Package ‘modeest’

February 20, 2015

Type Package
Title Mode Estimation
Version 2.1
Date 2012-10-15
Author P. PONCET
Description This package provides estimators of the mode of univariate unimodal data or univariate unimodal distributions
Maintainer P. PONCET <paul.poncet@yahoo.fr>
Depends R (>= 2.12.0)
Suggests stats, evd, fBasics, stabledist, VGAM
License GPL (>= 2)
Repository CRAN
Date/Publication 2012-10-15 11:21:35
NeedsCompilation yes

R topics documented:

asselin ................................................................. 2
distribMode ............................................................ 3
grenander ............................................................... 7
hrm ........................................................................ 9
hsm ....................................................................... 11
lientz ....................................................................... 12
mfv ....................................................................... 15
mlv ....................................................................... 16
modeest ................................................................. 20
naive ..................................................................... 22
parzen ............................................................... 24
skewness ............................................................... 25
tsybakov ............................................................... 27
venter ................................................................. 29
vieu .................................................................... 31
The Asselin de Beauville Mode Estimator

Description

This mode estimator is based on the algorithm described in Asselin de Beauville (1978).

Usage

```r
asselin(x, 
   bw = NULL, 
   ...) 
```

Arguments

- `x` : numeric. Vector of observations.
- `bw` : numeric. A number in \((0, 1]\). If \(bw = 1\), the selected 'modal chain' may be too long.
- `...` : further arguments to be passed to the `quantile` function.

Value

A numeric value is returned, the mode estimate.

Note

The user should preferentially call `asselin` through `mlv(x, method = "asselin", ...)`. This returns an object of class `mlv`.

Author(s)

P. Poncet

References


See Also

`mlv` for general mode estimation
Examples

```r
x <- rbeta(1000, shape1 = 2, shape2 = 5)

## True mode:
betaMode(shape1 = 2, shape2 = 5)

## Estimation:
asselin(x, bw = 1)
asselin(x, bw = 1/2)
M <- mlv(x, method = "asselin")
print(M)
plot(M)
```

Description

These functions return the mode of the main probability distributions implemented in R.

Usage

```r
## Continuous distributions

# Beta
betaMode(shape1, shape2, ncp = 0)

# Cauchy
cauchyMode(location = 0, ...)

# Chisquare
chisqMode(df, ncp = 0)

# Exponentiel
expMode(...)

# F
fMode(df1, df2)

# Fisk (package 'VGAM')
fiskMode(shape1.a, scale = 1)

# Frechet (package 'evd')
frechetMode(loc = 0, scale = 1, shape = 1, ...)

# Gamma
gammaMode(shape, rate = 1, scale = 1/rate)
```
# Normal (Gaussian)
normMode(mean = 0, ...)

