Package 'ks'

September 20, 2024

Version 1.14.3

Date 2024-09-20

Title Kernel Smoothing

Maintainer Tarn Duong <tarn.duong@gmail.com>

Depends R (>= 2.10.0)

Imports FNN (>= 1.1), kernlab, KernSmooth (>= 2.22), Matrix, mclust, mgcv, multicool, mvtnorm (>= 1.0-0), pracma

Suggests geometry, MASS, misc3d (>= 0.4-0), oz, plot3D, rgl (>= 0.66)

Description Kernel smoothers for univariate and multivariate data, with comprehensive visualisation and bandwidth selection capabilities, including for densities, density derivatives, cumulative distributions, clustering, classification, density ridges, significant modal regions, and two-sample hypothesis tests. Chacon & Duong (2018) <doi:10.1201/9780429485572>.

License GPL-2 | GPL-3

URL https://www.mvstat.net/mvksa/

NeedsCompilation yes

Author Tarn Duong [aut, cre] (<https://orcid.org/0000-0002-1198-3482>), Matt Wand [ctb] (<https://orcid.org/0000-0003-2555-896X>), Jose Chacon [ctb], Artur Gramacki [ctb] (<https://orcid.org/0000-0002-1610-9743>)

Repository CRAN

Date/Publication 2024-09-20 17:00:01 UTC

Contents

ks-package *ks*

Description

Kernel smoothing for data from 1- to 6-dimensions.

Details

There are three main types of functions in this package:

- computing kernel estimators - these function names begin with 'k'
- computing bandwidth selectors - these begin with 'h' (1-d) or 'H' (>1-d)
- displaying kernel estimators - these begin with 'plot'.

The kernel used throughout is the normal (Gaussian) kernel K. For 1-d data, the bandwidth h is the standard deviation of the normal kernel, whereas for multivariate data, the bandwidth matrix $\mathbf{H}$ is the variance matrix.

–For kernel density estimation, `kde` computes

$$\hat{f}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{X}_i).$$

The bandwidth matrix $\mathbf{H}$ is a matrix of smoothing parameters and its choice is crucial for the performance of kernel estimators. For display, its `plot` method calls `plot.kde`.

–For kernel density estimation, there are several varieties of bandwidth selectors

- plug-in `hpi` (1-d); `Hpi`, `Hpi.diag` (2- to 6-d)
- least squares (or unbiased) cross validation (LSCV or UCV) `hlscv` (1-d); `Hlscv`, `Hlscv.diag` (2- to 6-d)
- biased cross validation (BCV) `Hbcv`, `Hbcv.diag` (2- to 6-d)
- smoothed cross validation (SCV) `hscv` (1-d); `Hscv`, `Hscv.diag` (2- to 6-d)
- normal scale `hns` (1-d); `Hns` (2- to 6-d).

–For kernel density support estimation, the main function is `ksupp` which is (the convex hull of)

$$\{\boldsymbol{x} : \hat{f}(\boldsymbol{x}) > \tau\}$$

for a suitable level τ. This is closely related to the τ-level set of $\hat{f}$.

–For truncated kernel density estimation, the main function is `kde.truncate`

$$\hat{f}(\boldsymbol{x}) \mathbf{1}\{\boldsymbol{x} \in \Omega\} / \int_{\Omega} \hat{f}(\boldsymbol{x})\, d\boldsymbol{x}$$

for a bounded data support Ω. The standard density estimate $\hat{f}$ is truncated and rescaled to give unit integral over Ω. Its `plot` method calls `plot.kde`.

–For boundary kernel density estimation where the kernel function is modified explicitly in the boundary region, the main function is kde.boundary

$$n^{-1} \sum_{i=1}^{n} K^*_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{X}_i)$$

for a boundary kernel K^*. Its plot method calls plot.kde.

