Package: kedd (via r-universe)

November 20, 2026

Type Package

Title Kernel Estimator and Bandwidth Selection for Density and Its Derivatives

Version 2.0.2

Date 2026-11-20

Maintainer Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>

Description Kernel estimation of a univariate probability density and of its derivatives of any order r, with data-driven bandwidth selection. Provides the kernel estimator of the derivative of order r, the kernels, their derivatives and convolutions, and nine bandwidth selectors: the asymptotic MISE rule (AMISE), and unbiased (UCV), biased (BCV1, BCV2), complete (CCV), modified (MCV), trimmed (TCV), smoothed (SCV), partitioned (PCV) and maximum-likelihood (MLCV) cross-validation. Supports ten named kernels (Gaussian, Epanechnikov, uniform, triangular, biweight, triweight, tricube, cosine, Silverman, logistic) and the power family Pow(s,t), c(1-|x|^s)^t on [-1,1]. Results are S4 objects with 'ggplot2' graphics; the O(n^2) criteria are computed in C++ ('Rcpp') with 'OpenMP' parallelism. Includes five example samples from Marron and Wand (1992) normal mixtures, a vignette and demos. Companion to Guidoum (2020) <doi:10.48550/arXiv.2012.06102>.

License GPL (>= 2)

URL https://github.com/ac-guidoum/kedd, https://arxiv.org/abs/2012.06102

BugReports https://github.com/ac-guidoum/kedd/issues

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), methods, stats, graphics, ggplot2 (>= 3.4.0), tools, utils

LinkingTo Rcpp

Suggests knitr, rmarkdown, testthat, covr

VignetteBuilder knitr

NeedsCompilation yes

LazyData true

Encoding UTF-8

Config/roxygen2/version 8.1.1

Author Arsalane Chouaib Guidoum [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3781-2160>), Amel Henider [aut],
Iago Giné-Vázquez [ctb] (ORCID:
<https://orcid.org/0000-0002-6725-2638>)

Repository https://cran.r-universe.dev

Date/Publication 2026-11-20 17:50:14 UTC

RemoteUrl https://github.com/cran/kedd

RemoteRef HEAD

RemoteSha 9122a8b6795dbb3ab816558f5d9c292a4b6f08ce

Contents

kedd-package . . . 3
bandwidth-class . . . 5
dkde . . . 7
h.amise . . . 9
h.bcv . . . 11
h.ccv . . . 13
h.mcv . . . 15
h.mlcv . . . 17
h.pcv . . . 19
h.scv . . . 22
h.tcv . . . 24
h.ucv . . . 26
kedd-classes . . . 28
kedd-data . . . 29
kernel.conv . . . 30
kernel.fun . . . 32
kernel.pow . . . 34

Index **36**

kedd-package *Kernel Estimator and Bandwidth Selection for Density and Its Derivatives*

Description

Kernel smoothing of a univariate density f and of its r-th derivative $f^{(r)}$:

$$\hat{f}_h^{(r)}(x) = \frac{1}{nh^{r+1}} \sum_{i=1}^{n} K^{(r)}\left(\frac{x - X_i}{h}\right),$$

with S4 classes, **ggplot2** graphics and Rcpp/OpenMP data-driven bandwidth selectors.

Details

Kernels. Every function accepts `kernel` as one of `"gaussian"`, `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a member `"pow(s,t)"` of the power family $K(x) = c_{s,t}(1 - |x|^s)^t \mathbf{1}_{|x|\leq 1}$ (see `kernel.pow`).

Maximum derivative order r for each kernel (see the section below).

Bandwidth selectors (S4 class `bandwidth`):

`h.amise`	asymptotic MISE (normal reference)
`h.ucv`	unbiased (least-squares) cross-validation
`h.bcv`	biased cross-validation (BCV1, BCV2)
`h.ccv`	complete cross-validation
`h.mcv`	modified cross-validation
`h.tcv`	trimmed cross-validation
`h.scv`	smoothed cross-validation (Gaussian)
`h.pcv`	partitioned cross-validation
`h.mlcv`	maximum-likelihood cross-validation

Estimation and kernels: `dkde` (S4 class `dkde`), `kernel.fun` ($K^{(r)}$, class `kfun`), `kernel.conv` ($K^{(r)} * K^{(r)}$, class `kconv`). All classes have `show()` and **ggplot2**-based `plot()` methods; `lines()` on a `bandwidth` object returns a layer to add with `+`.

Parallelism: the $O(n^2)$ criteria run in C++ with OpenMP. The `nthreads` argument sets the threads of each call: `0` (default) = all processors, `k > 0` = `k` threads, capped by `OMP_THREAD_LIMIT`; it does not change results nor later calls. `Cpp_omp_threads(nthreads)` reports the threads actually used.

Data: five samples of size 200 from Marron and Wand (1992) normal mixtures, `kurtotic`, `outlier`, `bimodal`, `trimodal` and `claw` (see `kedd-data`).

Source code and bug reports: https://github.com/ac-guidoum/kedd and https://github.com/ac-guidoum/kedd/issues.

Documentation: a worked introduction is given in `vignette("kedd", package = "kedd")`, and demonstrations in `demo("kedd")`, `demo("kernels")` and `demo("bandwidths")`.

Maximum derivative order

The regularity q of a kernel is the largest m such that the pointwise derivative $K^{(m)}$ is the derivative of K in the sense of distributions ($K^{(m-1)}$ absolutely continuous). For $m > q$, $K^{(m)}$ misses the Dirac masses at the jumps of $K^{(q)}$, and a criterion that evaluates it at the pairwise differences $X_i - X_j$ no longer estimates its target. Each criterion therefore restricts r:

`dkde`, `h.amise`	$r \le q$
`h.ucv`, `h.tcv` ($K^{(2r)}$)	$2r \le q$
`h.mcv` ($K^{(2r+2)}$)	$2r+2 \le q$
`h.bcv`, BCV1 (convolution of $K^{(r+2)}$)	$r+2 \le q$
`h.bcv`, BCV2, and `h.ccv` ($K^{(2r+4)}$)	$2r+4 \le q$

In addition $K^{(r)}$ ($K^{(r+2)}$ for BCV1) must be square integrable, and the selectors are capped at $r = 8$ for the logistic kernel and $r = 6$ for `pow(s,t)`, the derivative orders implemented being 20 and 16. Above the limit the selectors return `h = NA` with a warning, and `dkde` gives a warning. "any" means no limit and "–" means not available, even at $r = 0$.

kernel	q	**dkde**	**AMISE**	**UCV, TCV**	**MCV**	**BCV1**	**BCV2, CCV**
`gaussian`	∞	any	any	any	any	any	any
`logistic`	∞	20	8	8	8	8	8
`silverman`	3	3	(*)	1	(*)	(*)	(*)
`triweight`	3	3	3	1	0	1	–
`tricube`	3	3	3	1	0	1	–
`biweight`	2	2	2	1	0	0	–
`epanechnikov`	1	1	1	0	–	–	–
`triangular`	1	1	1	0	–	–	–
`cosine`	1	1	1	0	–	–	–
`uniform`	0	0	0	0	–	–	–

(*) The Silverman kernel is of order 4 ($\mu_2(K) = 0$): `h.amise` returns the value of the Gaussian kernel with a warning, and the search interval of the selectors uses it too. BCV1, BCV2, MCV and CCV, built on the bias expansion of a second-order kernel ($\mu_2(K)$, $\mu_4(K)$), are not defined for it: with $\mu_2(K) = 0$ the bias term of BCV vanishes and the criterion decreases up to the end of the search interval.

For `pow(s,t)`, the factor $(1 - |x|)^t$ at $|x| = 1$ gives $q = \lceil t + 1 \rceil - 1$ ($q = t$ for an integer t), and $|x|^s$ at 0 gives $q = \lceil s + 1 \rceil - 1$ unless s is an even integer; q is the smaller of the two. $K^{(m)}$ is square integrable when $m < t + 1/2$ and, unless s is an even integer, $m < s + 1/2$. For example `pow(2,4)`: $q = 4$ and the limits 4, 4, 2, 1, 2, 0; `pow(2,8)`: 8, 6, 4, 3, 6, 2; `pow(1.5,2.5)`: $q = 2$ and 1, 1, 1, 0, –, –. `h.scv` (Gaussian kernel) has no limit, `h.mlcv` is for $r = 0$ only, and `h.pcv` has the limits of the score computed in each subsample.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz> (author, maintainer), Amel Henider <amel.henider@univ-tam.dz (author), Iago Giné-Vázquez <iago.gin-vaz@protonmail.com> (contributor).

References

Guidoum, A. C. (2020). Kernel estimator and bandwidth selection for density and its derivatives: the kedd package. *arXiv:2012.06102*.

Marron, J. S. and Wand, M. P. (1992). Exact mean integrated squared error. *The Annals of Statistics*, **20**, 712–736.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

dkde, kernel.fun, kernel.conv, kernel.pow, h.ucv, h.scv, h.pcv, bandwidth, kedd-data, vignette("kedd"), citation("kedd").

