr distribution
k	a parameter of the burr distribution
x	a vector of quantiles.
q	a vector of quantiles.
p	a vector of probabilities.

Details

The cumulative Burr distribution is

$$F(x) = 1 - (1 + (x^a))^{-k}, x > 0, a > 0, k > 0$$

Value

dburr gives the density, pburr gives the distribution function, qburr gives the quantile function, and rburr generates random deviates.

The length of the result is determined by n for rburr, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
plot(function(x) dburr(x,3,1), 0, 5,ylab="density",
main = " burr density ")

plot(function(x) pburr(x,3,1), 0, 5,ylab="distribution function",
     main = " burr Cumulative ")

plot(function(x) qburr(x,3,1), 0, 1,ylab="quantile",
     main = " burr Quantile ")

#generate a sample of burr distribution of size n
n <- 100
x <- rburr(n, 1, 1)
```

`cox.adapt` *Compute the extreme quantile procedure for Cox model*

Description

Compute the extreme quantile procedure for Cox model

Usage

```
cox.adapt(
  X,
  cph,
  cens = rep(1, length(X)),
  data = rep(0, length(X)),
  initprop = 1/10,
  gridlen = 100,
  r1 = 1/4,
  r2 = 1/20,
  CritVal = 10
)
```

Arguments

`X`	a numeric vector of data values.
`cph`	an output object of the function coxph from the package survival.
`cens`	a binary vector corresponding to the censored values.
`data`	a data frame containing the covariates values.
`initprop`	the initial proportion at which we begin to test the model.
`gridlen`	the length of the grid for which the test is done.
`r1`	a proportion value of the data from the right that we skip in the test statistic.
`r2`	a proportion value of the data from the left that we skip in the test statistic.
`CritVal`	the critical value assiociated to procedure.

Details

Given a vector of data, a vector of censorship and a data frame of covariates, this function compute the adaptive procedure described in Grama and Jaunatre (2018).

We suppose that the data are in the domain of attraction of the Frechet-Pareto type and that the hazard are somewhat proportionals. Otherwise, the procedure will not work.

Value

coefficients	the coefficients of the coxph procedure.
Xsort	the sorted vector of the data.
sortcens	the sorted vector of the censorship.
sortebz	the sorted matrix of the covariates.
ch	the Hill estimator associated to the baseline function.
TestingGrid	the grid used for the statistic test.
TS, TS1, TS.max, TS1.max	respectively the test statistic, the likelihood ratio test, the maximum of the test statistic and the maximum likelihood ratio test.
window1, window2	indices from which the threshold was chosen.
Paretodata	logical: if TRUE the distribution of the data is a Pareto distribution.
Paretotail	logical: if TRUE a Pareto tail was detected.
madapt	the first indice of the TestingGrid for which the test statistic exceeds the critical value.
kadapt	the adaptive indice of the threshold.
kadapt.maxlik	the maximum likelihood corresponding to the adaptive threshold in the selected testing grid.
hadapt	the adaptive weighted parameter of the Pareto distribution after the threshold.
Xadapt	the adaptive threshold.

Author(s)

Ion Grama, Kevin Jaunatre

References

Grama, I. and Jaunatre, K. (2018). Estimation of Extreme Survival Probabilities with Cox Model. arXiv:1805.01638.

See Also

coxph

Examples

```
library(survival)
data(bladder)

X <- bladder2$stop-bladder2$start
Z <- as.matrix(bladder2[, c(2:4, 8)])
delta <- bladder2$event

ord <- order(X)
X <- X[ord]
```

```
Z <- Z[ord,]
delta <- delta[ord]

cph<-coxph(Surv(X, delta) ~ Z)

ca <- cox.adapt(X, cph, delta, Z)
```

`CriticalValue` *Computation of the critical value in the hill.adapt function*

Description

For a given kernel function, compute the critical value (CritVal) of the test statistic in the hill.adapt function by Monte-Carlo simulations.

Usage

```
CriticalValue(
  NMC,
  n,
  kernel = TruncGauss.kernel,
  kpar = NULL,
  prob = 0.95,
  gridlen = 100,
  initprop = 0.1,
  r1 = 0.25,
  r2 = 0.05,
  plot = FALSE
)
```

Arguments

`NMC`	the number of Monte-Carlo simulations.
`n`	the sample size.
`kernel`	a kernel function for which the critical value is computed. The available kernel functions are Epanechnikov, Triangular, Truncated Gaussian, Biweight and Rectangular. The truncated gaussian kernel is by default.
`kpar`	a value for the kernel function parameter, with no default value.
`prob`	a vector of type 1 errors.
`gridlen`, `initprop`, `r1`, `r2`	parameters used in the function hill.adapt (see `hill.adapt`).
`plot`	If `TRUE`, the empirical cummulative distribution function and the critical values are plotted.

Value

For the type 1 errors $prob$, this function returns the critical values.

References

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

See Also

hill.adapt

Examples

```
n <- 1000
NMC <- 500
prob <- c(0.99)
## Not run:  #For computing time purpose
  CriticalValue(NMC, n, TruncGauss.kernel, kpar = c(sigma = 1), prob, gridlen = 100 ,
                initprop = 1/10, r1 = 1/4, r2 = 1/20, plot = TRUE)

## End(Not run)
```

---

dataOyster *High-frequency noninvasive valvometry data*

Description

The data frame provides the opening amplitude of one oyster's shells (in mm) with respect to the time (in hours). The opening velocity of the oyster's shells is also given.

Usage