# Generalised Extreme Value (package 'evd')
gevMode(loc = 0, scale = 1, shape = 0, ...)

# Generalised Hyperbolic (package 'fBasics')
ghMode(alpha = 1, beta = 0, delta = 1, mu = 0, 
        lambda = -1/2, ...)

# Gompertz (package 'VGAM')
gompertzMode(shape, scale = 1)

# Generalised Pareto (package 'evd')
gpdMode(loc = 0, scale = 1, shape = 0, ...)

# Gumbel (package 'evd')
gumbelMode(loc = 0, ...)

# Hyperbolic (package 'fBasics')
hypMode(alpha = 1, beta = 0, delta = 1, mu = 0, 
        pm = c(1, 2, 3, 4))

# Koenker (package 'VGAM')
koenkerMode(location = 0, ...)

# Kumaraswamy (package 'VGAM')
kumarMode(shape1, shape2)

# Laplace (package 'VGAM')
laplaceMode(location = 0, ...)

# Logistic
logisMode(location = 0, ...)

# Lognormal
lnormMode(meanlog = 0, sdlog = 1)

# Normal Inverse Gaussian (package 'fBasics')
nigMode(alpha = 1, beta = 0, delta = 1, mu = 0, ...)

# Stable (package 'stabledist')
stableMode(alpha, beta, gamma = 1, delta = 0, pm = 0, ...)

# Negative Weibull (package 'evd')
rweibullMode(loc = 0, scale = 1, shape = 1, ...)

# Paralogistic (package 'VGAM')
paralogisticMode(shape1.a, scale = 1)

# Pareto (package 'VGAM')
paretoMode(location, ...)

# Rayleigh (package 'VGAM')
rayleighMode(scale = 1)

# T (Student)
tMode(df, ncp = 0)

# Uniform
unifMode(min = 0, max = 1)

# Weibull
weibullMode(shape, scale = 1, ...)

## Discrete distributions

# Bernoulli
bernMode(prob)

# Binomial
binomMode(size, prob)

# Geometric
gemMode(...)

# Hypergeometric
hyperMode(m, n, k, ...)

# Negative Binomial
nbinomMode(size, prob, mu)

# Poisson
poisMode(lambda)

## Arguments

- **shape1**: First positive parameter of the Beta and Kumaraswamy distributions. See the package `VGAM` for more details.
- **shape2**: Second positive parameter of the Beta and Kumaraswamy distributions. See the package `VGAM` for more details.
- **shape1.a**: Shape parameter of the Fisk and Paralogistic distributions. See the package `VGAM` for more details.
- **ncp**: Non-centrality parameter of the Beta, Chisquare, and Student distributions.
location
Location parameter of the Cauchy, Koenker, Laplace, Logistic, and Pareto distributions. See the package VGAM for more details.
df
Degree of freedom of the Chisquare and Student distributions.
df1
First degree of freedom of the F distribution.
df2
Second degree of freedom of the F distribution.
loc
Location parameter of the Fréchet, Generalized Extreme Value, Generalized Pareto, Gumbel, and Negative Weibull distributions.
scale
Scale parameter of the Fisk, Fréchet, Gamma, Generalized Extreme Value, Gompertz, Generalized Pareto, Negative Weibull, Paralogistic, Rayleigh, and Weibull distributions. See the packages evd and VGAM for more details.
shape
Shape parameter of the Fréchet, Gamma, Generalized Extreme Value, Gompertz, Generalized Pareto, Negative Weibull, and Weibull distributions. See the packages evd and VGAM for more details.
r
An alternative way to specify the scale of the Gamma distribution.
mean
Mean of the Normal distribution.
alpha
Parameter of the Hyperbolic, Generalised Hyperbolic, Stable, and Normal Inverse Gaussian distributions. See the packages fBasics and stabledist for more details.
beta
Parameter of the Hyperbolic, Generalised Hyperbolic, Stable, and Normal Inverse Gaussian distributions. See the packages fBasics and stabledist for more details.
delta
Parameter of the Hyperbolic, Generalised Hyperbolic, Stable, and Normal Inverse Gaussian distributions. See the packages fBasics and stabledist for more details.
m
Parameter of the Hyperbolic, Generalised Hyperbolic, Normal Inverse Gaussian, and Negative binomial distributions. See the package fBasics for more details.
lambda
Vector of (non-negative) means of the Poisson distribution.
pm
Integer value for the selection of the parameterization of the Hyperbolic and Stable distributions. See the packages fBasics and stabledist for more details.
meanlog
Mean of the Lognormal distribution on the log scale.
sdlog
Standard deviation of the Lognormal distribution on the log scale.
gamma
Scale parameter of the Stable distribution. See the package stabledist for more details.
min
Lower limit of the Uniform distribution. Must be finite.
max
Upper limit of the Uniform distribution. Must be finite.
prob
Probability of success on each trial (between 0 and 1), used in the Bernoulli, Binomial and Negative Binomial distributions.
size
Number of trials (zero or more), used in the Binomial and Negative Binomial distributions.
m
Number of white balls in the urn for the Hypergeometric distribution.
n
Number of black balls in the urn for the Hypergeometric distribution.
k
Number of balls drawn from the urn for the Hypergeometric distribution.
...
Further arguments, which will be ignored.
Value

A numeric value is returned, the (true) mode of the distribution.

Note

Some functions like normMode or cauchyMode, which are related to symmetric distributions, are trivial, but are implemented for exhaustivity.

Author(s)

P. Poncet, except for hypMode written by David Scott, nigMode and stableMode written by Diethelm Wuertz; see packages fBasics and stabledist.

See Also

mlv for the estimation of the mode; the documentation of the related distributions Beta, GammaDist, etc.

Examples

layout(mat = matrix(1:2,1,2))

## Beta distribution
curve(dbeta(x, shape1 = 2, shape2 = 3.1), xlim = c(0,1), ylab = "Beta density")
M <- betaMode(shape1 = 2, shape2 = 3.1)
abline(v = M, col = 2)
mlv("beta", shape1 = 2, shape2 = 3.1)

## Lognormal distribution
curve(dlnorm(x, meanlog = 3, sdlog = 1.1), xlim = c(0,10), ylab = "Lognormal density")
M <- lnormMode(meanlog = 3, sdlog = 1.1)
abline(v = M, col = 2)
mlv("lnorm", meanlog = 3, sdlog = 1.1)

## Poisson distribution
poisMode(lambda = 6)
poisMode(lambda = 6.1)
mlv("poisson", lambda = 6.1)

layout(mat = matrix(1,1,1))
Usage

grenander(x, bw = NULL, k, p, ...)