Examples

```
x <- rnorm(150)
bw <- h.ucv(x, kernel = "biweight")
bw
plot(dkde(data = x, h = bw@h, kernel = "biweight"))

## logistic and power kernels
h.bcv(x, whichbcv = 2, kernel = "logistic")
h.bcv(x, whichbcv = 2, kernel = kernel.pow(2, 4))
```

---

bandwidth-class *Class* "bandwidth" — *Result of Bandwidth Selection*

Description

An S4 class returned by all bandwidth selector functions in **kedd** (h.ucv, h.bcv, h.ccv, h.mcv, h.tcv, h.scv, h.amise, h.mlcv, h.pcv). It stores the selected bandwidth together with the data, kernel, derivative order, and optimisation details.

Objects from the Class

Objects are created by calls to the bandwidth selector functions, or directly with new("bandwidth", ...).

Slots

data: Object of class "numeric". The (cleaned) data vector.

kernel: Object of class "character". Name of the kernel function used (one of "gaussian", "epanechnikov", "triangular", "biweight", "uniform", "triweight", "tricube", "cosine", "silverman", "logistic" or "pow(s,t)").

deriv.order: Object of class "integer". Derivative order $r \geq 0$.

h: Object of class "numeric". The selected bandwidth.

`min.obj`: Object of class `"numeric"`. The minimum value of the cross-validation (or other) criterion at `h`. `NA` for `h.amise`.

`hos`: Object of class `"numeric"`. The over-smoothing bandwidth h_{os}; the default search interval is $[0.01\, h_{os},\, 2\, h_{os}]$.

`method`: Object of class `"character"`. The selector used: one of `"UCV"`, `"BCV1"`, `"BCV2"`, `"CCV"`, `"MCV"`, `"TCV"`, `"SCV"`, `"AMISE"`, `"MLCV"`, `"PCV"`.

`criterion`: Object of class `"character"`. For `"PCV"`, the criterion computed in each subsample; empty otherwise.

`groups`: Object of class `"integer"`. For `"PCV"`, the subsample of each observation, for each random partition one after the other; empty otherwise.

Methods

`show(object)` Prints a compact summary: method (and, for PCV, the criterion and the number of subsamples), kernel, derivative order, selected bandwidth, minimum criterion.

`plot(x, h = NULL, type = "o", n.points = 100L, log_scale_x = FALSE, show.min = TRUE, show.hos = FALSE, ...)` Draws the cross-validation curve over a grid of bandwidths with **ggplot2** and marks the selected `h`. Returns invisibly a list with `h`, `cv` and the ggplot object `plot`. Not available for `"AMISE"` (closed-form rule, no criterion).

`lines(x, n.points = 100L, color, linewidth, linetype, ...)` Returns a `ggplot2::geom_line()` layer with the curve of `x`, to be added to a plot: `plot(bw1)$plot + lines(bw2)`.

`x$name` Compatibility with **kedd** 1.x, where the selectors returned a list: `bw$h` is `bw@h`, and `$` also accepts the slot names, `x` (the data), `n` and the 1.x names `min.ucv`, `min.bcv`, ... (the slot `min.obj`). Unknown names give `NULL`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

See Also

`h.ucv`, `h.bcv`, `h.ccv`, `h.mcv`, `h.tcv`, `h.scv`, `h.pcv`, `h.amise`, `h.mlcv`, `dkde`.

Examples

```
x <- rnorm(100)
bw <- h.ucv(x, deriv.order = 0)
show(bw)
p <- plot(bw)

## overlay a second criterion (ggplot2 layer)
p$plot + lines(h.bcv(x), color = "darkred")
```

dkde *Kernel Density Derivative Estimator*

Description

Computes the r'th derivative of a kernel density estimator at a grid of evaluation points. For `deriv.order = 0` this reduces to an ordinary kernel density estimate.

Usage

```
dkde(x = NULL, data, h, deriv.order = 0L,
     kernel = "gaussian", nthreads = 0L)
```

Arguments

`x`	numeric vector of evaluation points. If `NULL` (default), a regular grid of 512 points spanning $[\min(data) - 3h,\ \max(data) + 3h]$ is used.
`data`	numeric vector of observations.
`h`	positive bandwidth scalar. If missing, the UCV bandwidth `h.ucv(data, deriv.order, kernel)` is used.
`deriv.order`	non-negative integer: derivative order r. The estimate is the r-th derivative of the density estimate when $r \leq q$, where q is the regularity of the kernel: any for Gaussian; logistic 20; triweight, tricube and Silverman 3; biweight 2; Epanechnikov, triangular and cosine 1; uniform 0; `pow(s,t)` q, at most 16. Above it a warning is given. When `h` is missing it is chosen by `h.ucv`, whose limits are lower (see `kedd-package`, section 'Maximum derivative order').
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`nthreads`	number of OpenMP threads for this call: `0` (default) = all processors, `k > 0` = `k` threads, both capped by the environment variable `OMP_THREAD_LIMIT`. The result does not depend on it, and it never changes the threads used by later calls.

Details

The r'th derivative kernel density estimator is:

$$\hat{f}_h^{(r)}(x) = \frac{1}{n\, h^{r+1}} \sum_{i=1}^{n} K^{(r)}\left(\frac{x - X_i}{h}\right)$$

where $K^{(r)}(x) = d^r K(x)/dx^r$ (see `kernel.fun`).

Theoretical properties (Bhattacharya 1967; Schuster 1969):

- **Bias:** $E[\hat{f}_h^{(r)}(x)] - f^{(r)}(x) = \frac{h^2}{2}\mu_2(K)\, f^{(r+2)}(x) + o(h^2)$

- **Variance:** $Var[\hat{f}_h^{(r)}(x)] = \frac{f(x)\,R(K^{(r)})}{n\,h^{2r+1}} + o(1/(n\,h^{2r+1}))$

The AMISE-optimal bandwidth is of order $n^{-1/(2r+5)}$, so higher derivatives require larger bandwidths.

The **AMISE** is:

$$AMISE(\hat{f}_h^{(r)}) = \frac{R(K^{(r)})}{n\,h^{2r+1}} + \frac{\mu_2^2(K)}{4}\,h^4\,R(f^{(r+2)})$$

where $R(K^{(r)}) = \int (K^{(r)})^2\,dx$ and $\mu_2(K) = \int x^2 K(x)\,dx$.

Value

An S4 object of class `dkde` with slots `x` (evaluation points), `y` (estimates $\hat{f}_h^{(r)}(x_i)$), `h`, `kernel`, `deriv.order`, `n`, `data` and `call`. Use `show()` for a summary and `plot()` for a **ggplot2** graphic.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Bhattacharya, P. K. (1967). Estimation of a probability density function and its derivatives. *Sankhya: The Indian Journal of Statistics, Series A*, **29**, 373–382.

Radhey, S. S. (1987). MISE of kernel estimates of a density and its derivatives. *Statistics and Probability Letters*, **5**, 153–159.

Schuster, E. F. (1969). Estimation of a probability density function and its derivatives. *The Annals of Mathematical Statistics*, **40**(4), 1187–1195.

Scott, D. W. (1992). *Multivariate Density Estimation. Theory, Practice and Visualization*. New York: Wiley.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC. London.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

Härdle, W. (1991). *Smoothing Techniques, With Implementation in S*. Springer-Verlag, New York.

See Also

`kernel.fun`, `h.ucv`, `h.bcv`, `h.scv`, `h.pcv`, `h.amise`, `density` (for $r = 0$).

Examples

```
x <- rnorm(100)

## density (r = 0) with automatic UCV bandwidth
est0 <- dkde(data = x, deriv.order = 0)
est0
plot(est0)
```

```
## first derivative (r = 1)
est1 <- dkde(data = x, deriv.order = 1)
plot(est1)

## fixed bandwidth, other kernels; slots are accessed with @
est2 <- dkde(data = x, h = 0.5, kernel = "epanechnikov")
head(cbind(x = est2@x, y = est2@y))
plot(dkde(data = x, h = 0.6, kernel = kernel.pow(2, 3)), rug = FALSE)
```

`h.amise` *AMISE Reference (Maximal Smoothing) Bandwidth*

Description

Computes the reference bandwidth h_{os} for the r'th derivative of a kernel density estimator: the AMISE-optimal bandwidth in which $R(f^{(r+2)})$ is replaced by the value given by the maximal smoothing principle of Terrell (1990). For $r = 0$ it is the oversmoothed bandwidth, an upper bound for the optimal one. Returns an S4 object of class `bandwidth`.

Usage

```
h.amise(x, deriv.order = 0L, kernel = "gaussian")
```

Arguments

`x`	numeric vector of data values.
`deriv.order`	non-negative integer: derivative order r. The rule needs a square-integrable $K^{(r)}$, so $r \leq q$, where q is the regularity of the kernel. Maximum r: any for Gaussian; logistic 8; triweight and tricube 3; biweight 2; Epanechnikov, triangular and cosine 1; uniform 0; `pow(s,t)` q, at most 6, and lower when $K^{(r)}$ is not square integrable. The Silverman kernel is of order 4 ($\mu_2(K) = 0$): the value of the Gaussian kernel is returned with a warning. Above it `h = NA` is returned with a warning. See `kedd-package`, section 'Maximum derivative order'.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).