```
data("dataOyster")
```

Format

A list of 2 elements.

`$data` : a `data frame` with `54000` observations for `3` `variables` `time` Time of measurement (in hours).

`opening` opening amplitude between the two shells (in mm).

`velocity` a numeric vector (in mm/s). Negative values correspond to the opening velocity of the shells and positive values to the closing velocity of the shells.

`$Tgrid` : A `grid` of `time` to `perform` the `procedure`.

References

Durrieu, G., Grama, I., Pham, Q. & Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

Azais, R., Coudret R. & Durrieu G. (2014). A hidden renewal model for monitoring aquatic systems biosensors. Environmetrics, 25.3, 189-199.

Schmitt, F. G., De Rosa, M., Durrieu, G., Sow, M., Ciret, P., Tran, D., & Massabuau, J. C. (2011). Statistical study of bivalve high frequency microclosing behavior: Scaling properties and shot noise analysis. International Journal of Bifurcation and Chaos, 21(12), 3565-3576.

Sow, M., Durrieu, G., Briollais, L., Ciret, P., & Massabuau, J. C. (2011). Water quality assessment by means of HFNI valvometry and high-frequency data modeling. Environmental monitoring and assessment, 182(1-4), 155-170.

website : http://molluscan-eye.epoc.u-bordeaux1.fr/

Examples

```
data("dataOyster")
Velocity <- dataOyster$data[, 3]
time <- dataOyster$data[, 1]
plot(time, Velocity, type = "l", xlab = "time (hour)",
      ylab = "Velocity (mm/s)")

Tgrid <- seq(0, 24, 0.05)
#Grid with positive velocity
new.Tgrid <- dataOyster$Tgrid

X <- Velocity + (-min(Velocity)) #We shift the data to be positive

## Not run:  #For computing time purpose
#We find the h by minimizing the cross validation function

hgrid <- bandwidth.grid(0.05, 0.5, 50, type = "geometric")


#H <- bandwidth.CV(X, time, new.Tgrid, hgrid,
#                 TruncGauss.kernel, kpar = c(sigma = 1),
#                 pcv = 0.99, CritVal = 3.4, plot = TRUE)
#hcv <- H$h.cv

hcv <- 0.2981812
#we use our method with the h found previously
TS.Oyster <- hill.ts(X, t = time, new.Tgrid, h = hcv,
                   TruncGauss.kernel, kpar = c(sigma = 1),
                   CritVal = 3.4)

plot(time, Velocity, type = "l", ylim = c(-0.6, 1),
    main = "Extreme quantiles estimator",
    xlab = "Time (hour)", ylab = "Velocity (mm/s)")
pgrid <- c(0.999)
pred.quant.Oyster <- predict(TS.Oyster, newdata = pgrid, type = "quantile")
```

```
quant0.999 <- rep(0, length(Tgrid))
quant0.999[match(new.Tgrid, Tgrid)] <-
          as.numeric(pred.quant.Oyster$y)-
          (-min(Velocity))
lines(Tgrid, quant0.999, col = "magenta")



## End(Not run)
```

---

dataWind *Wind speed for Brest (France)*

Description

The data frame provides the wind speed of Brest from 1976 to 2005.

Usage

```
data("dataWind")
```

Format

The data is the wind speed in meters per second (m/s) every day from 1976 to 2005.

`Year` The year of the measure.

`Month` The month of the measure.

`Day` the day of the measure.

`Speed` The wind speed in meters per second

Examples

```
library(extremefit)
data("dataWind")
attach(dataWind)

pred <- NULL
for(m in 1:12){
  indices <- which(Month == m)
  X <- Speed[indices]*60*60/1000
  H <- hill.adapt(X)
  pred[m] <- predict(H, newdata = 100, type = "survival")$y
}
plot(pred, ylab = "Estimated survival probability", xlab = "Month")
```

Epa.kernel *Epanechnikov kernel function*

Description

Epanechnikov kernel function.

Usage

```
Epa.kernel(x)
```

Arguments

x	vector.

Details

Epanechnikov kernel:

$$K(x) = 3/4(1 - x^2)(abs(x) <= 1)$$

We recommend a critical value of 6.1 for this kernel function.

Examples

```
plot(function(x) Epa.kernel(x), -2, 2, ylab = "Epanechnikov kernel function")
```

Gaussian.kernel *Gaussian kernel function*

Description

Gaussian kernel function

Usage

```
Gaussian.kernel(x)
```

Arguments

x	a vector.

Details

Gaussian Kernel with the value of standard deviation equal to 1/3.

$$K(x) = (1/(1/3) * sqrt(2\pi)exp(-(3 * x)^2/2))(abs(x) <= 1)$$

We recommend a critical value of 8.3 for this kernel.

Examples

```
plot(function(x) Gaussian.kernel(x), -2, 2,
main = " Gaussian kernel")
```

`goftest` *Goodness of fit test statistics*

Description

goftest is a generic function whose application depends on the class of its argument.

Usage

```
goftest(object, ...)
```

Arguments

`object`	model object.
`...`	further arguments passed to or from other methods.

Value

The form of the value returned by goftest depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

References

Grama, I. and Spokoiny, V. (2008). Statistics of extremes by oracle estimation. Ann. of Statist., 36, 1619-1648.

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

See Also

`goftest.hill.adapt`, `goftest.hill.ts`

goftest.hill.adapt *Goodness of fit test statistics*

Description

Give the results of the goodness of fit tests for testing the null hypothesis that the tail is fitted by a Pareto distribution, starting from the adaptive threshold, against the Pareto change point distribution for all possible change points (for more details see pages 447 and 448 of Durrieu et al. (2015)).

Usage

```
## S3 method for class 'hill.adapt'
goftest(object, plot = FALSE, ...)
```

Arguments

object	output of the function hill.adapt.
plot	If TRUE, the test statistics are plotted.
...	further arguments passed to or from other methods.

Value

TS.window	the test statistic inside the window. (pages 447 and 448 of Durrieu et al.(2015))
TS	the test statistic.
CritVal	the critical value of the test.

References

Grama, I. and Spokoiny, V. (2008). Statistics of extremes by oracle estimation. Ann. of Statist., 36, 1619-1648.

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

See Also

hill.adapt, goftest

Examples

```
x <- abs(rcauchy(100))
HH <- hill.adapt(x, weights=rep(1, length(x)), initprop = 0.1,
               gridlen = 100 , r1 = 0.25, r2 = 0.05, CritVal=10)

#the critical value 10 is assiociated to the rectangular kernel.