Arguments

x numeric. Vector of observations.
bw numeric. The bandwidth to be used. Should belong to (0, 1].
k numeric. Parameter 'k' in Grenander's mode estimate.
p numeric. Parameter 'p' in Grenander's mode estimate. If p = Inf, function venter is used.
... further arguments to be passed to link{venter}

Details

The Grenander estimate is defined by

\[
\sum_{j=1}^{n-k} \frac{(x_{j+k}+x_j)}{2(x_{j+k}-x_j)^p} \quad \sum_{j=1}^{n-k} \frac{1}{(x_{j+k}-x_j)^p}
\]

If \( p \) tends to infinity, this estimate tends to the Venter mode estimate; this justifies to call venter if \( p = \text{Inf} \).

The user should either give the bandwidth bw or the argument k, k being taken equal to \( \text{ceiling}(bw*n) - 1 \) if missing.

Value

A numeric value is returned, the mode estimate. If \( p = \text{Inf} \), the Venter mode estimator is returned.

Note

The user should preferentially call grenander through mlv(x, method = "grenander", bw, k, p, ...). This returns an object of class mlv.

Author(s)

D.R. Bickel for the original code,
P. Poncet for the slight modifications introduced.
References


See Also

mlv for general mode estimation; venter for the Venter mode estimate

Examples

```r
# Unimodal distribution
x <- rnorm(1000, mean = 23, sd = 0.5)

## True mode
normMode(mean = 23, sd = 0.5) # (!)

## Parameter 'k'
k <- 5

## Many values of parameter 'p'
p <- seq(0.1, 4, 0.01)

## Estimate of the mode with these parameters
M <- sapply(p, function(pp) grenander(x, p = pp, k = k))

## Distribution obtained
plot(density(M), xlim = c(22.5, 23.5))
```

### Description

This function computes Bickel’s half range mode estimator described in Bickel (2002).

#### Usage

```r
hrm(x, bw = NULL, ...)
```
Arguments

- **x** numeric. Vector of observations
- **bw** numeric. The bandwidth to be used. Should belong to (0, 1]. This gives the fraction of the observations to consider at each step of the iterative algorithm.
- ... further arguments.

Details

The mode estimator is computed by iteratively identifying densest half ranges. A densest half range is an interval whose width equals half the current range, and which contains the maximal number of observations. The subset of observations falling in the selected densest half range is then used to compute a new range, and the procedure is iterated.

Value

A numeric value is returned, the mode estimate.

Note

The user should preferentially call `hrm` through `mlv(x, method = "hrm", bw)`. This returns an object of class `mlv`.

Author(s)

The C and R code are due to Richard Bourgon <bourgon@stat.berkeley.edu>, see package `genefilter`. The algorithm is described in Bickel (2002).

References


See Also

`mlv` for general mode estimation; `hsm` for the half sample mode; `venter` for the Venter mode estimate

Examples

```r
# Unimodal distribution
x <- rgamma(1000, shape = 31.9)

## True mode
gammaMode(shape = 31.9)
```
## Estimate of the mode

```r
hrm(x, bw = 0.4)
M <- mlv(x, method = "hrm", bw = 0.4)
print(M)
plot(M)
```

### Description

This function computes the Robertson-Cryer mode estimator described in Robertson and Cryer (1974), also called half sample mode (if `bw = 1/2`) or fraction sample mode (for some other `bw`) by Bickel (2006).

### Usage

```r
hsm(x, bw = NULL, k, tie.action = "mean", tie.limit = 0.05, ...)
```

### Arguments

- `x` numeric. Vector of observations.
- `bw` numeric or function. The bandwidth to be used. Should belong to (0, 1].
- `k` numeric. See 'Details'.
- `tie.action` character. The action to take if a tie is encountered.
- `tie.limit` numeric. A limit deciding whether or not a warning is given when a tie is encountered.
- `...` further arguments.

### Details

The modal interval, i.e. the shortest interval among intervals containing `k+1` observations, is computed iteratively, until only one value is found, the mode estimate. At each step `i`, one takes $k = \text{ceiling}(bw \times n) - 1$, where `n` is the length of the modal interval computed at step `$i-1$.

If `bw` is of class "function", then $k = \text{ceiling}(bw(n)) - 1$ instead.

### Value

A numeric value is returned, the mode estimate.
Note

The user should preferentially call hsm through mlv(x, method = "hsm", ...). This returns an object of class mlv.

Author(s)

D.R. Bickel for the original code,
P. Poncet for the slight modifications introduced.