Details

The AMISE of the r'th derivative estimator is (Scott 1992, p.\,131):

$$AMISE(h;r) = \frac{R(K^{(r)})}{nh^{2r+1}} + \frac{\mu_2^2(K)}{4}\, h^4\, R\Big(f^{(r+2)}\Big)$$

The minimising bandwidth is:

$$h_r^* = \left[\frac{(2r+1)\, R(K^{(r)})}{\mu_2^2(K)\, R(f^{(r+2)})}\right]^{1/(2r+5)} n^{-1/(2r+5)}$$

$R(f^{(r+2)})$ is replaced by $35/(243\,\hat{\sigma}^{2r+5})$ (maximal smoothing principle, Terrell 1990), where $\hat{\sigma}$ is the sample standard deviation, which gives

$$h_{os} = \hat{\sigma}\left[\frac{243\,(2r+1)\,R(K^{(r)})}{35\,\mu_2^2(K)\,n}\right]^{1/(2r+5)}.$$

The same value bounds the search interval of the cross-validation selectors ($[0.01\,h_{os}, 2\,h_{os}]$).

No optimisation is performed; the result is computed directly. $R(K^{(r)}) = \int K^{(r)}(x)^2\,dx$, $\mu_2(K) = \int x^2 K(x)\,dx$.

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` reference bandwidth h_{os}.

`min.obj` `NA` (no minimisation is performed).

`hos` same as `h` (the reference bandwidth).

`method` `"AMISE"`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Radhey, S. S. (1987). MISE of kernel estimates of a density and its derivatives. *Statistics and Probability Letters*, **5**, 153–159.

Scott, D. W. (1992). *Multivariate Density Estimation. Theory, Practice and Visualization*. New York: Wiley.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC. London.

Terrell, G. R. (1990). The maximal smoothing principle in density estimation. *Journal of the American Statistical Association*, **85**, 470–477.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

`bandwidth`, `h.ucv`, `h.bcv`, `h.ccv`, `h.mcv`, `h.tcv`, `h.scv`, `h.pcv`, `h.mlcv`, `dkde`.

Examples

```
x <- rnorm(100)

## density (r = 0)
bw0 <- h.amise(x, deriv.order = 0)
show(bw0)

## first derivative (r = 1)
bw1 <- h.amise(x, deriv.order = 1)
show(bw1)
## other kernels on a Marron-Wand sample
h.amise(kurtotic, kernel = "logistic")
h.amise(kurtotic, kernel = kernel.pow(2, 4))
```

---

h.bcv *Biased Cross-Validation for Bandwidth Selection*

Description

Computes the biased cross-validation bandwidth selector (BCV1 or BCV2) of the r'th derivative of a kernel density estimator (one-dimensional). Returns an S4 object of class `bandwidth`.

Usage

```
h.bcv(x, whichbcv = 1L, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

`x`	numeric vector of data values.
`whichbcv`	integer: `1` for BCV1 (Scott and Terrell 1987, default) or `2` for BCV2 (Jones and Kappenman 1992).
`deriv.order`	non-negative integer: derivative order r. BCV1 uses the convolution of $K^{(r+2)}$, so $r+2 \le q$, where q is the regularity of the kernel: any for Gaussian; logistic 8; triweight and tricube 1; biweight 0; not available for Epanechnikov, uniform, triangular, cosine and Silverman (order 4, $\mu_2(K)=0$); `pow(s,t)` $q-2$, at most 6. BCV2 evaluates $K^{(2r+4)}$, so $2r+4 \le q$: any for Gaussian; logistic 8; not available for the other named kernels; `pow(s,t)` $\lfloor q/2 \rfloor - 2$, at most 6. Above it `h = NA` is returned with a warning. See `kedd-package`, section 'Maximum derivative order'.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`lower, upper`	bounds of the search interval. Defaults to `[0.01 * hos, 2 * hos]`.
`tol`	convergence tolerance for `optimize`. Default `0.01 * lower` (after validation of `lower`).

`nthreads` number of OpenMP threads for this call: `0` (default) = all processors, `k > 0` = `k` threads, both capped by the environment variable `OMP_THREAD_LIMIT`. The result does not depend on it, and it never changes the threads used by later calls.

Details

Let

$$\hat{\theta}_r(h) = \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_{i=1}^{n} \sum_{j \neq i} K^{(r)} * K^{(r)} \left(\frac{X_j - X_i}{h} \right)$$

and

$$\bar{\theta}_r(h) = \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_{i=1}^{n} \sum_{j \neq i} K^{(2r)} \left(\frac{X_j - X_i}{h} \right)$$

BCV1 (Scott and Terrell 1987) minimises:

$$BCV1(h; r) = \frac{R\left(K^{(r)}\right)}{nh^{2r+1}} + \frac{\mu_2^2(K)}{4} h^4 \hat{\theta}_{r+2}(h)$$

BCV2 (Jones and Kappenman 1992) minimises:

$$BCV2(h; r) = \frac{R\left(K^{(r)}\right)}{nh^{2r+1}} + \frac{\mu_2^2(K)}{4} h^4 \bar{\theta}_{r+2}(h)$$

where $K^{(r)} * K^{(r)}$ is the convolution of the r'th kernel derivative (see `kernel.conv`, `kernel.fun`), $R(K^{(r)}) = \int K^{(r)}(x)^2 \, dx$, $\mu_2(K) = \int x^2 K(x) \, dx$.

BCV2 may have several local minima; `optimize` returns one of them without a warning, so the curve `plot(h.bcv(x, whichbcv = 2))` should be checked.

The search is bounded by the over-smoothing bandwidth `hos` (Terrell and Scott 1985; Scott 1992, p.\,165; Wand and Jones 1995, p.\,61).

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` selected bandwidth.

`min.obj` minimum BCV value.

`hos` over-smoothing bandwidth (upper bound).

`method` `"BCV1"` or `"BCV2"`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Jones, M. C. and Kappenman, R. F. (1992). On a class of kernel density estimate bandwidth selectors. *Scandinavian Journal of Statistics*, **19**, 337–349.

Peter, H. and Marron, J. S. (1987). Estimation of integrated squared density derivatives. *Statistics and Probability Letters*, **6**, 109–115.

Scott, D. W. and Terrell, G. R. (1987). Biased and unbiased cross-validation in density estimation. *Journal of the American Statistical Association*, **82**, 1131–1146.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

Härdle, W. (1991). *Smoothing Techniques, With Implementation in S*. Springer-Verlag, New York.

See Also

bandwidth, h.ucv, h.ccv, h.mcv, h.tcv, h.scv, h.pcv, h.amise, kernel.conv, kernel.fun, dkde.

Examples

```
x <- rnorm(100)

## BCV1, density (r = 0)
bw1 <- h.bcv(x, whichbcv = 1, deriv.order = 0)
show(bw1)

## BCV2, first derivative (r = 1)
bw2 <- h.bcv(x, whichbcv = 2, deriv.order = 1)
show(bw2)
plot(bw2)
## other kernels on a Marron-Wand sample
h.bcv(bimodal, kernel = "logistic")
h.bcv(bimodal, whichbcv = 2, kernel = kernel.pow(2, 4))
```

---

h.ccv *Complete Cross-Validation for Bandwidth Selection*

Description

Computes the complete cross-validation bandwidth selector of the r'th derivative of a kernel density estimator (one-dimensional). Returns an S4 object of class bandwidth.

Usage

```
h.ccv(x, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

`x`	numeric vector of data values.
`deriv.order`	non-negative integer: derivative order r. CCV evaluates $K^{(2r+4)}$ at the pairwise differences, so $2r+4 \le q$, where q is the regularity of the kernel. Maximum r: any for Gaussian; logistic 8; `pow(s,t)` $\lfloor q/2 \rfloor - 2$, at most 6; not available for the other named kernels ($q \le 3$; Silverman is also of order 4). Above it `h = NA` is returned with a warning. See `kedd-package`, section 'Maximum derivative order'.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`lower, upper`	bounds of the search interval. Defaults to `[0.01 * hos, 2 * hos]`.
`tol`	convergence tolerance for `optimize`. Default `0.01 * lower` (after validation of `lower`).
`nthreads`	number of OpenMP threads for this call: `0` (default) = all processors, `k > 0` = `k` threads, both capped by the environment variable `OMP_THREAD_LIMIT`. The result does not depend on it, and it never changes the threads used by later calls.