goftest(HH, plot = TRUE)
```

```
# we observe that for this data, the null hypothesis that the tail
# is fitted by a Pareto distribution is not rejected as the maximal
# value in the graph does not exceed the critical value.
```

`goftest.hill.ts` *Goodness of fit test statistics for time series*

Description

Give the results of the goodness of fit test for testing the null hypothesis that the tail is fitted by a Pareto distribution starting from the adaptive threshold (for more details see pages 447 and 448 of Durrieu et al. (2015)).

Usage

```
## S3 method for class 'hill.ts'
goftest(object, X, t, plot = FALSE, ...)
```

Arguments

`object`	output of the hill.ts function.
`X`	a vector of the observed values.
`t`	a vector of time covariates which should have the same length as X.
`plot`	If `TRUE`, the test statistic are plotted.
`...`	further arguments passed to or from other methods.

Value

`TS.window`	the maximum value of test statistics inside the window for each t in Tgrid (see help(hill.ts)).
`TS.max`	the maximum value of test statistics for each t in Tgrid (see help(hill.ts)).
`CritVal`	the critical value of the test.

References

Grama, I. and Spokoiny, V. (2008). Statistics of extremes by oracle estimation. Ann. of Statist., 36, 1619-1648.

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

See Also

`hill.ts`, `goftest`

Examples

```
theta<-function(t){0.5+0.25*sin(2*pi*t)}
n<-5000
t<-1:n/n
Theta<-theta(t)
Data<-NULL
Tgrid<-seq(0.01,0.99,0.01)
#example with fixed bandwidth
for(i in 1:n){Data[i]<-rparetomix(1,a=1/Theta[i],b=5/Theta[i]+5,c=0.75,precision=10^(-5))}
## Not run:  #For computing time purpose
  #example
  hgrid <- bandwidth.grid(0.009, 0.2, 20, type = "geometric")
  TgridCV <- seq(0.01, 0.99, 0.1)
  hcv <- bandwidth.CV(Data, t, TgridCV, hgrid, pcv = 0.99,
         TruncGauss.kernel, kpar = c(sigma = 1), CritVal = 3.6, plot = TRUE)

  Tgrid <- seq(0.01,0.99,0.01)
  hillTs <- hill.ts(Data, t, Tgrid, h = hcv$h.cv, TruncGauss.kernel, kpar = c(sigma = 1),
                  CritVal = 3.6, gridlen = 100, initprop = 1/10, r1 = 1/4, r2 = 1/20)
  goftest(hillTs, Data, t, plot = TRUE)

  # we observe that for this data, the null hypothesis that the tail
  # is fitted by a Pareto distribution is not rejected
  # for all points on the Tgrid


## End(Not run)
```

`hill` *Hill estimator*

Description

Compute the weighted Hill estimator.

Usage

```
hill(X, weights = rep(1, length(X)), grid = X)
```

Arguments

`X`	a vector of data.
`weights`	a vector of weights assiociated to x.
`grid`	a vector of values for which the Hill estimator is computed.

Details

Compute the weighted Hill estimator for vectors $grid$, data and weights (see references below).

Value

xsort	the sorted data.
wsort	the weights assiociated to $xsort$.
grid	the grid for which the Hill estimator is computed.
hill	the Hill estimators.

Author(s)

Ion Grama

References

Grama, I. and Spokoiny, V. (2008). Statistics of extremes by oracle estimation. Ann. of Statist., 36, 1619-1648.

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

Hill, B.M. (1975). A simple general approach to inference about the tail of a distribution. Annals of Statistics, 3, 1163-1174.

Examples

```
X <- abs(rcauchy(100))
weights <- rep(1, length(X))
wh <- hill(X, w = weights)
```

---

hill.adapt *Compute the extreme quantile procedure*

Description

Compute the extreme quantile procedure

Usage

```
hill.adapt(
  X,
  weights = rep(1, length(X)),
  initprop = 1/10,
  gridlen = 100,
  r1 = 1/4,
  r2 = 1/20,
  CritVal = 10,
  plot = F
)
```

Arguments

X	a numeric vector of data values.
weights	a numeric vector of weigths associated to the vector X.
initprop	the initial proportion at which we begin to test the model.
gridlen	the length of the grid for which the test is done.
r1	a proportion value of the data from the right that we skip in the test statistic.
r2	a proportion value of the data from the left that we skip in the test statistic.
CritVal	the critical value assiociated to the weights.
plot	If TRUE, the results are plotted.

Details

Given a vector of data and assiociated weights, this function compute the adaptive procedure described in Grama and Spokoiny (2008) and Durrieu et al. (2015).

We suppose that the data are in the domain of attraction of the Frechet-Pareto type. Otherwise, the procedure will not work.

Value

Xsort	the sorted vector of the data.
sortweights	the weights associated to Xsort.
wh	the weighted Hill estimator associated to X (output of the function hill).
TestingGrid	the grid used for the statistic test.
TS, TS1, TS.max, TS1.max	respectively the test statistic, the likelihood ratio test, the maximum of the test statistic and the maximum likelihood ratio test.
Paretodata	logical: if TRUE the distribution of the data is a Pareto distribution.
Paretotail	logical: if TRUE a Pareto tail was detected.
madapt	the first indice of the TestingGrid for which the test statistic exceeds the critical value.
kadapt	the adaptive indice of the threshold.
kadapt.maxlik	the maximum likelihood corresponding to the adaptive threshold in the selected testing grid.
hadapt	the adaptive weighted parameter of the Pareto distribution after the threshold.
Xadapt	the adaptive threshold.

Author(s)

Ion Grama

References

Grama, I. and Spokoiny, V. (2008). Statistics of extremes by oracle estimation. Ann. of Statist., 36, 1619-1648.

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M. (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

Durrieu, G. and Grama, I. and Jaunatre, K. and Pham, Q.-K. and Tricot, J.-M. (2018). extremefit: A Package for Extreme Quantiles. Journal of Statistical Software, 87, 1–20.

Examples