–For variable kernel density estimation where the bandwidth is not a constant matrix, the main functions are kde.balloon

$$\hat{f}_{\mathrm{ball}}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} K_{\mathbf{H}(\boldsymbol{x})}(\boldsymbol{x} - \boldsymbol{X}_i)$$

and kde.sp

$$\hat{f}_{\mathrm{SP}}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} K_{\mathbf{H}(\boldsymbol{X}_i)}(\boldsymbol{x} - \boldsymbol{X}_i).$$

For the balloon estimation $\hat{f}_{\mathrm{ball}}$ the bandwidth varies with the estimation point $\boldsymbol{x}$, whereas for the sample point estimation $\hat{f}_{\mathrm{SP}}$ the bandwidth varies with the data point $\boldsymbol{X}_i, i = 1, \ldots, n$. Their plot methods call plot.kde. The bandwidth selectors for kde.balloon are based on the normal scale bandwidth Hns(,deriv.order=2) via the MSE minimal formula, and for kde.SP on Hns(,deriv.order=4) via the Abramson formula.

–For kernel density derivative estimation, the main function is kdde

$$\mathsf{D}^{\otimes r} \hat{f}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} \mathsf{D}^{\otimes r} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{X}_i).$$

The bandwidth selectors are a modified subset of those for kde, i.e. Hlscv, Hns, Hpi, Hscv with deriv.order>0. Its plot method is plot.kdde for plotting each partial derivative singly.

–For kernel summary curvature estimation, the main function is kcurv

$$\hat{s}(\boldsymbol{x}) = -\mathbf{1}\{\mathsf{D}^2 \hat{f}(\boldsymbol{x}) < 0\} \mathrm{abs}(|\mathsf{D}^2 \hat{f}(\boldsymbol{x})|)$$

where $\mathsf{D}^2 \hat{f}(\boldsymbol{x})$ is the kernel Hessian matrix estimate. It has the same structure as a kernel density estimate so its plot method calls plot.kde.

–For kernel discriminant analysis, the main function is kda which computes density estimates for each the groups in the training data, and the discriminant surface. Its plot method is plot.kda. The wrapper function hkda, Hkda computes bandwidths for each group in the training data for kde, e.g. hpi, Hpi.

–For kernel functional estimation, the main function is kfe which computes the r-th order integrated density functional

$$\hat{\boldsymbol{\psi}}_r = n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathsf{D}^{\otimes r} K_{\mathbf{H}}(\boldsymbol{X}_i - \boldsymbol{X}_j).$$

The plug-in selectors are hpi.kfe (1-d), Hpi.kfe (2- to 6-d). Kernel functional estimates are usually not required to computed directly by the user, but only within other functions in the package.

–For kernel-based 2-sample testing, the main function is `kde.test` which computes the integrated L_2 distance between the two density estimates as the test statistic, comprising a linear combination of 0-th order kernel functional estimates:

$$\hat{T} = \hat{\psi}_{0,1} + \hat{\psi}_{0,2} - (\hat{\psi}_{0,12} + \hat{\psi}_{0,21}),$$

and the corresponding p-value. The ψ are zero order kernel functional estimates with the subscripts indicating that 1 = sample 1 only, 2 = sample 2 only, and 12, 21 = samples 1 and 2. The bandwidth selectors are `hpi.kfe`, `Hpi.kfe` with `deriv.order=0`.

–For kernel-based local 2-sample testing, the main function is `kde.local.test` which computes the squared distance between the two density estimates as the test statistic

$$\hat{U}(\boldsymbol{x}) = [\hat{f}_1(\boldsymbol{x}) - \hat{f}_2(\boldsymbol{x})]^2$$

and the corresponding local p-values. The bandwidth selectors are those used with `kde`, e.g. `hpi`, `Hpi`.

–For kernel cumulative distribution function estimation, the main function is `kcde`

$$\hat{F}(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} \mathcal{K}_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{X}_i)$$

where $\mathcal{K}$ is the integrated kernel. The bandwidth selectors are `hpi.kcde`, `Hpi.kcde`. Its `plot` method is `plot.kcde`. There exist analogous functions for the survival function $\hat{\bar{F}}$.

–For kernel estimation of a ROC (receiver operating characteristic) curve to compare two samples from $\hat{F}_1, \hat{F}_2$, the main function is `kroc`

$$\{\hat{F}_{\hat{Y}_1}(z), \hat{F}_{\hat{Y}_2}(z)\}$$

based on the cumulative distribution functions of $\hat{Y}_j = \hat{\bar{F}}_1(\boldsymbol{X}_j), j = 1, 2$.