Details

Jones and Kappenman (1992) proposed *complete cross-validation* (CCV) for kernel density estimation. The method estimates the integrated squared density derivative using $\bar{\theta}_r(h)$ (Peter and Marron 1987). $\hat{h}_{CCV}^{(r)}$ minimises:

$$CCV(h;r) = R\left(\hat{f}_h^{(r)}\right) - \bar{\theta}_r(h) + \tfrac{1}{2}\mu_2(K)\, h^2\, \bar{\theta}_{r+1}(h) + \tfrac{1}{24}\left(6\mu_2^2(K) - \delta(K)\right) h^4\, \bar{\theta}_{r+2}(h)$$

with

$$R\left(\hat{f}_h^{(r)}\right) = \frac{R(K^{(r)})}{nh^{2r+1}} + \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_i \sum_{j \ne i} K^{(r)} * K^{(r)}\left(\tfrac{X_j - X_i}{h}\right)$$

and

$$\bar{\theta}_r(h) = \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_i \sum_{j \ne i} K^{(2r)}\left(\tfrac{X_j - X_i}{h}\right)$$

where $R(K^{(r)}) = \int K^{(r)}(x)^2\, dx$, $\mu_2(K) = \int x^2 K(x)\, dx$, $\delta(K) = \int x^4 K(x)\, dx$.

The search is bounded by the over-smoothing bandwidth `hos` (Scott 1992, p.\,165; Wand and Jones 1995, p.\,61).

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` selected bandwidth.

`min.obj` minimum CCV value.

`hos` over-smoothing bandwidth (upper bound).

`method` "CCV".

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Jones, M. C. and Kappenman, R. F. (1992). On a class of kernel density estimate bandwidth selectors. *Scandinavian Journal of Statistics*, **19**, 337–349.

Peter, H. and Marron, J. S. (1987). Estimation of integrated squared density derivatives. *Statistics and Probability Letters*, **6**, 109–115.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

bandwidth, h.ucv, h.bcv, h.mcv, h.tcv, h.scv, h.pcv, h.amise, kernel.conv, kernel.fun.

Examples

```
x <- rnorm(100)

## density (r = 0)
bw0 <- h.ccv(x, deriv.order = 0)
show(bw0)
plot(bw0)

## first derivative (r = 1)
bw1 <- h.ccv(x, deriv.order = 1)
show(bw1)
## other kernels on a Marron-Wand sample
h.ccv(trimodal, kernel = "logistic")
h.ccv(trimodal, kernel = "pow(2,4)")
```

---

h.mcv *Modified Cross-Validation for Bandwidth Selection*

Description

Computes the modified cross-validation bandwidth selector of the r'th derivative of a kernel density estimator (one-dimensional). Returns an S4 object of class bandwidth.

Usage

```
h.mcv(x, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

x	numeric vector of data values.
deriv.order	non-negative integer: derivative order r. MCV evaluates $K^{(2r+2)}$ at the pairwise differences, so $2r+2 \le q$, where q is the regularity of the kernel. Maximum r: any for Gaussian; logistic 8; biweight, triweight and tricube 0; pow(s,t) $\lfloor q/2 \rfloor - 1$, at most 6; not available for Epanechnikov, uniform, triangular and cosine, nor for Silverman (order 4, $\mu_2(K) = 0$). Above it h = NA is returned with a warning. See kedd-package, section 'Maximum derivative order'.
kernel	character string giving the kernel; one of "gaussian" (default), "epanechnikov", "uniform", "triangular", "biweight", "triweight", "tricube", "cosine", "silverman", "logistic", or a power kernel "pow(s,t)" (see kernel.pow).
lower, upper	bounds of the search interval. Defaults to [0.01 * hos, 2 * hos].
tol	convergence tolerance for optimize. Default 0.01 * lower (after validation of lower).
nthreads	number of OpenMP threads for this call: 0 (default) = all processors, k > 0 = k threads, both capped by the environment variable OMP_THREAD_LIMIT. The result does not depend on it, and it never changes the threads used by later calls.

Details

Stute (1992) proposed *modified cross-validation* (MCV), a modification of unbiased cross-validation that approximates the problematic term via the Hajek projection. $\hat{h}_{MCV}^{(r)}$ minimises:

$$MCV(h;r) = \frac{R(K^{(r)})}{nh^{2r+1}} + \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_i \sum_{j \neq i} \varphi^{(r)}\left(\frac{X_j - X_i}{h}\right)$$

where

$$\varphi^{(r)}(c) = \left(K^{(r)} * K^{(r)} - K^{(2r)} - \tfrac{\mu_2(K)}{2} K^{(2r+2)}\right)(c)$$

and $K^{(r)} * K^{(r)}$ is the convolution of the r'th kernel derivative (see kernel.conv, kernel.fun), $R(K^{(r)}) = \int K^{(r)}(x)^2\,dx$, $\mu_2(K) = \int x^2 K(x)\,dx$.

The search is bounded by the over-smoothing bandwidth hos (Scott 1992, p.\,165; Wand and Jones 1995, p.\,61).

Value

An S4 object of class bandwidth with slots:

data the (cleaned) data vector.

kernel name of the kernel used.

deriv.order derivative order r.

`h` selected bandwidth.

`min.obj` minimum MCV value.

`hos` over-smoothing bandwidth (upper bound).

`method` "MCV".

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Heidenreich, N. B., Schindler, A. and Sperlich, S. (2013). Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *Advances in Statistical Analysis*.

Stute, W. (1992). Modified cross validation in density estimation. *Journal of Statistical Planning and Inference*, **30**, 293–305.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

`bandwidth`, `h.ucv`, `h.bcv`, `h.ccv`, `h.tcv`, `h.scv`, `h.pcv`, `h.amise`, `kernel.conv`, `kernel.fun`.

Examples

```
x <- rnorm(100)

## density (r = 0)
bw0 <- h.mcv(x, deriv.order = 0)
show(bw0)
plot(bw0)

## first derivative (r = 1)
bw1 <- h.mcv(x, deriv.order = 1)
show(bw1)
## other kernels on a Marron-Wand sample
h.mcv(claw, kernel = "logistic")
h.mcv(claw, kernel = kernel.pow(2, 2))
```

---

h.mlcv *Maximum-Likelihood Cross-Validation for Bandwidth Selection*

Description

Computes the maximum-likelihood (Kullback-Leibler) cross-validation bandwidth selector for a kernel density estimator (density estimation only, `deriv.order = 0`). Returns an S4 object of class `bandwidth`.

Usage

```
h.mlcv(x, kernel = "gaussian", lower, upper, tol, nthreads = 0L)
```

Arguments

x	numeric vector of data values.
kernel	character string giving the kernel; one of "gaussian" (default), "epanechnikov", "uniform", "triangular", "biweight", "triweight", "tricube", "cosine", "silverman", "logistic", or a power kernel "pow(s,t)" (see kernel.pow).
lower, upper	bounds of the search interval. Defaults to [0.01 * hos, 2 * hos].
tol	convergence tolerance for optimize. Default 0.01 * lower.
nthreads	number of OpenMP threads for this call: 0 (default) = all processors, k > 0 = k threads, both capped by the environment variable OMP_THREAD_LIMIT. The result does not depend on it, and it never changes the threads used by later calls.

Details

Habbema, Hermans, and Van den Broek (1974) and Duin (1976) proposed maximising the leave-one-out log-likelihood:

$$MLCV(h) = n^{-1} \sum_{i=1}^{n} \log\left[\hat{f}_{h,i}(X_i)\right]$$

where the leave-one-out estimator is

$$\hat{f}_{h,i}(X_i) = \frac{1}{(n-1)h} \sum_{j \neq i} K\left(\frac{X_j - X_i}{h}\right)$$

Equivalently,

$$h_{mlcv} = \arg\max_h \left(n^{-1} \sum_i \log\left[\sum_{j \neq i} K\left(\frac{X_j - X_i}{h}\right)\right] - \log[(n-1)h] \right)$$

Only applicable to density estimation ($r = 0$).

Value

An S4 object of class bandwidth with slots:

data the (cleaned) data vector.

kernel name of the kernel used.

deriv.order 0L (density only).

h selected bandwidth.

min.obj negative of the maximum MLCV value.

hos over-smoothing bandwidth (upper bound).

method "MLCV".

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Duin, R. P. W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. *IEEE Transactions on Computers*, **C-25**, 1175–1179.

Habbema, J. D. F., Hermans, J. and Van den Broek, K. (1974). A stepwise discrimination analysis program using density estimation. *Compstat 1974: Proceedings in Computational Statistics*. Physica Verlag, Vienna.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

bandwidth, h.ucv, h.amise, h.scv, h.pcv, dkde.

Examples

```
x <- rnorm(100)

bw <- h.mlcv(x)
show(bw)
plot(bw)

bw2 <- h.mlcv(x, kernel = "epanechnikov")
show(bw2)
## other kernels on a Marron-Wand sample
h.mlcv(outlier, kernel = "logistic")
h.mlcv(outlier, kernel = "pow(2,2)")
```

---

h.pcv *Partitioned Cross-Validation for Bandwidth Selection*

Description

Computes the partitioned cross-validation (PCV) bandwidth selector of Marron (1987) for the r'th derivative of a kernel density estimator (one-dimensional): the sample is partitioned at random into `m` subsamples, a cross-validation score is computed in each subsample, the bandwidth minimises their average and is rescaled from the subsample size to the sample size. Returns an S4 object of class bandwidth.

Usage