```
x <- abs(rcauchy(100))
HH <- hill.adapt(x, weights=rep(1, length(x)), initprop = 0.1,
               gridlen = 100 , r1 = 0.25, r2 = 0.05, CritVal=10,plot=TRUE)
#the critical value 10 is assiociated to the rectangular kernel.
HH$Xadapt # is the adaptive threshold
HH$hadapt # is the adaptive parameter of the Pareto distribution
```

`hill.ts` *Compute the extreme quantile procedure on a time dependent data*

Description

Compute the function hill.adapt on time dependent data.

Usage

```
hill.ts(
  X,
  t,
  Tgrid = seq(min(t), max(t), length = 10),
  h,
  kernel = TruncGauss.kernel,
  kpar = NULL,
  CritVal = 3.6,
  gridlen = 100,
  initprop = 1/10,
  r1 = 1/4,
  r2 = 1/20
)

## S3 method for class 'hill.ts'
print(x, ...)
```

Arguments

X	a vector of the observed values.
t	a vector of time covariates which should have the same length as X.
Tgrid	a grid of time (can be any sequence in the interval [min(t) , max(t)]).
h	a bandwidth value (vector values are not admitted).
kernel	a kernel function used to compute the weights in the time domain, with default the truncated Gaussian kernel.
kpar	a value for the kernel function parameter, with no default value.
CritVal	a critical value associated to the kernel function given by CriticalValue. The default value is 3.6 corresponding to the truncated Gaussian kernel.
gridlen	the gridlen parameter used in the function hill.adapt. The length of the grid for which the test will be done.
initprop	the initprop parameter used in the function hill.adapt. The initial proportion at which we will begin to test the model.
r1	the r1 parameter used in the function hill.adapt. The proportion from the right that we will skip in the test statistic.
r2	the r2 parameter used in the function hill.adapt. The proportion from the left that we will skip in the test statistic.
x	the result of the hill.ts function
...	further arguments to be passed from or to other methods.

Details

For a given time serie and kernel function, the function hill.ts will give the results of the adaptive procedure for each t. The adaptive procedure is described in Durrieu et al. (2005).

The kernel implemented in this packages are : Biweight kernel, Epanechnikov kernel, Rectangular kernel, Triangular kernel and the truncated Gaussian kernel.

Value

Tgrid	the given vector $Tgrid$.
h	the given value h.
Threshold	the adaptive threshold τ for each t in $Tgrid$.
Theta	the adaptive estimator of θ for each t in $Tgrid$.

References

Durrieu, G. and Grama, I. and Pham, Q. and Tricot, J.- M (2015). Nonparametric adaptive estimator of extreme conditional tail probabilities quantiles. Extremes, 18, 437-478.

Durrieu, G. and Grama, I. and Jaunatre, K. and Pham, Q.-K. and Tricot, J.-M. (2018). extremefit: A Package for Extreme Quantiles. Journal of Statistical Software, 87, 1–20.

See Also

hill.adapt, Biweight.kernel, Epa.kernel, Rectangular.kernel, Triang.kernel, TruncGauss.kernel

Examples

```
theta <- function(t){
   0.5+0.25*sin(2*pi*t)
 }
n <- 5000
t <- 1:n/n
Theta <- theta(t)
Data <- NULL
Tgrid <- seq(0.01, 0.99, 0.01)
#example with fixed bandwidth
## Not run:  #For computing time purpose
  for(i in 1:n){
   Data[i] <- rparetomix(1, a = 1/Theta[i], b = 5/Theta[i]+5, c = 0.75, precision = 10^(-5))
   }

  #example
  hgrid <- bandwidth.grid(0.009, 0.2, 20, type = "geometric")
  TgridCV <- seq(0.01, 0.99, 0.1)
  hcv <- bandwidth.CV(Data, t, TgridCV, hgrid, pcv = 0.99, TruncGauss.kernel,
                    kpar = c(sigma = 1), CritVal = 3.6, plot = TRUE)

  Tgrid <- seq(0.01, 0.99, 0.01)
  hillTs <- hill.ts(Data, t, Tgrid, h = hcv$h.cv, kernel = TruncGauss.kernel,
         kpar = c(sigma = 1), CritVal = 3.6,gridlen = 100, initprop = 1/10, r1 = 1/4, r2 = 1/20)
  plot(hillTs$Tgrid, hillTs$Theta, xlab = "t", ylab = "Estimator of theta")
  lines(t, Theta, col = "red")


## End(Not run)
```

LoadCurve *Load curve of an habitation*

Description

The data frame provides electric consumption of an habitation in France over one month.

Usage

```
data("LoadCurve")
```

Format

The data is the electric consumption of an habitation in Kilovolt-amps (kVA) every 10 minutes during one month. The habitation has a contract that allows a maximum power of 6 kVA.A list of 2 elements.

`$data` : a `data frame` with `24126` observations for `2` variables `Time` the number of day since the 1st of January, 1970.

`Value` the value of the electric consumtion in kVA.

`$Tgrid` : A `grid` of `time` to `perform` the `procedure`.

Source

Electricite Reseau Distribution France

Examples

```
data("LoadCurve")

X<-LoadCurve$data$Value
days<-LoadCurve$data$Time
Tgrid <- seq(min(days), max(days), length = 400)
new.Tgrid <- LoadCurve$Tgrid
## Not run:  #For computing time purpose
# Choice of the bandwidth by cross validation.
# We choose the truncated Gaussian kernel and the critical value
# of the goodness-of-fit test 3.4.
# As the computing time is high, we give the value of the bandwidth.