The bandwidth selectors are those used with `kcde`, e.g. `hpi.kcde`, `Hpi.kcde` for $\hat{F}_{\hat{Y}_j}, \hat{\bar{F}}_1$. Its `plot` method is `plot.kroc`.

–For kernel estimation of a copula, the main function is `kcopula`

$$\hat{C}(\boldsymbol{z}) = \hat{F}(\hat{F}_1^{-1}(z_1), \ldots, \hat{F}_d^{-1}(z_d))$$

where $\hat{F}_j^{-1}(z_j)$ is the z_j-th quantile of of the j-th marginal distribution $\hat{F}_j$. The bandwidth selectors are those used with `kcde` for $\hat{F}, \hat{F}_j$. Its `plot` method is `plot.kcde`.

–For kernel mean shift clustering, the main function is `kms`. The mean shift recurrence relation of the candidate point $\boldsymbol{x}$

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \mathbf{H}\mathsf{D}\hat{f}(\boldsymbol{x}_j)/\hat{f}(\boldsymbol{x}_j),$$

where $j \geq 0$ and $\boldsymbol{x}_0 = \boldsymbol{x}$, is iterated until $\boldsymbol{x}$ converges to its local mode in the density estimate $\hat{f}$ by following the density gradient ascent paths. This mode determines the cluster label for $\boldsymbol{x}$. The bandwidth selectors are those used with `kdde(,deriv.order=1)`.

–For kernel density ridge estimation, the main function is `kdr`. The kernel density ridge recurrence relation of the candidate point $\boldsymbol{x}$

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \mathbf{U}_{(d-1)}(\boldsymbol{x}_j)\mathbf{U}_{(d-1)}(\boldsymbol{x}_j)^T\mathbf{H}\mathsf{D}\hat{f}(\boldsymbol{x}_j)/\hat{f}(\boldsymbol{x}_j),$$

where $j \geq 0$, $\boldsymbol{x}_0 = \boldsymbol{x}$ and $\mathbf{U}_{(d-1)}$ is the 1-dimensional projected density gradient, is iterated until $\boldsymbol{x}$ converges to the ridge in the density estimate. The bandwidth selectors are those used with `kdde(,deriv.order=2)`.

– For kernel feature significance, the main function `kfs`. The hypothesis test at a point $\boldsymbol{x}$ is $H_0(\boldsymbol{x})$: $\mathsf{H}f(\boldsymbol{x}) < 0$, i.e. the density Hessian matrix $\mathsf{H}f(\boldsymbol{x})$ is negative definite. The test statistic is

$$W(\boldsymbol{x}) = \|\mathbf{S}(\boldsymbol{x})^{-1/2}\text{vech } \mathsf{H}\hat{f}(\boldsymbol{x})\|^2$$

where $\mathsf{H}\hat{f}$ is the Hessian estimate, vech is the vector-half operator, and $\mathbf{S}$ is an estimate of the null variance. $W(\boldsymbol{x})$ is approximately χ^2 distributed with $d(d+1)/2$ degrees of freedom. If $H_0(\boldsymbol{x})$ is rejected, then $\boldsymbol{x}$ belongs to a significant modal region. The bandwidth selectors are those used with `kdde(,deriv.order=2)`. Its `plot` method is `plot.kfs`.

–For deconvolution density estimation, the main function is `kdcde`. A weighted kernel density estimation with the contaminated data $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$,

$$\hat{f}_w(\boldsymbol{x}) = n^{-1} \sum_{i=1}^{n} \alpha_i K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{W}_i),$$

is utilised, where the weights $\alpha_1, \ldots, \alpha_n$ are chosen via a quadratic optimisation involving the error variance and the regularisation parameter. The bandwidth selectors are those used with `kde`.

–Binned kernel estimation is an approximation to the exact kernel estimation and is available for d=1, 2, 3, 4. This makes kernel estimators feasible for large samples.

–For an overview of this package with 2-d density estimation, see `vignette("kde")`.