```
h.pcv(x, m = 2L, criterion = "UCV", nrep = 1L, deriv.order = 0L,
      kernel = "gaussian", lower, upper, tol, nthreads = 0L)
```

Arguments

x	numeric vector of data values.
m	number of subsamples, a positive integer; each subsample must contain at least 2 observations. m = 1 gives the score itself.
criterion	cross-validation score computed in each subsample: one of "UCV" (default), "TCV", "MCV", "CCV", "BCV1" and "BCV2" (see h.ucv, h.tcv, h.mcv, h.ccv and h.bcv).
nrep	number of random partitions, a positive integer. nrep = 1 (default) is the method of Marron (1987); with nrep > 1 the average score is also averaged over nrep independent random partitions.
deriv.order	non-negative integer: derivative order r. The maximum order is that of criterion for the kernel; above it h = NA is returned with a warning. See kedd-package, section 'Maximum derivative order'.
kernel	character string giving the kernel; one of "gaussian" (default), "epanechnikov", "uniform", "triangular", "biweight", "triweight", "tricube", "cosine", "silverman", "logistic", or a power kernel "pow(s,t)" (see kernel.pow).
lower, upper	bounds of the search interval for the bandwidth h of the whole sample. Defaults to [0.01 * hos, 2 * hos], where hos is the over-smoothing bandwidth of the whole sample.
tol	convergence tolerance passed to optimize. Default 0.01 * lower (after validation of lower).
nthreads	number of OpenMP threads for this call: 0 (default) = all processors, k > 0 = k threads, both capped by the environment variable OMP_THREAD_LIMIT.

Details

Marron (1987) proposed partitioned cross-validation to reduce the across-sample variability of the cross-validation bandwidth. The indices $1, \ldots, n$ are partitioned at random, independently of the data, into m subsamples $S_1, \ldots, S_m$ of sizes about n/m (they differ by at most one). If $CV_j(b; r)$ denotes the score criterion computed on the observations of S_j with the bandwidth b, the average score $m^{-1} \sum_j CV_j(b; r)$ is minimised by a bandwidth $\hat{b}$ suited to a sample of size n/m, and since the optimal bandwidth is of order $n^{-1/(2r+5)}$,

$$\hat{h}_{PCV} = m^{-1/(2r+5)}\, \hat{b}$$

is the bandwidth of the whole sample. Equivalently, h.pcv() minimises in h

$$PCV(h; r) = \frac{1}{m} \sum_{j=1}^{m} CV_j\big(h\, m^{1/(2r+5)}; r\big).$$

For $r = 0$ and UCV, this is exactly the method of Marron (1987), with the rescaling $m^{-1/5}$. The other scores, which Marron suggests (for BCV) but does not study, the derivatives ($r \geq 1$) and nrep > 1, an average over several random partitions in the spirit of the suggestion of N. I. Fisher reported by Marron (1987, Section 4), are extensions. For the fourth-order Silverman kernel the exponent $1/(2r + 5)$ is replaced by $1/(2r + 9)$.

A larger m reduces the variance of the bandwidth across samples but increases a bias, since the rescaling is only asymptotically correct (Marron 1987, Section 3); for derivatives the subsamples should be larger. The bandwidth also depends on the random partition; `nrep` > 1 makes it more stable, at a cost multiplied by `nrep`. Each subsample contains about $n^2/(2m^2)$ pairs, so that one partition costs about $1/m$ of the score on the whole sample.

As for the other selectors, the average score is minimised with `optimize` over the search interval; it may have several local minima, and for CCV and BCV2 a spurious one at small bandwidths, so that its curve (`plot()`) should be checked and `lower` or `upper` changed if needed. The partition is random: use `set.seed` for reproducible results. The partitions are kept in the slot `groups`, so that `plot()` and `lines()` draw $PCV(h; r)$.

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` selected bandwidth for the whole sample.

`min.obj` minimum value of $PCV(h; r)$.

`hos` over-smoothing bandwidth of the whole sample.

`method` `"PCV"`.

`criterion` the score computed in each subsample.

`groups` the subsample of each observation, for each partition one after the other (length `nrep * length(data)`).

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Hall, P. and Robinson, A. P. (2009). Reducing variability of crossvalidation for smoothing-parameter choice. *Biometrika*, **96**, 175–186.

Marron, J. S. (1987). Partitioned cross-validation. *Econometric Reviews*, **6**(2), 271–283. doi:10.1080/07474938708800136

See Also

`bandwidth`, `h.ucv`, `h.tcv`, `h.mcv`, `h.ccv`, `h.bcv`, `h.scv`, `h.amise`, `h.mlcv`, `dkde`.

Examples

```
set.seed(1)
x <- rnorm(400)

## density (r = 0), UCV in 4 subsamples: the method of Marron (1987)
bw0 <- h.pcv(x, m = 4)
bw0
plot(bw0)

## first derivative (r = 1), TCV in 2 subsamples
h.pcv(x, m = 2, criterion = "TCV", deriv.order = 1)

## second derivative (r = 2), UCV in 4 subsamples averaged over 10 partitions
h.pcv(x, m = 4, nrep = 10, deriv.order = 2)

## another kernel
h.pcv(bimodal, m = 2, kernel = "pow(2,4)")
```

---

h.scv *Smoothed Cross-Validation for Bandwidth Selection*

Description

Selects the bandwidth of the r'th derivative kernel density estimator by minimising the smoothed cross-validation (SCV) criterion of Hall, Marron and Park (1992). Returns an S4 object of class bandwidth.

Usage

```
h.scv(x, g = NULL, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

x	numeric vector of data values.
g	pilot bandwidth (positive number). If NULL (default), $g = \hat{\sigma}\, 0.9266\, n^{-2/13}$ (Hall et al. 1992, eq. 4.2), whatever deriv.order.
deriv.order	non-negative integer: derivative order r; no upper limit (Gaussian kernel). See kedd-package, section 'Maximum derivative order'.
kernel	must be "gaussian", the only kernel for which the SCV criterion is available in closed form.
lower, upper	bounds of the search interval. Defaults to [0.01 * hos, 2 * hos].
tol	convergence tolerance for optimize. Default 0.01 * lower (after validation of lower).
nthreads	number of OpenMP threads for this call: 0 (default) = all processors, k > 0 = k threads, both capped by the environment variable OMP_THREAD_LIMIT. The result does not depend on it, and it never changes the threads used by later calls.

Details

With a Gaussian kernel and a Gaussian pilot estimate $\tilde{f}_g$, the integrated squared bias of $\hat{f}_h^{(r)}$ is estimated in closed form by derivatives of normal densities:

$$SCV(h;r) = \frac{R(K^{(r)})}{nh^{2r+1}} + \frac{(-1)^r\, 2}{n(n-1)} \sum_{i<j} \left[\phi_{s_1}^{(2r)} - 2\phi_{s_2}^{(2r)} + \phi_{s_3}^{(2r)}\right](X_j - X_i),$$

where $s_1 = \sqrt{2(h^2+g^2)}$, $s_2 = \sqrt{h^2+2g^2}$, $s_3 = \sqrt{2}\,g$ and $\phi_s^{(k)}(u) = \phi^{(k)}(u/s)/s^{k+1}$. It follows from $\int[((K_h-\delta)*\tilde{f}_g)^{(r)}]^2 = (-1)^r \int((K_h-\delta)*\tilde{f}_g)\,((K_h-\delta)*\tilde{f}_g)^{(2r)}$. For $r=0$ it is the criterion of Hall, Marron and Park (1992).

The default pilot bandwidth is the density pilot of Hall et al., the same for every r; it is not optimal for derivatives, for which a pilot can be supplied through g.

Value

An S4 object of class `bandwidth` with `method = "SCV"`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Hall, P., Marron, J. S. and Park, B. U. (1992). Smoothed cross-validation. *Probability Theory and Related Fields*, **92**, 1–20.

See Also

`h.ucv`, `h.bcv`, `h.pcv`, `bandwidth`.

Examples

```
x <- rnorm(200)
bw <- h.scv(x)
show(bw)
plot(bw)

## first derivative (r = 1)
bw1 <- h.scv(x, deriv.order = 1)
plot(dkde(data = x, h = bw1@h, deriv.order = 1))
## on a Marron-Wand sample (SCV is defined for the Gaussian kernel)
h.scv(bimodal)
h.scv(bimodal, deriv.order = 2)
```

h.tcv *Trimmed Cross-Validation for Bandwidth Selection*

Description

Computes the trimmed cross-validation bandwidth selector of the r'th derivative of a kernel density estimator (one-dimensional). Returns an S4 object of class `bandwidth`.