#hgrid <- bandwidth.grid(0.8, 5, 60)
#hcv<-bandwidth.CV(X=X, t=days, new.Tgrid, hgrid, pcv = 0.99,
#                  kernel = TruncGauss.kernel, CritVal = 3.4, plot = FALSE)
#h.cv <- hcv$h.cv

h.cv <- 3.444261
HH<-hill.ts(X, days, new.Tgrid, h=h.cv, kernel = TruncGauss.kernel, CritVal = 3.4)

Quant<-rep(NA,length(Tgrid))
Quant[match(new.Tgrid, Tgrid)]<-as.numeric(predict(HH,
            newdata = 0.99, type = "quantile")$y)

Date<-as.POSIXct(days*86400, origin = "1970-01-01",
                 tz = "Europe/Paris")
plot(Date, X/1000, ylim = c(0, 8),
      type = "l", ylab = "Electric consumption (kVA)", xlab = "Time")

lines(as.POSIXlt((Tgrid)*86400, origin = "1970-01-01",
                 tz = "Europe/Paris"), Quant/1000, col = "red")

## End(Not run)
```

`Pareto Distribution` *Pareto distribution*

Description

Density, distribution function, quantile function and random generation for the Pareto distribution where a, loc and $scale$ are respectively the shape, the location and the scale parameters.

Usage

```
ppareto(q, a = 1, loc = 0, scale = 1)

dpareto(x, a = 1, loc = 0, scale = 1)

qpareto(p, a = 1, loc = 0, scale = 1)

rpareto(n, a = 1, loc = 0, scale = 1)
```

Arguments

`q`	a vector of quantiles.
`a`	a vector of shape parameter of the Pareto distribution.
`loc`	a vector of location parameter of the Pareto distribution.
`scale`	a vector of scale parameter of the Pareto distribution.
`x`	a vector of quantiles.
`p`	a vector of probabilities.
`n`	a number of observations. If length(n) > 1, the length is taken to be the number required.

Details

If $shape$, loc or $scale$ parameters are not specified, the respective default values are 1, 0 and 1.

The cumulative Pareto distribution is

$$F(x) = 1 - ((x - loc)/scale)^{-1/a}, x > 0, a > 0, scale > 0$$

where a is the shape of the distribution.

Value

dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, and rpareto generates random deviates.

The length of the result is determined by n for rpareto, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
par(mfrow = c(3,1))
plot(function(x) dpareto(x), 1, 5,ylab="density",
     main = " Pareto density ")

plot(function(x) ppareto(x), 1, 5,ylab="distribution function",
     main = " Pareto Cumulative ")

plot(function(x) qpareto(x), 0, 1,ylab="quantile",
     main = " Pareto Quantile ")

#generate a sample of pareto distribution of size n
n <- 100
x <- rpareto(n)
```

`Pareto mix` *Pareto mixture distribution*

Description

Density, distribution function, quantile function and random generation for the Pareto mixture distribution with a equal to the shape of the first Pareto Distribution, b equal to the shape of the second Pareto Distribution and c is the mixture proportion. The locations and the scales parameters are equals to 0 and 1.

Usage

```
pparetomix(q, a = 1, b = 2, c = 0.75)

dparetomix(x, a = 1, b = 2, c = 0.75)

qparetomix(
  p,
  a = 1,
  b = 2,
  c = 0.75,
  precision = 10^(-10),
  initvalue = 0.5,
  Nmax = 1000
)

rparetomix(n, a = 1, b = 2, c = 0.75, precision = 10^(-10))
```

Arguments

`q`	a vector of quantiles.
`a`	the shape parameter of the first Pareto Distribution.

b	the shape parameter of the second Pareto Distribution.
c	the value of the mixture proportion.
x	a vector of quantiles.
p	a vector of probabilities.
precision	the precision of the Newton method.
initvalue	the initial value of the Newton method.
Nmax	the maximum of iteration done for the Newton method.
n	the number of observations. If length(n) > 1, the length is taken to be the number required.

Details

If the a, b and c are not specified, they respectively take the default values 1, 2 and 0.75.

The cumulative Pareto mixture distribution is

$$F(x) = c(1 - x^{-a}) + (1 - c)(1 - x^{-b}), x \geq 1, a > 0, b > 0, 0 \leq c \leq 1$$

where a and b are the shapes of the distribution and c is the mixture proportion.

Value

dparetomix gives the density, pparetomix gives the distribution function, qparetomix gives the quantile function, and rparetomix generates random deviates.

The length of the result is determined by n for rparetomix, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
par(mfrow = c(3,1))
plot(function(x) dparetomix(x), 0, 5,ylab="density",
     main = " Pareto mixture density ")
mtext("dparetomix(x)", adj = 0)

plot(function(x) pparetomix(x), 0, 5,ylab="distribution function",
     main = " Pareto mixture Cumulative ")
mtext("pparetomix(x)", adj = 0)

plot(function(x) qparetomix(x), 0, 1,ylim=c(0,5),ylab="quantiles",
     main = " Pareto mixture Quantile ")
mtext("qparetomix(x)", adj = 0)

#generate a sample of the Pareto mix distribution of size n
n <- 100
x <- rparetomix(n)
```

`plot.hill` *Hill plot*

Description

Graphical representation of the hill estimator.

Usage

```
## S3 method for class 'hill'
plot(x, xaxis = "ranks", ...)
```

Arguments

`x`	output object of the function hill.
`xaxis`	either "ranks" or "xsort".
`...`	further arguments passed to or from other methods.

Details

If xaxis="ranks", the function draws the Hill estimators for each ranks of the grid output of the function hill. If xaxis="xsort", the function draws the Hill estimators for each data of the grid output of the function hill.

See Also

`hill`

Examples

```
x <- abs(rcauchy(100))
hh <- hill(x)
par(mfrow = c(2, 1))
plot(hh, xaxis = "ranks")
plot(hh, xaxis = "xsort")
```

`plot.hill.adapt` *Hill.adapt plot*

Description

Graphical representation of the hill.adapt function last iteration

Usage

```
## S3 method for class 'hill.adapt'
plot(x, ...)
```

Arguments

`x`	output object of the function hill.adapt.
`...`	further arguments passed to or from other methods.

Details

The weighted hill estimator, the test statistic, the penalized likelihood graphs of the last iteration and the survival function are given. The blue line corresponds to the threshold (indice or value). The magenta lines correspond to the window (r1, r2) where the estimation is computed. The red lines corresponds to the initial proportion (initprop) and the last non rejected point of the statistic test (madapt).

See Also

`hill.adapt`, `plot`

Examples