–For **ks** $\geq$ 1.11.1, the **misc3d** and **rgl** (3-d plot), **oz** (Australian map) packages, and for **ks** $\geq$ 1.14.2, the **plot3D** (3-d plot) package, have been moved from Depends to Suggests. This was done to allow **ks** to be installed on systems where these latter graphical-based packages can't be installed. Furthermore, since the future of OpenGL in R is not certain, **plot3D** becomes the default for 3D plotting for **ks** $\geq$ 1.12.0. RGL plots are still supported though these may be deprecated in the future.

Author(s)

Tarn Duong for most of the package. M. P. Wand for the binned estimation, univariate plug-in selector and univariate density derivative estimator code. J. E. Chacon for the unconstrained pilot functional estimation and fast implementation of derivative-based estimation code. A. and J. Gramacki for the binned estimation for unconstrained bandwidth matrices.

References

Bowman, A. & Azzalini, A. (1997) *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, Oxford.

Chacon, J.E. & Duong, T. (2018) *Multivariate Kernel Smoothing and Its Applications*. Chapman & Hall/CRC, Boca Raton.

Duong, T. (2004) *Bandwidth Matrices for Multivariate Kernel Density Estimation.* Ph.D. Thesis, University of Western Australia.

Scott, D.W. (2015) *Multivariate Density Estimation: Theory, Practice, and Visualization (2nd edn)*. John Wiley & Sons, New York.

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

Simonoff, J. S. (1996) *Smoothing Methods in Statistics*. Springer-Verlag, New York.

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

See Also

feature, **sm**, **KernSmooth**

`air` *Air quality measurements in an underground train station*

Description

This data set contains the hourly mean air quality measurements from 01 January 2013 to 31 December 2016 in the Chatelet underground train station in the Paris metro.

Usage

```
data(air)
```

Format

A matrix with 35039 rows and 8 columns. Each row corresponds to an hourly measurement. The first column is the date (yyyy-mm-dd), the second is the time (hh:mm), the third is the nitric oxide NO concentration (g/m3), the fourth is the nitrogen dioxide NO_2 concentration (g/m3), the fifth is the concentration of particulate matter less than 10 microns PM10 (ppm), the sixth is the carbon dioxide concentration CO_2 (g/m3), the seventh is the temperature (degrees Celsius), the eighth is the relative humidity (percentage).

Source

RATP (2016) Qualite de l'air mesuree dans la station Chatelet, https://data.iledefrance.fr/explore/dataset/qualite-de-l-air-mesuree-dans-la-station-chatelet-rer-a. Regie autonome des transports parisiens - Departement Developpement, Innovation et Territoires. Accessed 2017-09-27.

binning *Linear binning for multivariate data*

Description

Linear binning for 1- to 4-dimensional data.

Usage

```
binning(x, H, h, bgridsize, xmin, xmax, supp=3.7, w, gridtype="linear")
```

Arguments

x	matrix of data values
H, h	bandwidth matrix, scalar bandwidth
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal is [-supp,supp]
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
gridtype	not yet implemented

Details

For **ks** $\geq$ 1.10.0, binning is available for unconstrained (non-diagonal) bandwidth matrices. Code is used courtesy of A. & J. Gramacki, and M.P. Wand. Default bgridsize are d=1: 401; d=2: rep(151, 2); d=3: rep(51, 3); d=4: rep(21, 4).

Value

Returns a list with 2 fields

counts	linear binning counts
eval.points	vector (d=1) or list (d>=2) of grid points in each dimension

References

Gramacki, A. & Gramacki, J. (2016) FFT-based fast computation of multivariate kernel estimators with unconstrained bandwidth matrices. *Journal of Computational & Graphical Statistics*, **26**, 459-462.

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

Examples

```
data(unicef)
ubinned <- binning(x=unicef)
```

cardio *Foetal cardiotocograms*

Description

This data set contains the cardiotocographic measurements from healthy, suspect and pathological foetuses.

Usage

```
data(cardio)
```

Format

A matrix with 2126 rows and 8 columns. Each row corresponds to a foetal cardiotocogram. The class label for the foetal state is the last column: N = normal, S = suspect, P = pathological. Details for all variables are found in the link below.

Source

Lichman, M. (2013) UCI Machine learning repository: cardiotocography data set. University of California, Irvine, School of Information and Computer Sciences. Accessed 2017-05-18.

contour *Contour functions*

Description

Contour levels and sizes.