Usage

```
h.tcv(x, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

`x`	numeric vector of data values.
`deriv.order`	non-negative integer: derivative order r. The criterion evaluates $K^{(2r)}$ at the pairwise differences, so $2r \leq q$, where q is the regularity of the kernel. Maximum r: any for Gaussian; logistic 8; biweight, triweight, tricube and Silverman 1; Epanechnikov, uniform, triangular and cosine 0; `pow(s,t)` $\lfloor q/2 \rfloor$, at most 6. Above it `h = NA` is returned with a warning. See `kedd-package`, section 'Maximum derivative order'.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`lower, upper`	bounds of the search interval. Defaults to `[0.01 * hos, 2 * hos]`.
`tol`	convergence tolerance for `optimize`. Default `0.01 * lower` (after validation of `lower`).
`nthreads`	number of OpenMP threads for this call: `0` (default) = all processors, `k > 0` = `k` threads, both capped by the environment variable `OMP_THREAD_LIMIT`. The result does not depend on it, and it never changes the threads used by later calls.

Details

Feluch and Koronacki (1992) proposed *trimmed cross-validation* (TCV), a modification of unbiased cross-validation in which the term $-2K^{(2r)}$ of the pairs at small distances is removed. $\hat{h}_{TCV}^{(r)}$ minimises:

$$TCV(h;r) = \frac{R(K^{(r)})}{nh^{2r+1}} + \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_i \sum_{j \neq i} \varphi^{(r)}\left(\frac{X_j - X_i}{h}\right)$$

where

$$\varphi^{(r)}(c) = \left(K^{(r)} * K^{(r)} - 2\,K^{(2r)}\chi\big(|c| > c_n/h^{2r+1}\big)\right)(c)$$

with $c_n = 1/n$ (ensuring convergence), $\chi(.)$ the indicator function, and $K^{(r)} * K^{(r)}$ the convolution of the r'th kernel derivative (see `kernel.conv`, `kernel.fun`).

The threshold c_n/h^{2r+1} applies to the scaled differences $(X_j - X_i)/h$; for $r = 0$ it is the fixed distance $|X_j - X_i| > 1/n$. TCV is therefore not scale equivariant: the bandwidth of `c * x` is not `c` times the bandwidth of `x`, and the data are best standardised first.

The search is bounded by the over-smoothing bandwidth `hos` (Scott 1992, p.\,165; Wand and Jones 1995, p.\,61).

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` selected bandwidth.

`min.obj` minimum TCV value.

`hos` over-smoothing bandwidth (upper bound).

`method` `"TCV"`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Feluch, W. and Koronacki, J. (1992). A note on modified cross-validation in density estimation. *Computational Statistics and Data Analysis*, **13**, 143–151.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

See Also

`bandwidth`, `h.ucv`, `h.bcv`, `h.ccv`, `h.mcv`, `h.amise`, `h.scv`, `h.pcv`, `kernel.conv`, `kernel.fun`.

Examples

```
x <- rnorm(100)

## density (r = 0)
bw0 <- h.tcv(x, deriv.order = 0)
show(bw0)
plot(bw0)

## first derivative (r = 1)
bw1 <- h.tcv(x, deriv.order = 1)
show(bw1)
## other kernels on a Marron-Wand sample
h.tcv(kurtotic, kernel = "logistic")
h.tcv(kurtotic, kernel = "pow(2,3)")
```

h.ucv *Unbiased (Least-Squares) Cross-Validation for Bandwidth Selection*

Description

Computes the unbiased (least-squares) cross-validation bandwidth selector of the r'th derivative of a kernel density estimator (one-dimensional). Returns an S4 object of class `bandwidth`.

Usage

```
h.ucv(x, deriv.order = 0L, kernel = "gaussian",
      lower, upper, tol, nthreads = 0L)
```

Arguments

`x`	numeric vector of data values.
`deriv.order`	non-negative integer: derivative order r. The criterion evaluates $K^{(2r)}$ at the pairwise differences, so $2r \leq q$, where q is the regularity of the kernel. Maximum r: any for Gaussian; logistic 8; biweight, triweight, tricube and Silverman 1; Epanechnikov, uniform, triangular and cosine 0; `pow(s,t)` $\lfloor q/2 \rfloor$, at most 6. Above it `h = NA` is returned with a warning. See `kedd-package`, section 'Maximum derivative order'.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`lower, upper`	bounds of the search interval. Defaults to `[0.01 * hos, 2 * hos]` where `hos` is the over-smoothing bandwidth computed internally.
`tol`	convergence tolerance passed to `optimize`. Default `0.01 * lower` (after validation of `lower`).
`nthreads`	number of OpenMP threads for this call: `0` (default) = all processors, `k > 0` = `k` threads, both capped by the environment variable `OMP_THREAD_LIMIT`. The result does not depend on it, and it never changes the threads used by later calls.

Details

Rudemo (1982) and Bowman (1984) proposed unbiased (least-squares) cross-validation (UCV) for kernel density estimation. Härdle, Marron and Wand (1990) extended the criterion to the r'th derivative estimator. The bandwidth $\hat{h}_{UCV}^{(r)}$ minimises:

$$UCV(h;r) = \frac{R\left(K^{(r)}\right)}{nh^{2r+1}} + \frac{(-1)^r}{n(n-1)h^{2r+1}} \sum_{i=1}^{n} \sum_{j \neq i} \left(K^{(r)} * K^{(r)} - 2\,K^{(2r)} \right) \left(\frac{X_j - X_i}{h} \right)$$

where $K^{(r)} * K^{(r)}(x)$ is the convolution of the r'th derivative of the kernel (see `kernel.conv`, `kernel.fun`), and $R(K^{(r)}) = \int K^{(r)}(x)^2\,dx$.

UCV is unbiased in the sense that $E[UCV(h;r)] = MISE[\hat{f}_h^{(r)}] - R(f^{(r)})$ (Scott and Terrell 1987).

The search is bounded by the over-smoothing bandwidth `hos` (Terrell and Scott 1985; Scott 1992, p.\,165; Wand and Jones 1995, p.\,61).

Value

An S4 object of class `bandwidth` with slots:

`data` the (cleaned) data vector.

`kernel` name of the kernel used.

`deriv.order` derivative order r.

`h` selected bandwidth.

`min.obj` minimum UCV value.

`hos` over-smoothing bandwidth (upper bound).

`method` `"UCV"`.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Bowman, A. (1984). An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, **71**, 353–360.

Rudemo, M. (1982). Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, **9**, 65–78.

Scott, D. W. and Terrell, G. R. (1987). Biased and unbiased cross-validation in density estimation. *Journal of the American Statistical Association*, **82**, 1131–1146.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

Härdle, W., Marron, J. S. and Wand, M. P. (1990). Bandwidth choice for density derivatives. *Journal of the Royal Statistical Society, Series B*, 223–232.

See Also

`bandwidth`, `h.bcv`, `h.ccv`, `h.mcv`, `h.tcv`, `h.scv`, `h.pcv`, `h.amise`, `kernel.conv`, `kernel.fun`, `dkde`.

Examples

```
x <- rnorm(100)

## density (r = 0)
bw0 <- h.ucv(x, deriv.order = 0)
show(bw0)
```

```
plot(bw0)

## first derivative (r = 1)
bw1 <- h.ucv(x, deriv.order = 1)
show(bw1)
## other kernels on a Marron-Wand sample
h.ucv(claw, kernel = "logistic")
h.ucv(claw, kernel = "pow(2,3)")
```

---

kedd-classes *Classes* "dkde", "kfun" *and* "kconv"

Description

S4 classes returned by dkde (kernel density derivative estimate), kernel.fun (kernel derivative $K^{(r)}$) and kernel.conv (convolution $K^{(r)} * K^{(r)}$).

Slots

Common to the three classes:

x: "numeric", evaluation points.

y: "numeric", values at x.

kernel: "character", kernel name.

deriv.order: "integer", derivative order r.

call: "call", the matched call.

Class "dkde" has in addition:

h: "numeric", bandwidth used.

n: "integer", sample size.

data: "numeric", the (cleaned) observations.

Methods

show(object) prints a short summary.

plot(x, type = "l", zero.line, main.title, x.title, y.title, colors, title.size, title.face, axis.size, t
: draws the curve with **ggplot2**, prints it and returns the ggplot object invisibly. type is "l", "p" or "o"; zero.line (default TRUE when $r > 0$) adds a dashed horizontal line at 0. For "dkde" objects, rug and band (default TRUE) add the observations and a $\pm$ one standard-error band.

x$name Compatibility with **kedd** 1.x, where these functions returned lists: $kx is the slot y of "kfun" and "kconv" objects, and $eval.points, $est.fx are the slots x, y of "dkde" objects; the slot names can also be used. Unknown names give NULL.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

See Also

dkde, kernel.fun, kernel.conv, bandwidth.

Examples