```
x <- abs(rcauchy(100))
HH <- hill.adapt(x, weights=rep(1, length(x)), initprop = 0.1,
                 gridlen = 50 , r1 = 0.25, r2 = 0.05, CritVal=10)
plot(HH)
```

pparetoCP *Pareto change point distribution*

Description

Distribution function, quantile function and random generation for the Pareto change point distribution with $a0$ equal to the shape of the first pareto distribution, $a1$ equal to the shape of the second pareto distribution, $x0$ equal to the scale and $x1$ equal to the change point.

Usage

```
pparetoCP(x, a0 = 1, a1 = 2, x0 = 1, x1 = 6)

qparetoCP(p, a0 = 1, a1 = 2, x0 = 1, x1 = 6)

rparetoCP(n, a0 = 1, a1 = 2, x0 = 1, x1 = 6)
```

Arguments

x	a vector of quantiles.
a0	a vector of shape parameter of the Pareto distribution before $x1$.
a1	a vector of shape parameter of the Pareto distribution after $x1$.
x0	a vector of scale parameter of the function.
x1	a vector of change point value.
p	a vector of probabilities.
n	a number of observations. If length(n) > 1, the length is taken to be the number required.

Details

If not specified, $a0, a1, x0$ and $x1$ are taking respectively the values $1, 2, 1$ and 6

The cumulative Pareto change point distribution is given by :

$$F(x) = (x <= x1) * (1 - x^{-a0}) + (x > x1) * (1 - x^{-a1} * x1^{-a0+a1})$$

Value

pparetoCP gives the distribution function, qparetoCP gives the quantile function, and rparetoCP generates random deviates.

The length of the result is determined by n for rparetoCP, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Examples

```
par(mfrow = c(2,1))

plot(function(x) pparetoCP(x), 0, 5,ylab="distribution function",
     main = " Pareto change point Cumulative ")
mtext("pparetoCP(x)", adj = 0)

plot(function(x) qparetoCP(x), 0, 1,ylab="quantiles",
     main = " Pareto change point Quantile")
mtext("qparetoCP(x)", adj = 0)

#generate a sample of pareto distribution of size n
n <- 100
x <- rparetoCP(n)
```

`predict.cox.adapt` *Predict the survival or quantile function from the extreme procedure for the Cox model*

Description

Give the survival or quantile function from the extreme procedure for the Cox model

Usage

```
## S3 method for class 'cox.adapt'
predict(
  object,
  newdata = NULL,
  input = NULL,
  type = "quantile",
  aggregation = "none",
  AggInd = object$kadapt,
  M = 10,
  ...
)
```

Arguments

`object`	output object of the function cox.adapt.
`newdata`	a data frame with which to predict.
`input`	optionnaly, the name of the variable to estimate.
`type`	either "quantile" or "survival".
`aggregation`	either "none", "simple" or "adaptive".
`AggInd`	Indices of thresholds to be aggregated.
`M`	Number of thresholds to be aggregated.
`...`	further arguments passed to or from other methods.

Details

$newdata$ must be a data frame with the co-variables from which to predict and a variable of probabilities with its name starting with a "p" if type = "quantile" or a variable of quantiles with its name starting with a "x" if type = "survival". The name of the variable from which to predict can also be written as $input$.

Value

The function provide the quantile assiociated to the adaptive model for the probability grid if type = "quantile". And the survival function assiociated to the adaptive model for the quantile grid if type = "survival".

See Also

cox.adapt

Examples

```
library(survival)
data(bladder)

X <- bladder2$stop-bladder2$start
Z <- as.matrix(bladder2[, c(2:4, 8)])
delta <- bladder2$event

ord <- order(X)
X <- X[ord]
Z <- Z[ord,]
delta <- delta[ord]

cph<-coxph(Surv(X, delta) ~ Z)

ca <- cox.adapt(X, cph, delta, bladder2[ord,])

xgrid <- X
newdata <- as.data.frame(cbind(xgrid,bladder2[ord,]))

Plac <- predict(ca, newdata = newdata, type = "survival")
Treat <- predict(ca, newdata = newdata, type = "survival")

PlacSA <- predict(ca, newdata = newdata,
                      type = "survival", aggregation = "simple", AggInd = c(10,20,30,40))
TreatSA <- predict(ca, newdata = newdata,
                      type = "survival", aggregation = "simple", AggInd = c(10,20,30,40))


PlacAA <- predict(ca, newdata = newdata,
                          type = "survival", aggregation = "adaptive", M=10)
TreatAA <- predict(ca, newdata = newdata,
                          type = "survival", aggregation = "adaptive", M=10)
```

predict.hill *Predict the adaptive survival or quantile function*

Description

Give the adaptive survival function or quantile function

Usage

```
## S3 method for class 'hill'
predict(
  object,
  newdata = NULL,
  type = "quantile",
  input = NULL,
  threshold.rank = 0,
  threshold = 0,
  ...
)
```

Arguments

object	output object of the function hill.
newdata	optionally, a data frame or a vector with which to predict. If omitted, the original data points are used.
type	either "quantile" or "survival".
input	optionnaly, the name of the variable to estimate.
threshold.rank	the rank value for the hill output of the threshold, with default value 0.
threshold	the value of threshold, with default value 0.
...	further arguments passed to or from other methods.

Details

If type = "quantile", $newdata$ must be between 0 and 1. If type = "survival", $newdata$ must be in the domain of the data from the hill function. If $newdata$ is a data frame, the variable from which to predict must be the first one or its name must start with a "p" if type = "quantile" and "x" if type = "survival". The name of the variable from which to predict can also be written as $input$.

Value

The function provide the quantile assiociated to the adaptive model for the probability grid (transformed to -log(1-p) in the output) if type = "quantile". And the survival function assiociated to the adaptive model for the quantile grid if type = "survival".

See Also

hill

Examples