References


See Also

mlv for general mode estimation; venter for the Venter mode estimate

Examples

```r
# Unimodal distribution
x <- rweibull(10000, shape = 3, scale = 0.9)

## True mode
weibullMode(shape = 3, scale = 0.9)

## Estimate of the mode
bandwidth <- function(n, alpha) {1/n^alpha}
hsm(x, bw = bandwidth, alpha = 2)
M <- mlv(x, method = "hsm", bw = bandwidth, alpha = 2)
print(M)
plot(M)
```

Description

The Lientz mode estimator is nothing but the value minimizing the empirical Lientz function. A ‘plot’ and a ‘print’ methods are provided.
lientz

Usage

lientz(x,  
  bw = NULL)

  ## S3 method for class 'lientz'
mlv(x,  
  bw = NULL,  
  abc = FALSE,  
  par = shorth(x),  
  optim.method = "BFGS",  
  ...
)

  ## S3 method for class 'lientz'
plot(x,  
  zoom = FALSE,  
  ...
)

  ## S3 method for class 'lientz'
print(x,  
  digits = NULL,  
  ...
)

Arguments

  x numeric (vector of observations) or an object of class "lientz".
  bw numeric. The smoothing bandwidth to be used. Should belong to (0, 1). Parameter 'beta' in Lientz (1970) function.
  abc logical. If FALSE (the default), the Lientz empirical function is minimised using optim.
  par numeric. The initial value used in optim.
  optim.method character. If abc = FALSE, the method used in optim.
  zoom logical. If TRUE, one can zoom on the graph created.
  digits numeric. Number of digits to be printed.
  ... if abc = FALSE, further arguments to be passed to optim, or further arguments to be passed to plot.default.

Details

The Lientz function is the smallest non-negative quantity $S(x, \beta)$, where $\beta = bw$, such that

$$F(x + S(x, \beta)) - F(x - S(x, \beta)) \geq \beta.$$ 

Lientz (1970) provided a way to estimate $S(x, \beta)$; this estimate is what we call the empirical Lientz function.
Value

lientz returns an object of class c("lientz", "function"); this is a function with additional attributes:

- `x`: the x argument
- `bw`: the bw argument
- `call`: the call which produced the result

mlv.lientz returns a numeric value, the mode estimate. If `abc = TRUE`, the x value minimizing the Lientz empirical function is returned. Otherwise, the `optim` method is used to perform minimization, and the attributes: 'value', 'counts', 'convergence' and 'message', coming from the `optim` method, are added to the result.

Note

The user should preferentially call `mlv.lientz` through `mlv(x, method = "lientz", ...)`. This returns an object of class `mlv`.

Author(s)

P. Poncet

References


See Also

`mlv` for general mode estimation; `shorth` for the shorth estimate of the mode

Examples

```r
# Unimodal distribution
x <- rbeta(1000, 23, 4)

## True mode
betaMode(23, 4)

## Lientz object
f <- lientz(x, 0.2)
print(f)
plot(f)

## Estimate of the mode
mlv(f)               # optim(shorth(x), fn = f)
```
Description
This function returns the most frequent value(s) in a given numerical vector.

Usage
mfv(x, ...)

Arguments
x numeric. Vector of observations.
...

Details
Argument x is to come from a discrete distribution. This function uses function `tabulate` of R.

Value
The most frequent value(s) found in x is (are) returned.

Note
The user should preferentially call mfv through mlv(x, method = "mfv") (or mlv(x, method = "discrete").
This returns an object of class mlv.

Author(s)
P. Poncet

References
See Also

*mlv* for general mode estimation; *geomMode*, *poisMode*, etc. for computation of the mode of the usual discrete distributions

Examples

```r
# Unimodal distribution
x <- rbinom(100, size = 10, prob = 0.8)

## True mode
binomMode(size = 10, prob = 0.8)

## Most frequent value
mfv(x)
mlv(x, method = "discrete")

# Bimodal distribution
x <- rpois(100, lambda = 7)

## True mode
poisMode(lambda = 7)

## Most frequent value
mfv(x)
M <- mlv(x, method = "discrete")
print(M)
plot(M)
```

---

**mlv**

*Estimation of the Mode*

Description

*mlv* is a generic function which enables to compute an estimate of the mode of a univariate distribution. Many different estimates (or methods) are provided:

- *mfv*, which returns the most frequent value(s) in a given numerical vector,
- the *Lientz* mode estimator, which is the value minimizing the Lientz function estimate,
- the Chernoff mode estimator, also called *naive* mode estimator, which is defined as the center of the interval of given length containing the most observations,
- the *Venter* mode estimator, including the *shorth*, i.e. the midpoint of the modal interval,
- the *Grenander* mode estimator,
- the half sample mode (*HSM*) and the half range mode (*HRM*), which are iterative versions of the Venter mode estimator,
- *Parzen’s* kernel mode estimator, which is the value maximizing the kernel density estimate,
- the *Tsybakov* mode estimator, based on a gradient-like recursive algorithm,
• the Asselin de Beauville mode estimator, based on an algorithm detecting chains and holes in the sample,
• the Vieu mode estimator.

mlv can also be used to compute the mode of a given distribution, with mlv.character.
A 'plot' and a 'print' methods are provided.