```
kf <- kernel.fun(kernel = "logistic", deriv.order = 1L)
kf
p <- plot(kf, type = "o")

kc <- kernel.conv(kernel = "biweight")
plot(kc)

d <- dkde(data = rnorm(100), h = 0.4)
plot(d, band = FALSE)
```

---

kedd-data *Samples from Marron and Wand (1992) Normal Mixture Densities*

Description

Five samples of size 200 drawn from normal mixtures of Marron and Wand (1992), classical test cases for density (derivative) estimation and bandwidth selection. Writing $N(\mu, \sigma^2)$ for a normal density:

`kurtotic` kurtotic unimodal density (#4): $\frac{2}{3}N(0,1) + \frac{1}{3}N(0,(1/10)^2)$.

`outlier` outlier density (#5): $\frac{1}{10}N(0,1) + \frac{9}{10}N(0,(1/10)^2)$.

`bimodal` separated bimodal density (#7): $\frac{1}{2}N(-\frac{3}{2},(\frac{1}{2})^2) + \frac{1}{2}N(\frac{3}{2},(\frac{1}{2})^2)$.

`trimodal` trimodal density (#9): $\frac{9}{20}N(-\frac{6}{5},(\frac{3}{5})^2) + \frac{9}{20}N(\frac{6}{5},(\frac{3}{5})^2) + \frac{1}{10}N(0,(\frac{1}{4})^2)$.

`claw` claw density (#10): $\frac{1}{2}N(0,1) + \sum_{l=0}^{4} \frac{1}{10}N(\frac{l}{2} - 1,(\frac{1}{10})^2)$.

Usage

```
bimodal
claw
kurtotic
outlier
trimodal
```

Format

Each object is a numeric vector of length 200.

Details

The datasets are loaded lazily: `claw` is available as soon as the package is attached (`data(claw)` also works). The true densities are known, so the estimates $\hat{f}_h^{(r)}$ and the selected bandwidths can be compared with the truth.

Source

Simulated samples, as in previous versions of **kedd**.

References

Marron, J. S. and Wand, M. P. (1992). Exact mean integrated squared error. *The Annals of Statistics*, **20**, 712–736.

See Also

dkde, h.ucv, kedd-package.

Examples

```
## the five samples
sapply(list(kurtotic = kurtotic, outlier = outlier, bimodal = bimodal,
            trimodal = trimodal, claw = claw), summary)

## claw density: estimate with an MCV bandwidth and the true density
bw <- h.mcv(claw)
f_claw <- function(x) 0.5 * dnorm(x) +
  rowSums(sapply(0:4, function(l) 0.1 * dnorm(x, l / 2 - 1, 0.1)))
p <- plot(dkde(data = claw, h = bw@h))
p + ggplot2::stat_function(fun = f_claw, linetype = "dashed")

## first derivative of the bimodal sample
plot(dkde(data = bimodal, h = h.ucv(bimodal, deriv.order = 1)@h, deriv.order = 1))
```

---

kernel.conv *Evaluate kernel convolution $K^(r) * K^(r)$*

Description

Evaluates the self-convolution $(K^{(r)} * K^{(r)})(x)$ of the r-th derivative of the kernel K. Returns an S4 object of class `kconv`.

Usage

```
kernel.conv(x = NULL, deriv.order = 0L, kernel = "gaussian", n.grid = 512L)
```

Arguments

x	numeric vector of evaluation points; if NULL a grid of n.grid points over $[-8, 8]$ (Gaussian, Silverman, logistic) or $[-2.1, 2.1]$ (compact-support kernels) is used.
deriv.order	derivative order $r \geq 0$ (integer). Maximum r with a non-zero convolution: any for Gaussian, cosine and Silverman; logistic 20; Epanechnikov 2; uniform 0; triangular 1; biweight 4; triweight 6; tricube 9; pow(s,t) $\min(st, 16)$ (16 if s or t is not an integer). See kedd-package, section 'Maximum derivative order'. Above the regularity q of the kernel (see kedd-package, section 'Maximum derivative order'), the values are pointwise derivatives, which miss the Dirac masses of the derivative of K.
kernel	character string giving the kernel; one of "gaussian" (default), "epanechnikov", "uniform", "triangular", "biweight", "triweight", "tricube", "cosine", "silverman", "logistic", or a power kernel "pow(s,t)" (see kernel.pow).
n.grid	number of grid points when x is NULL (default 512).

Details

Closed forms are used for the Gaussian, Silverman and finite-support named kernels. The logistic convolution has no closed form and is computed by composite Gauss-Legendre quadrature centred at $x/2$. For "pow(s,t)" kernels the convolution is computed once per (s, t, r) by quadrature on $[0, 2]$ (split where the integrand is not smooth) and then interpolated. At $x = 0$, $(K^{(r)} * K^{(r)})(0) = (-1)^r R(K^{(r)})$.

Value

An S4 object of class kconv with slots:

x numeric evaluation grid.

y $(K^{(r)} * K^{(r)})(x)$ values.

kernel kernel name.

deriv.order derivative order r.

call the matched call.

Use show() for a summary and plot() for a ggplot2 graphic.

See Also

kernel.fun, kernel.pow, kconv

Examples

```
kc <- kernel.conv(kernel = "gaussian", deriv.order = 0L)
show(kc)
plot(kc)

kc2 <- kernel.conv(kernel = "epanechnikov", deriv.order = 1L)
plot(kc2)
```

```
## logistic (any r) and Pow(s,t)
plot(kernel.conv(kernel = "logistic", deriv.order = 1L))
plot(kernel.conv(kernel = kernel.pow(2, 3)))
```

`kernel.fun` *Evaluate Kernel Function and Its Derivatives*

Description

Evaluates the r-th derivative $K^{(r)}(x)$ of a kernel function K and returns an S4 object of class `kfun`.

Usage

```
kernel.fun(x = NULL, deriv.order = 0L, kernel = "gaussian", n.grid = 512L)
```

Arguments

`x`	numeric vector of evaluation points. If `NULL` (default), a uniform grid of `n.grid` points over $[-8, 8]$ (Gaussian, Silverman, logistic) or $[-1.1, 1.1]$ (compact-support kernels) is used.
`deriv.order`	non-negative integer: derivative order $r \geq 0$. Maximum r with $K^{(r)} \neq 0$: any for Gaussian, cosine and Silverman; logistic 20; Epanechnikov 2; uniform 0; triangular 1; biweight 4; triweight 6; tricube 9; `pow(s,t)` $\min(st, 16)$ (16 if s or t is not an integer). Above it, zero is returned (with a warning for logistic and pow). See `kedd-package`, section 'Maximum derivative order'. Above the regularity q of the kernel (see `kedd-package`, section 'Maximum derivative order'), the values are pointwise derivatives, which miss the Dirac masses of the derivative of K.
`kernel`	character string giving the kernel; one of `"gaussian"` (default), `"epanechnikov"`, `"uniform"`, `"triangular"`, `"biweight"`, `"triweight"`, `"tricube"`, `"cosine"`, `"silverman"`, `"logistic"`, or a power kernel `"pow(s,t)"` (see `kernel.pow`).
`n.grid`	number of grid points when `x` is `NULL` (default 512).

Details

The supported kernels at order $r = 0$:

- **Gaussian**: $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, support $(-\infty, +\infty)$.
- **Epanechnikov**: $K(x) = \frac{3}{4}(1 - x^2)\mathbf{1}_{|x| \leq 1}$.
- **Uniform**: $K(x) = \frac{1}{2}\mathbf{1}_{|x| \leq 1}$.
- **Triangular**: $K(x) = (1 - |x|)\mathbf{1}_{|x| \leq 1}$.
- **Biweight**: $K(x) = \frac{15}{16}(1 - x^2)^2\mathbf{1}_{|x| \leq 1}$.
- **Triweight**: $K(x) = \frac{35}{32}(1 - x^2)^3\mathbf{1}_{|x| \leq 1}$.
- **Tricube**: $K(x) = \frac{70}{81}(1 - |x|^3)^3\mathbf{1}_{|x| \leq 1}$.

- **Cosine**: $K(x) = \frac{\pi}{4}\cos\left(\frac{\pi x}{2}\right)\mathbf{1}_{|x|\le 1}$.
- **Silverman**: $K(x) = \frac{1}{2}e^{-|x|/\sqrt{2}}\sin\left(\frac{|x|}{\sqrt{2}} + \frac{\pi}{4}\right)$, support $(-\infty, +\infty)$.
- **Logistic**: $K(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{4}\mathrm{sech}^2(x/2)$, support $(-\infty, +\infty)$.
- **Pow(s,t)**: $K(x) = c_{s,t}(1-|x|^s)^t\mathbf{1}_{|x|\le 1}$, $c_{s,t} = s/(2B(1/s, t+1))$, given as `"pow(s,t)"` (see `kernel.pow`).

Derivatives are computed analytically via Hermite polynomials (Gaussian, Silverman), Guidoum polynomials (finite-support kernels), the identity $K^{(r)} = \sigma(1-\sigma)Q_{r+1}(\sigma)$ with $\sigma(x) = 1/(1+e^{-x})$ (logistic, $r \le 20$) and exact power expansions (Pow, $r \le 16$).

Value

An S4 object of class `kfun` with slots:

`x` numeric evaluation grid.

`y` $K^{(r)}(x)$ values.

`kernel` kernel name.

`deriv.order` derivative order r.

`call` the matched call.

Use `show()` for a text summary and `plot()` for a ggplot2 graphic.

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

References

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, London.

See Also

`kernel.conv`, `kfun`, `dkde`, `h.ucv`.

Examples