```
x <- abs(rcauchy(100))
hh <- hill(x)
#example for a fixed value of threshold
predict(hh, threshold = 3)
#example for a fixed rank value of threshold
predict(hh, threshold.rank = 30)
```

---

predict.hill.adapt *Predict the adaptive survival or quantile function*

Description

Give the adaptive survival function or quantile function

Usage

```
## S3 method for class 'hill.adapt'
predict(object, newdata = NULL, type = "quantile", input = NULL, ...)
```

Arguments

object	output object of the function hill.adapt.
newdata	optionally, a data frame or a vector with which to predict. If omitted, the original data points are used.
type	either "quantile" or "survival".
input	optionnaly, the name of the variable to estimate.
...	further arguments passed to or from other methods.

Details

If type = "quantile", $newdata$ must be between 0 and 1. If type = "survival", $newdata$ must be in the domain of the data from the hill.adapt function. If $newdata$ is a data frame, the variable from which to predict must be the first one or its name must start with a "p" if type = "quantile" and "x" if type = "survival". The name of the variable from which to predict can also be written as $input$.

Value

The function provide the quantile assiociated to the adaptive model for the probability grid (transformed to -log(1-p) in the output) if type = "quantile". And the survival function assiociated to the adaptive model for the quantile grid if type = "survival".

References

Durrieu, G. and Grama, I. and Jaunatre, K. and Pham, Q.-K. and Tricot, J.-M. (2018). extremefit: A Package for Extreme Quantiles. Journal of Statistical Software, 87, 1–20.

See Also

hill.adapt

Examples

```
x <- rparetoCP(1000)

HH <- hill.adapt(x, weights=rep(1, length(x)), initprop = 0.1,
                gridlen = 100 , r1 = 0.25, r2 = 0.05, CritVal=10)

newdata <- probgrid(p1 = 0.01, p2 = 0.999, length = 100)
pred.quantile <- predict(HH, newdata, type = "quantile")
newdata <- seq(0, 50, 0.1)
pred.survival <- predict(HH, newdata, type = "survival")#survival function

#compare the theorical quantile and the adaptive one.
predict(HH, 0.9999, type = "quantile")
qparetoCP(0.9999)

#compare the theorical probability and the adaptive one assiociated to a quantile.
predict(HH, 20, type = "survival")
1 - pparetoCP(20)
```

`predict.hill.ts` *Predict the adaptive survival or quantile function for a time serie*

Description

Give the adaptive survival function or quantile function of a time serie

Usage

```
## S3 method for class 'hill.ts'
predict(object, newdata = NULL, type = "quantile", input = NULL, ...)
```

Arguments

`object`	output object of the function hill.ts.
`newdata`	optionally, a data frame or a vector with which to predict. If omitted, the original data points are used.
`type`	either "quantile" or "survival".
`input`	optionnaly, the name of the variable to estimate.
`...`	further arguments passed to or from other methods.

Details

If type = "quantile", $newdata$ must be between 0 and 1. If type = "survival", $newdata$ must be in the domain of the data from the function `hill.ts`. If $newdata$ is a data frame, the variable from which to predict must be the first one or its name must start with a "p" if type = "quantile" and "x" if type = "survival". The name of the variable from which to predict can also be written as $input$.

Value

`p`	the input vector of probabilities.
`x`	the input vector of values.
`Tgrid`	Tgrid output of the function hill.ts.
`quantiles`	the estimted quantiles assiociated to newdata.
`survival`	the estimated survival function assiociated to newdata.

References

Durrieu, G. and Grama, I. and Jaunatre, K. and Pham, Q.-K. and Tricot, J.-M. (2018). extremefit: A Package for Extreme Quantiles. Journal of Statistical Software, 87, 1–20.

See Also

`hill.ts`

Examples

```
#Generate a pareto mixture sample of size n with a time varying parameter
theta <- function(t){
   0.5+0.25*sin(2*pi*t)
 }
n <- 4000
t <- 1:n/n
Theta <- theta(t)
Data <- NULL
set.seed(1240)
for(i in 1:n){
  Data[i] <- rparetomix(1, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75, precision = 10^(-5))
 }
## Not run:  #For computing time purpose
  #choose the bandwidth by cross validation
  Tgrid <- seq(0, 1, 0.1)#few points to improve the computing time
  hgrid <- bandwidth.grid(0.01, 0.2, 20, type = "geometric")
  hcv <- bandwidth.CV(Data, t, Tgrid, hgrid, TruncGauss.kernel,
         kpar = c(sigma = 1), pcv = 0.99, CritVal = 3.6, plot = TRUE)
  h.cv <- hcv$h.cv

  #we modify the Tgrid to cover the data set
  Tgrid <- seq(0, 1, 0.02)
  hillTs <- hill.ts(Data, t, Tgrid, h = h.cv, TruncGauss.kernel, kpar = c(sigma = 1),
           CritVal = 3.6, gridlen = 100, initprop = 1/10, r1 = 1/4, r2 = 1/20)
  p <- c(0.999)
```

```
pred.quantile.ts <- predict(hillTs, newdata = p, type = "quantile")
true.quantile <- NULL
for(i in 1:n){
   true.quantile[i] <- qparetomix(p, a = 1/Theta[i], b = 1/Theta[i]+5, c = 0.75)
 }
plot(Tgrid, log(as.numeric(pred.quantile.ts$y)),
   ylim = c(0, max(log(as.numeric(pred.quantile.ts$y)))), ylab = "log(0.999-quantiles)")
lines(t, log(true.quantile), col = "red")
lines(t, log(Data), col = "blue")