Usage

mlv(x, 
    ...) 

    # Default S3 method:
    mlv(x,
        bw = NULL,
        method,
        na.rm = FALSE,
        boot = FALSE,
        R = 100,
        B = length(x),
        ...)

    # S3 method for class 'factor'
    mlv(x,
        ...) 

    # S3 method for class 'integer'
    mlv(x,
        na.rm = FALSE,
        ...) 

    # S3 method for class 'character'
    mlv(x,
        ...) 

    # S3 method for class 'density'
    mlv(x,
        all = TRUE,
        abc = FALSE,
        ...) 

    # S3 method for class 'mlv'
    plot(x,
        ...) 

    # S3 method for class 'mlv'
    print(x,
        digits = NULL,
        ...)
## S3 method for class 'mlv'
as.numeric(x, ...)

### Arguments

- **x** numeric (vector of observations), or an object of class "factor", "integer", etc. For the function as.numeric, an object of class "mlv".
- **bw** numeric. The bandwidth to be used. This may have different meanings regarding the method used.
- **method** character. One of the methods available for computing the mode estimate. See 'Details'.
- **na.rm** logical. Should missing values be removed?
- **boot** logical. Should bootstrap resampling be done?
- **R** numeric. If boot = TRUE, the number of bootstrap resampling rounds to use.
- **B** numeric. If boot = TRUE, the size of the bootstrap samples drawn from x. Default is to use a sample which is the same size as data. For large data sets, this may be slow and unnecessary.
- **all** logical.
- **abc** logical. If FALSE (the default), the estimate of the density function is maximised using optim.
- **digits** numeric. Number of digits to be printed.
- **...** Further arguments to be passed to the function called for computation. This function is related to the method argument.

### Details

For the function mlv.default, available methods are "mfv", "lientz", "naive", "venter", "grenander", "hsm", "hrm", "parzen", "tsybakov", and "asselin". See the description above and the associated links.

If x is of class "factor" or "integer", the most frequent value found in x is returned.
If x is of class "character", x should be one of "beta", "cauchy", "gev", etc. i.e. a character for which a function 'x'Mode exists (for instance betaMode, cauchyMode, etc.). See distribMode for the available functions. The mode of the corresponding distribution is returned.
If x is of class "density", the value where the density is maximised is returned.
For the S3 function mlv.lientz, see lientz for more details.

### Value

mlv returns an object of class "mlv".
An object of class "mlv" is a list containing at least the following components:

- **M** the value of the mode
- **skewness** Bickel's measure of skewness
- **x** the argument x
method  the argument method
bw     the bandwidth

boot   the argument boot
boot.M  if boot = TRUE, the resampled values of the mode

call   the call which produced the result

Author(s)

P. Poncet

References

See the references on mode estimation on the modeest-package's page.

See Also

mfv, Lientz, naive, venter, grenander, hrm, hsm, parzen, tsybakov, skewness

Examples

# Unimodal distribution
x <- rbeta(1000, 23, 4)

## True mode
betaMode(23, 4)

# or
mlv("beta", 23, 4)

## Estimate of the mode
mlv(x, method = "lientz", bw = 0.2)
mlv(x, method = "naive", bw = 1/3)
mlv(x, method = "venter", type = "shorth")
mlv(x, method = "grenander", p = 4)
mlv(x, method = "hrm", bw = 0.3)
mlv(x, method = "hsm")
mlv(x, method = "parzen", kernel = "gaussian")
mlv(x, method = "tsybakov", kernel = "gaussian")
mlv(x, method = "asselin", bw = 2/3)
mlv(x, method = "vieu")

## Bootstrap
M <- mlv(x, method = "kernel", boot = TRUE, R = 150)
print(M)
plot(M)
print(mean(M["boot.M"]))
Description

This package intends to provide estimators of the mode of univariate unimodal (and sometimes multimodal) data and values of the modes of usual probability distributions.

For a complete list of functions, use `library(help = "modeest")` or `help.start()`.

Details

- **Package:** modeest
- **Type:** Package
- **Version:** 2.1
- **Date:** 2012-10-15
- **License:** GPL version 2 or newer

Author(s)

P. Poncet

References


**See Also**

`mlv` for general mode estimation

---

```
naive

The Chernoff Mode Estimator
```

**Description**

This estimator, also called the *naive* mode estimator, is defined as the center of the interval of given length containing the most observations. It is identical to Parzen’s kernel mode estimator, when the kernel is chosen to be the uniform kernel.
Usage

naive(x, 
   bw = 1/2)

Arguments

x    numeric. Vector of observations.

bw   numeric. The smoothing bandwidth to be used. Should belong to (0, 1). See below.