```
## Gaussian kernel (r = 0)
kf <- kernel.fun(kernel = "gaussian", deriv.order = 0L)
show(kf)
plot(kf)

## First derivative of Epanechnikov kernel
kf2 <- kernel.fun(kernel = "epanechnikov", deriv.order = 1L)
plot(kf2)
```

```
## Evaluate at specific points
kf3 <- kernel.fun(x = seq(-2, 2, by = 0.1),
                  kernel = "biweight", deriv.order = 2L)

## Logistic and Pow(s,t) kernels
plot(kernel.fun(kernel = "logistic", deriv.order = 2L))
plot(kernel.fun(kernel = kernel.pow(1.5, 2.5)))
```

`kernel.pow` *Power Kernel Family Pow(s,t)*

Description

Builds the name of a kernel of the power family

$$K_{s,t}(x) = c_{s,t}\,(1-|x|^s)^t\,\mathbf{1}_{|x|\le 1}, \qquad c_{s,t} = \frac{s}{2\,B(1/s,\,t+1)},$$

which can be passed as `kernel` to every function of the package (bandwidth selectors, `dkde`, `kernel.fun`, `kernel.conv`).

Usage

```
kernel.pow(s = 2, t = 2)
```

Arguments

`s`	single positive number: shape exponent $s > 0$.
`t`	single non-negative number: power $t \ge 0$.

Details

The kernel can equally be given directly as the string `"pow(s,t)"`, e.g. `kernel = "pow(2,3)"`; `"pow"` alone means `"pow(2,2)"`. The family contains several classical kernels:

`"pow(s,0)"`	uniform
`"pow(1,1)"`	triangular
`"pow(2,1)"`	Epanechnikov
`"pow(2,2)"`	biweight
`"pow(2,3)"`	triweight
`"pow(3,3)"`	tricube

The constants used by the bandwidth selectors are exact: $\mu_k(K) = B((k+1)/s,\,t+1)/B(1/s,\,t+1)$ for even k and $R(K) = s\,B(1/s,\,2t+1)/(2\,B(1/s,\,t+1)^2)$. For $x > 0$, $K^{(r)}(x)$ is a finite sum of terms $a\,x^p(1-x^s)^q$: an exact binomial expansion when t is an integer ($t \le 10$), a recurrence otherwise; derivatives are available up to order 16.

$R(K^{(r)})$ is finite only when $2r < 2t + 1$ and, unless s is an even integer, $2r < 2s + 1$ (otherwise $K^{(r)}$ is not square integrable at $|x| = 1$ or at 0). Bandwidth selectors return NA with a warning for such derivative orders, as for the other finite-support kernels.

Maximum derivative order. The regularity of $K_{s,t}$ is $q = \min(\lceil t + 1 \rceil - 1, \lceil s + 1 \rceil - 1)$, the term in s being dropped when s is an even integer ($q = t$ for an integer t): for $m \leq q$ the pointwise derivative $K^{(m)}$ is the derivative of K. A criterion that evaluates $K^{(m)}$ needs $m \leq q$: $r \leq q$ for dkde and h.amise, $2r \leq q$ for h.ucv and h.tcv, $2r + 2 \leq q$ for h.mcv, $r + 2 \leq q$ for BCV1 and $2r + 4 \leq q$ for BCV2 and h.ccv, with $K^{(r)}$ square integrable; the selectors are capped at $r = 6$ and dkde at $r = 16$. For example pow(2,4): $q = 4$ and 4, 4, 2, 1, 2, 0; pow(2,3): $q = 3$ and 3, 3, 1, 0, 1, not available. See kedd-package, section 'Maximum derivative order'.

Value

A character string "pow(s,t)".

Author(s)

Arsalane Chouaib Guidoum <acguidoum@univ-tam.dz>, Amel Henider <amel.henider@univ-tam.dz> and Iago Giné-Vázquez <iago.gin-vaz@protonmail.com>

See Also

kernel.fun, kernel.conv, h.ucv, dkde.

Examples

```
kernel.pow(2, 3)                # "pow(2,3)"

x <- rnorm(200)
h.ucv(x, kernel = kernel.pow(2, 2))   # same bandwidth as kernel = "biweight"
h.ucv(x, kernel = "pow(1.5,2.5)")

plot(kernel.fun(kernel = kernel.pow(1.5, 2.5), deriv.order = 1L))
```

Index

∗ **bandwidth selection**
h.amise, 9
h.bcv, 11
h.ccv, 13
h.mcv, 15
h.mlcv, 17
h.pcv, 19
h.tcv, 24
h.ucv, 26
∗ **classes**
bandwidth-class, 5
kedd-classes, 28
∗ **datasets**
kedd-data, 29
∗ **density derivative**
dkde, 7
∗ **kernel**
kernel.fun, 32
kernel.pow, 34
∗ **nonparametric**
bandwidth-class, 5
dkde, 7
h.amise, 9
h.bcv, 11
h.ccv, 13
h.mcv, 15
h.mlcv, 17
h.pcv, 19
h.scv, 22
h.tcv, 24
h.ucv, 26
kernel.fun, 32
kernel.pow, 34
∗ **package**
kedd-package, 3
∗ **smooth**
bandwidth-class, 5
dkde, 7
h.amise, 9
h.bcv, 11
h.ccv, 13
h.mcv, 15
h.mlcv, 17
h.pcv, 19
h.scv, 22
h.tcv, 24
h.ucv, 26
$,bandwidth-method *(*bandwidth-class*)*, 5
$,dkde-method *(*kedd-classes*)*, 28
$,kconv-method *(*kedd-classes*)*, 28
$,kfun-method *(*kedd-classes*)*, 28

bandwidth, *3*, *5*, *9–19*, *21–27*, *29*
bandwidth-class, 5
bimodal *(*kedd-data*)*, 29

claw *(*kedd-data*)*, 29
Cpp_omp_threads, *3*

density, *8*
dkde, *3–6*, 7, *8*, *10*, *13*, *19*, *21*, *27–30*, *33–35*
dkde-class *(*kedd-classes*)*, 28

h.amise, *3–6*, *8*, 9, *13*, *15*, *17*, *19*, *21*, *25*, *27*, *35*
h.bcv, *3–6*, *8*, *10*, 11, *15*, *17*, *20, 21*, *23*, *25*, *27*
h.ccv, *3–6*, *10*, 13, *13*, *17*, *20, 21*, *25*, *27*, *35*
h.mcv, *3–6*, *10*, *13*, 15, *15*, *20, 21*, *25*, *27*, *35*
h.mlcv, *3–6*, *10*, 17, *21*
h.pcv, *3–6*, *8*, *10*, *13*, *15*, *17*, 19, *19*, *23*, *25*, *27*
h.scv, *3–6*, *8*, *10*, *13*, *15*, *17*, *19*, *21*, 22, *25*, *27*
h.tcv, *3–6*, *10*, *13*, *15*, *17*, *20, 21*, 24, *27*, *35*
h.ucv, *3–8*, *10*, *13*, *15*, *17*, *19–21*, *23*, *25*, 26, *30*, *33*, *35*

kconv, *31*
kconv-class *(*kedd-classes*)*, 28
kedd *(*kedd-package*)*, 3
kedd-classes, 28
kedd-data, 29

kedd-package, 3
kernel.conv, *3*, *5*, *12, 13*, *15–17*, *24–29*, 30, *33–35*
kernel.fun, *3*, *5*, *7, 8*, *12, 13*, *15–17*, *24–29*, *31*, 32, *34, 35*
kernel.pow, *3*, *5*, *7*, *9*, *11*, *14*, *16*, *18*, *20*, *24*, *26*, *31–33*, 34
kfun, *32, 33*
kfun-class *(*kedd-classes*)*, 28
kurtotic *(*kedd-data*)*, 29

lines,bandwidth-method *(*bandwidth-class*)*, 5

optimize, *11, 12*, *14*, *16*, *18*, *20–22*, *24*, *26*
outlier *(*kedd-data*)*, 29

plot,bandwidth-method *(*bandwidth-class*)*, 5
plot,dkde-method *(*kedd-classes*)*, 28
plot,kconv-method *(*kedd-classes*)*, 28
plot,kfun-method *(*kedd-classes*)*, 28

set.seed, *21*
show,bandwidth-method *(*bandwidth-class*)*, 5
show,dkde-method *(*kedd-classes*)*, 28
show,kconv-method *(*kedd-classes*)*, 28
show,kfun-method *(*kedd-classes*)*, 28

trimodal *(*kedd-data*)*, 29