Package 'logKDE'

April 25, 2018

Type Package

Title Computing Log-Transformed Kernel Density Estimates for Positive Data

Version 0.1.0

Date 2018-04-25

Author Hien D. Nguyen, Andrew T. Jones, and Geoffrey J. McLachlan

Maintainer Andrew Thomas Jones <andrewthomasjones@gmail.com>

Description Computes log-transformed kernel density estimates for positive data using a variety of kernels. It follows the methods
described in Jones, Nguyen and McLachlan (2018) <arXiv:1804.08365>.

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports Rcpp, pracma

SystemRequirements C++11

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-25 10:21:08 UTC

R topics documented:

bw.logCV *Optimal CV BW estimation for strictly positive distributions.*

Description

Computes least squares cross-validation (CV) bandwidth (BW) for log domain KDE.

Usage

```
bw.logCV(x, grid = 21, NB = 512)
```

Arguments

x	numeric vector of the data. Must be strictly positive, will be log transformed during estimation.
grid	number of points used for BW selection CV grid.
NB	number of points at which to estimate the KDE at during the CV loop.

Value

bw the optimal least squares CV bandwidth.

Examples

```
bw.logCV(rchisq(100,10), grid=21, NB=512)
```

bw.logG *Bandwidth estimation for strictly positive distributions.*

Description

Computes bandwidth for log domain KDE using the Silverman rule.

Usage

```
bw.logG(x)
```

Arguments

x	numeric vector of the data. Must be strictly positive, will be log transformed during estimation.

Value

bw the optimal bandwidth.

Examples

```
bw.logG(rchisq(100,10))
```

`logdensity` *Kernel Density Estimates of strictly positive distributions.*

Description

The function `logdensity` computes kernel density estimates (KDE) of strictly positive distributions by performing the KDE in the log domain and then transforming the result back again. The syntax and function structure is largely borrowed from the function `density` in package stats.

Usage

```
logdensity(x, bw = "nrd0", adjust = 1, kernel = "gaussian",
  weights = NULL, n = 512, from, to, cut = 3, na.rm = FALSE)
```

Arguments

`x`	the data from which the estimate is to be computed.
`bw`	the smoothing bandwidth to be used. Can also be can also be a character string giving a rule to choose the bandwidth. Like `density` defaults to "nrd0". All options in `help(bw.nrd)` are available as well as `"bw.logCV"` and `"bw.logG"`.
`adjust`	the bandwidth used is actually `adjust*bw`.
`kernel`	a character string giving the smoothing kernel to be used. Choose from "gaussian", "epanechnikov", "triangular", "uniform", "laplace" and "logistic". Default value is "gaussian".
`weights`	numeric vector of non-negative observation weights of the same length as `x`.
`n`	the number of equally spaced points at which the density is to be estimated. Note that these are equally spaced in the original domain.
`from, to`	the left and right-most points of the grid at which the density is to be estimated; the defaults are cut * bw outside of range(x).
`cut`	by default, the values of from and to are cut bandwidths beyond the extremes of the data
`na.rm`	logical; if TRUE, missing values are removed from `x`. If FALSE any missing values cause an error.

Value

An object with class "density". See `help(density)` for details.

See Also

`density`, `plot.density`, `logdensity_fft`, `bw.nrd`, `bw.logCV`, `bw.logG`.

Examples

```
logdensity(abs(rnorm(100)), from =.1, to=2, kernel='triangular')
```

`logdensity_fft` *Kernel Density Estimates of strictly positive distributions using FFT.*

Description

The function `logdensity_fft` computes kernel density estimates (KDE) of strictly positive distributions by performing the KDE via fast fourier transform utilizing the `fft` function. The syntax and function structure is largely borrowed from the function `density` in package stats.

Usage

```
logdensity_fft(x, bw = "nrd0", adjust = 1, kernel = "gaussian",
  weights = NULL, n = 512, from, to, cut = log(3), na.rm = FALSE)
```

Arguments

`x`	the data from which the estimate is to be computed.
`bw`	the smoothing bandwidth to be used. Can also be can also be a character string giving a rule to choose the bandwidth. Like `density` defaults to "nrd0". All options in `help(bw.nrd)` are available as well as `"bw.logCV"` and `"bw.logG"`.
`adjust`	the bandwidth used is actually `adjust*bw`.
`kernel`	a character string giving the smoothing kernel to be used. Choose from "gaussian", "epanechnikov", "triangular", "uniform", "laplace" and "logistic". Default value is "gaussian".
`weights`	numeric vector of non-negative observation weights of the same length as `x`.
`n`	the number of equally spaced points at which the density is to be estimated. Note that these are equally spaced in the log domain for `logdensity_fft`, and thus on a log scale when transformed back to the original domain.
`from, to`	the left and right-most points of the grid at which the density is to be estimated; the defaults are cut * bw outside of range(x).
`cut`	by default, the values of from and to are cut bandwidths beyond the extremes of the data
`na.rm`	logical; if TRUE, missing values are removed from `x`. If FALSE any missing values cause an error.

Value

An object with class "density". See `help(density)` for details.

See Also

density, plot.density, logdensity, bw.nrd, bw.logCV, bw.logG.

Examples

```
logdensity_fft(abs(rnorm(100)), from =0.01, to= 2.5, kernel = 'logistic')
```

Index