Value

A numeric vector is returned, the mode estimate, which is the center of the interval of length 2*bw containing the most observations.

Note

The user should preferentially call naive through mlv(x, method = "naive", bw). This returns an object of class mlv.

Author(s)

P. Poncet

References


See Also

mlv for general mode estimation; parzen for Parzen’s kernel mode estimation

Examples

# Unimodal distribution
x <- rf(10000, df1 = 40, df2 = 30)

## True mode
fMode(df1 = 40, df2 = 30)

## Estimate of the mode
mean(naive(x, bw = 1/4))
M <- mlv(x, method = "naive", bw = 1/4)
print(M)
plot(M, xlim = c(0,2))
Parzen's Kernel Mode Estimator

Description

Parzen's kernel mode estimator is the value maximizing the kernel density estimate.

Usage

\[
\text{parzen}(x, \\
\quad \text{bw} = \text{NULL}, \\
\quad \text{kernel} = \text{"gaussian"}, \\
\quad \text{abc} = \text{FALSE}, \\
\quad \text{par} = \text{shorth}(x), \\
\quad \text{optim.method} = \text{"BFGS"}, \\
\quad \ldots)
\]

Arguments

- **x**: numeric. Vector of observations.
- **bw**: numeric. The smoothing bandwidth to be used.
- **kernel**: character. The kernel to be used. Available kernels are "biweight", "cosine", "eddy", "epanechnikov", "gaussian", "optcosine", "rectangular", "triangular", "uniform". See `density.default` for more details on some of these kernels.
- **abc**: logical. If FALSE (the default), the kernel density estimate is maximised using `optim`.
- **par**: numeric. The initial value used in `optim`.
- **optim.method**: character. If abc = FALSE, the method used in `optim`.
- **...**: if abc = FALSE, further arguments to be passed to `optim`.

Details

If kernel = "uniform", the naive mode estimate is returned.

Value

`parzen` returns a numeric value, the mode estimate. If abc = TRUE, the x value maximizing the density estimate is returned. Otherwise, the `optim` method is used to perform maximization, and the attributes: 'value', 'counts', 'convergence' and 'message', coming from the `optim` method, are added to the result.

Note

The user should preferentially call `parzen` through `mlv(x, method = "kernel", ...) or `mlv(x, method = "parzen", ...)`. This returns an object of class `mlv`.

Presently, `parzen` is quite slow.
skewness

Author(s)

P. Poncet

References


See Also

mlv, naive

Examples

```r
# Unimodal distribution
x <- rlnorm(10000, meanlog = 3.4, sdlog = 0.2)

## True mode
lnormMode(meanlog = 3.4, sdlog = 0.2)

## Estimate of the mode
M <- mlv(x, method = "kernel", kernel = "gaussian", bw = 0.3, par = shorth(x))
print(M)
plot(M)
```

<table>
<thead>
<tr>
<th>skewness</th>
<th>Skewness</th>
</tr>
</thead>
</table>

Description

The `skewness` function from package fBasics is completed in order to implement Bickel’s measure of skewness, based on the mode of the distribution considered.
Usage

```r
skewness(x, 
    ...)

## Default S3 method:
skewness(x, 
    na.rm = FALSE, 
    method = c("moment", "fisher", "bickel"), 
    M = shorth(x), 
    ...)
```

Arguments

- `x` numeric. Vector of observations.
- `na.rm` logical. Should missing values be removed?
- `method` character. Specifies the method of computation. These are either "moment", "fisher" or "bickel". The "moment" method is based on the definition of skewness for distributions; this form should be used when resampling (bootstrap or jackknife). The "fisher" method corresponds to the usual "unbiased" definition of sample variance, although in the case of skewness exact unbiasedness is not possible.
- `M` numeric. (An estimate of) the mode of the observations `x`. Default value is `shorth(x)`.
- `...` arguments to be passed.

Value

`skewness` returns a numeric value. An attribute which reports the method used is added.

Author(s)

Diethelm Wuertz and other authors for the original `skewness` function from package `fBasics`; P. Poncet for the slight modification introduced.

References


See Also

package `fBasics`; `mlv` for general mode estimation; `shorth` for the shorth estimate of the mode
Examples

```r
## Skewness = 0
x <- rnorm(1000)
skewness(x, method = "bickel", M = shorth(x))

## Skewness > 0 (left skewed case)
x <- rbeta(1000, 2, 5)
skewness(x, method = "bickel", M = betaMode(2, 5))

## Skewness < 0 (right skewed case)
x <- rbeta(1000, 7, 2)
skewness(x, method = "bickel", M = hsm(x, bw = 1/3))
```

Description

This mode estimator is based on a gradient-like recursive algorithm. It includes the Mizoguchi-Shimura (1976) mode estimator, based on the window training procedure.