#comparison with other fixed bandwidths

plot(Tgrid, log(as.numeric(pred.quantile.ts$y)),
   ylim = c(0, max(log(as.numeric(pred.quantile.ts$y)))), ylab = "log(0.999-quantiles)")
lines(t, log(true.quantile), col = "red")

hillTs <- hill.ts(Data, t, Tgrid, h = 0.1, TruncGauss.kernel, kpar = c(sigma = 1),
                  CritVal = 3.6, gridlen = 100,initprop = 1/10, r1 = 1/4, r2 = 1/20)
pred.quantile.ts <- predict(hillTs, p, type = "quantile")
lines(Tgrid, log(as.numeric(pred.quantile.ts$y)), col = "green")


hillTs <- hill.ts(Data, t, Tgrid, h = 0.3, TruncGauss.kernel, kpar = c(sigma = 1),
             CritVal = 3.6, gridlen = 100, initprop = 1/10, r1 = 1/4, r2 = 1/20)
pred.quantile.ts <- predict(hillTs, p, type = "quantile")
lines(Tgrid, log(as.numeric(pred.quantile.ts$y)), col = "blue")


hillTs <- hill.ts(Data, t, Tgrid, h = 0.04, TruncGauss.kernel, kpar = c(sigma = 1),
           CritVal = 3.6, gridlen = 100, initprop = 1/10, r1 = 1/4, r2 = 1/20)
pred.quantile.ts <- predict(hillTs ,p, type = "quantile")
lines(Tgrid, log(as.numeric(pred.quantile.ts$y)), col = "magenta")

## End(Not run)
```

`probgrid` *Probability grid*

Description

Create a geometric grid of probabilities

Usage

```
probgrid(p1, p2, length = 50)
```

Arguments

p1	the first element of the grid.
p2	the last element of the grid.
length	the length of the grid.

Details

Create a geometric grid of length $length$ between $p1$ and $p2$.The default value of $length$ is 50.

Value

A vector of probabilities between $p1$ and $p2$ and length $length$.

Examples

```
p1 <- 0.01
p2 <- 0.99
length <- 500
pgrid <- probgrid(p1, p2, length)
```

---

rburr.dependent *Generate Burr dependent data*

Description

Random generation function for the dependent Burr with a, b two shapes parameters and alpha the dependence parameter.

Usage

```
rburr.dependent(n, a, b, alpha)
```

Arguments

n	the number of observations. If length(n) > 1, the length is taken to be the number required.
a	a parameter of the function.
b	a parameter of the function.
alpha	the dependence parameter. It is defined by a single value between 0 and 1. The value 1 corresponds to the full independence. The closer to 0 the value of alpha is, the stronger is the dependence. $alpha$ cannot take the value 0.

Details

The description of the dependence is described in *Fawcett and Walshaw (2007)*. The Burr distribution is : $F(x) = 1 - (1 + (x^a))^{-b}, x > 0, a > 0, b > 0$ where a and b are shapes of the distribution.

Value

Generates a vector of random deviates. The length of the result is determined by n.

References

Fawcett, D. and Walshaw, D. (2007). Improved estimation for temporally clustered extremes. Environmetrics, 18.2, 173-188.

Examples

```
theta <- function(t){
   1/2*(1/10+sin(pi*t))*(11/10-1/2*exp(-64*(t-1/2)^2))
 }
n <- 200
t <- 1:n/n
Theta <- theta(t)
plot(theta)
alpha <- 0.6
Burr.dependent <- rburr.dependent(n, 1/Theta, 1, alpha)
```

Rectangular.kernel *Rectangular kernel function*

Description

Rectangular kernel function

Usage

```
Rectangular.kernel(x)
```

Arguments

x	a vector.

Details

Rectangular kernel function

Rectangular Kernel

$$K(x) = 1(abs(x) <= 1)$$

We recommend a critical value of 10 for this kernel.

Examples

```
plot(function(x) Rectangular.kernel(x), -2, 2,
main = " Rectangular kernel ")
```

Triang.kernel *Triangular kernel function*

Description

Triangular kernel function

Usage

```
Triang.kernel(x)
```

Arguments

x	a vector.

Details

Triangular Kernel

$$K(x) = (1 - abs(x))(abs(x) <= 1)$$

We recommend a critical value of 6.9 for this kernel.

Examples

```
plot(function(x) Triang.kernel(x), -2, 2,
main = " Triangular kernel")
```

TruncGauss.kernel *Truncated Gaussian kernel function*

Description

Truncated Gaussian kernel function

Usage

```
TruncGauss.kernel(x, sigma = 1)
```

Arguments

x	a vector.
sigma	the standard deviation of the truncated gaussian kernel.

Details

Truncated Gaussian Kernel with $sigma$ the standard deviation parameter with default value 1.

$$K(x) = (1/sigma * sqrt(2\pi)exp(-(x/sigma)^2/2))(abs(x) <= 1)$$

We recommend a critical value of 3.6 for this kernel with sigma=1.

Examples

```
plot(function(x) TruncGauss.kernel(x), -2, 2,
main = " Truncated Gaussian kernel")
```

`wecdf` *Weighted empirical cumulative distribution function*

Description

Calculate the values of the weighted empirical cumulative distribution function for a given vector of data

Usage

```
wecdf(X, x, weights = rep(1, length(X)))
```

Arguments

`X`	the vector of data to create the wecdf.
`x`	the vector of data that you want the corresponding wecdf values.
`weights`	the weights applicated to the vector X.

Details

Give the value of the wecdf. If the weights are 1 (the default value), the wecdf become the ecdf of X.

Value

Return a vector of the wecdf values corresponding to x given a reference vector X with weights $weights$.

Examples

```
X <- rpareto(10)
x <- seq(0.8, 50, 0.01)
plot(x, wecdf(X, x, rep(1,length(X))))

#to compare with the ecdf function
f <- ecdf(X)
lines(x, f(x), col = "red", type = "s")
```

`wquantile` *Weighted quantile*

Description

Compute the weighted quantile of order p.

Usage

```
wquantile(X, p, weights = rep(1, length(X)))
```

Arguments

`X`	a vector of data.
`p`	a vector of probabilities.
`weights`	the weights assiociated to the vector X.

Details

Give the weighted quantile for a given p

Value

A vector of quantile assiociated to the probabilities vector given in input.

Examples

```
X <- rpareto(10)
p <- seq(0.01, 0.99, 0.01)
plot(p, wquantile(X, p, rep(1,length(X))), type = "s")
```

Index