Usage

```r
tsybakov(x, 
  bw = NULL, 
  a, 
  alpha = 0.9, 
  kernel = "triangular", 
  dmp = TRUE, 
  par = shorth(x))
```

Arguments

- `x` numeric. Vector of observations.
- `bw` numeric. Vector of length length(x) giving the sequence of smoothing bandwidths to be used.
- `a` numeric. Vector of length length(x) used in the gradient algorithm.
- `alpha` numeric. An alternative way of specifying a. See ‘Details’.
- `kernel` character. The kernel to be used. Available kernels are "biweight", "cosine", "eddy", "epanechnikov", "gaussian", "optcosine", "rectangular", "triangular", "uniform". See `density.default` for more details on some of these kernels.
- `dmp` logical. If TRUE, Djeddour et al. version of the estimate is used.
- `par` numeric. Initial value in the gradient algorithm. Default value is `shorth(x)`.
Details

If bw is missing, then bw = (1:length(x))^(−1/7), which is the default value advised by Djeddour et al (2003). If a is missing, then a = (1:length(x))^(−alpha) (with alpha = 0.9 if alpha is missing), which is the default value advised by Djeddour et al (2003).

Value

A numeric value is returned, the mode estimate.

Warning

The Tsybakov mode estimate as it is presently computed does not work very well. The reasons of this inefficiency are under investigation.

Note

The user should preferentially call tsybakov through mlv(x, method = "tsybakov", ...). This returns an object of class mlv.

Author(s)
P. Poncet

References


See Also

mlv for general mode estimation

Examples

```r
x <- rbeta(1000, shape1 = 2, shape2 = 5)
```

```r
## True mode:
betaMode(shape1 = 2, shape2 = 5)
```

```r
## Estimation:
tsybakov(x, kernel = "triangular")
tsybakov(x, kernel = "gaussian", alpha = 0.99)
```
The Venter / Dalenius / LMS Mode Estimator

**Description**

This function computes the Venter mode estimator, also called the Dalenius, or LMS (Least Median Square) mode estimator.

**Usage**

```r
venter(x,
    bw = NULL,
    k,
    iter = 1,
    type = 1,
    tie.action = "mean",
    tie.limit = 0.05)

shorth(x,
    ...
```

**Arguments**

- `x` numeric. Vector of observations.
- `bw` numeric. The bandwidth to be used. Should belong to (0, 1]. See 'Details'.
- `k` numeric. See 'Details'.
- `iter` numeric. Number of iterations.
- `type` numeric or character. The type of Venter estimate to be computed. See 'Details'.
- `tie.action` character. The action to take if a tie is encountered.
- `tie.limit` numeric. A limit deciding whether or not a warning is given when a tie is encountered.
- `...` Further arguments.

**Details**

The modal interval, i.e. the shortest interval among intervals containing k+1 observations, is first computed. The user should either give the bandwidth `bw` or the argument `k`, `k` being taken equal to `ceiling(bw*n) - 1` if missing.

If `type = 1`, the midpoint of the modal interval is returned. If `type = 2`, the `floor(((k+1)/2)`th element of the modal interval is returned. If `type = 3` or `type = "dalenius"`, the median of the modal interval is returned. If `type = 4` or `type = "shorth"`, the mean of the modal interval is returned. If `type = 5` or `type = "ekblom"`, Ekblom’s $L_{-\infty}$ estimate is returned, see Ekblom (1972). If `type = 6` or `type = "hsm"`, the half sample mode (hsm) is computed, see `hsm`. 

---

```r
M <- mly(x, method = "tsybakov", kernel = "gaussian", alpha = 0.99)
print(M)
plot(M)
```
Value
A numeric value is returned, the mode estimate.

Note
The user should preferentially call `venter` through `mlv(x, method = "venter", ...). This returns an object of class `mlv`.

Author(s)
P. Poncet

References

See Also
- `mlv` for general mode estimation, `hsm` for the half sample mode

Examples

```r
library(evd)

# Unimodal distribution
x <- rgev(1000, loc = 23, scale = 1.5, shape = 0)

## True mode
gevMode(loc = 23, scale = 1.5, shape = 0)

## Estimate of the mode
venter(x, bw = 1/3)
M <- mlv(x, method = "venter", bw = 1/3)
print(M)
plot(M, xlim = c(20, 30))
```
vieu

Vieu’s Mode Estimator

Description

Vieu’s mode estimator is the value at which the kernel density derivative estimate is null.

Usage

```r
vieu(x,
    bw = NULL,
    kernel = "gaussian",
    abc = FALSE,
    ...)```

Arguments

- **x**: numeric. Vector of observations.
- **bw**: numeric. The smoothing bandwidth to be used.
- **kernel**: character. The kernel to be used. Available kernels are "biweight", "cosine", "eddy", "epanechnikov", "gaussian", "optcosine", "rectangular", "triangular", "uniform". See `density.default` for more details on some of these kernels.
- **abc**: logical. If FALSE (the default), the root of the density derivative estimate is searched with `uniroot`.
- **...**: if abc = FALSE, further arguments to be passed to `uniroot`.

Value

`vieu` returns a numeric value, the mode estimate. If abc = TRUE, the x value at which the density derivative estimate is null is returned. Otherwise, the `uniroot` method is used.

Note

The user should preferentially call `vieu` through `mlv(x, method = "vieu", ...)`. This returns an object of class `mlv`.

Presently, `vieu` is quite slow.

Author(s)

P. Poncet

References

See Also

mlv, parzen

Examples

# Unimodal distribution
x <- rlnorm(10000, meanlog = 3.4, sdlog = 0.2)

## True mode
lnormMode(meanlog = 3.4, sdlog = 0.2)

## Estimate of the mode
M <- mlv(x, method = "vieu", kernel = "gaussian")
print(M)
plot(M)
Index

*Topic **distribution**
  distribMode, 3
  modeest, 20

*Topic **nonparametric**
  asselin, 2
  grenander, 7
  hrm, 9
  hsm, 11
  mlv, 16
  modeest, 20
  tsybakov, 27

*Topic **robust**
  hrm, 9
  hsm, 11
  mlv, 16
  modeest, 20

*Topic **univar**
  asselin, 2
  distribMode, 3
  grenander, 7
  hrm, 9
  hsm, 11
  lientz, 12
  mfv, 15
  mlv, 16
  modeest, 20
  naive, 22
  parzen, 24
  skewness, 25
  tsybakov, 27
  venter, 29
  vieu, 31

  as.numeric.mlV (mlV), 16
  Asselin, 17
  Asselin (asselin), 2
  asselin, 2

  bernMode (distribMode), 3
  Beta, 7

  betaMode (distribMode), 3
  Bickel (hrm), 9
  bickel (hrm), 9
  binomMode (distribMode), 3
  cauchyMode (distribMode), 3
  chisqMode (distribMode), 3
  class, 18
  density.default, 24, 27, 31
  discrete (mfv), 15
  distribMode, 3, 18
  expMode (distribMode), 3
  fiskMode (distribMode), 3
  fMode (distribMode), 3
  frechetMode (distribMode), 3
  GammaDist, 7
  gammaMode (distribMode), 3
  geomMode, 16
  geomMode (distribMode), 3
  gevMode (distribMode), 3
  ghMode (distribMode), 3
  gompertzMode (distribMode), 3
  gpdMode (distribMode), 3
  Grenander, 16
  Grenander (grenander), 7
  grenander, 7, 19
  gumbelMode (distribMode), 3
  half.range.mode (hrm), 9
  HRM, 16
  HRM (hrm), 9
  hrm, 9, 19
  HSM, 16
  HSM (hsm), 11
  hsm, 10, 11, 19, 29, 30
  hyperMode (distribMode), 3
  hypMode (distribMode), 3
koenkerMode (distribMode), 3
tsybakov (tsybakov), 27

kumarMode (distribMode), 3
tsybakov, 19, 27

laplaceMode (distribMode), 3
Lientz, 16, 18, 19
Lientz (lientz), 12
distribmode, 3
lms (venter), 29
lnormMode (distribMode), 3
logisMode (distribMode), 3

mfv, 15, 16, 19
mlv, 2, 7–10, 12, 14–16, 16, 22–26, 28, 30–32
mlv.lientz (lientz), 12
modeest, 20
modeest-package (modeest), 20

naive, 16, 19, 22, 24, 25
nbinomMode (distribMode), 3
nigMode (distribMode), 3
normMode (distribMode), 3

optim, 13, 14, 18, 24

paralogisticMode (distribMode), 3
pararModel (distribMode), 3
Parzen, 16
Parzen (parzen), 24
parzen, 19, 23, 24, 32
plot.default, 13
plot.lientz (lientz), 12
plot.mlvm (mlv), 16
poisMode, 16
poisMode (distribMode), 3
print.lientz (lientz), 12
print.mlvm (mlv), 16

quantile, 2

rayleighMode (distribMode), 3
rweibullMode (distribMode), 3

shorth, 14, 16, 26, 27
shorth (venter), 29
skewness, 18, 19, 25
stableMode (distribMode), 3

tabulate, 15
tMode (distribMode), 3
Tsybakov, 16

unifMode (distribMode), 3
uniroot, 31

Venter, 8, 16
Venter (venter), 29
venter, 8–10, 12, 19, 29
Vieu, 17
Vieu (vieu), 31
vieu, 31

weibullMode (distribMode), 3