Usage

```
contourLevels(x, ...)
## S3 method for class 'kde'
 contourLevels(x, prob, cont, nlevels=5, approx=TRUE, ...)
## S3 method for class 'kda'
 contourLevels(x, prob, cont, nlevels=5, approx=TRUE, ...)
## S3 method for class 'kdde'
contourLevels(x, prob, cont, nlevels=5, approx=TRUE, which.deriv.ind=1, ...)

contourSizes(x, abs.cont, cont=c(25,50,75), approx=TRUE)
contourProbs(x, abs.cont, cont=c(25,50,75), approx=TRUE)
```

Arguments

x	object of class kde, kdde or kda
prob	vector of probabilities corresponding to highest density regions
cont	vector of percentages which correspond to the complement of prob
abs.cont	vector of absolute contour levels
nlevels	number of pretty contour levels
approx	flag to compute approximate contour levels. Default is TRUE.
which.deriv.ind	partial derivative index. Default is 1.
...	other parameters

Details

–For contourLevels, the most straightforward is to specify prob. The heights of the corresponding highest density region with probability prob are computed. The cont parameter here is consistent with cont parameter from plot.kde, plot.kdde, and plot.kda i.e. cont=(1-prob)*100%. If both prob and cont are missing then a pretty set of nlevels contours are computed.

–For contourSizes, the length, area, volume etc. and for contourProbs, the probability, are approximated by Riemann sums. These are rough approximations and depend highly on the estimation grid, and so should be interpreted carefully.

If approx=FALSE, then the exact KDE is computed. Otherwise it is interpolated from an existing KDE grid: this can dramatically reduce computation time for large data sets.

Value

–For contourLevels, for kde objects, returns vector of heights. For kda objects, returns a list of vectors, one for each training group. For kdde objects, returns a matrix of vectors, one row for each partial derivative.

–For contourSizes, returns an approximation of the Lebesgue measure of level set, i.e. length (d=1), area (d=2), volume (d=3), hyper-volume (d>4).

–For contourProbs, returns an approximation of the probability measure of level set.

See Also

contour, contourLines

Examples

```
set.seed(8192)
x <- rmvnorm.mixt(n=1000, mus=c(0,0), Sigmas=diag(2), props=1)
fhat <- kde(x=x, binned=TRUE)
contourLevels(fhat, cont=c(75, 50, 25))
contourProbs(fhat, abs.cont=contourLevels(fhat, cont=50))
  ## compare approx prob with target prob=0.5
contourSizes(fhat, cont=25, approx=TRUE)
   ## compare to approx circle of radius=0.75 with area=1.77
```

grevillea *Geographical locations of grevillea plants*

Description

This data set contains the geographical locations of the specimens of *Grevillea uncinulata*, more commonly known as the Hook leaf grevillea, which is an endemic floral species to south Western Australia. This region is one of the 25 'biodiversity hotspots' which are 'areas featuring exceptional concentrations of endemic species and experiencing exceptional loss of habitat'.

Usage

```
data(grevillea)
```

Format

A matrix with 222 rows and 2 columns. Each row corresponds to an observed plant. The first column is the longitude (decimal degrees), the second is the latitude (decimal degrees).

Source

CSIRO (2016) Atlas of Living Australia: *Grevillea uncinulata Diels*, https://bie.ala.org.au/species/https://id.biodiversity.org.au/node/apni/2895039. Commonwealth Scientific and Industrial Research Organisation. Accessed 2016-03-11.

Hbcv *Biased cross-validation (BCV) bandwidth matrix selector for bivariate data*

Description

BCV bandwidth matrix for bivariate data.

Usage

```
Hbcv(x, whichbcv=1, Hstart, binned=FALSE, amise=FALSE, verbose=FALSE)
Hbcv.diag(x, whichbcv=1, Hstart, binned=FALSE, amise=FALSE, verbose=FALSE)
```

Arguments

x	matrix of data values
whichbcv	1 = BCV1, 2 = BCV2. See details below.
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation. Default is FALSE.
amise	flag to return the minimal BCV value. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.

Details

Use `Hbcv` for unconstrained bandwidth matrices and `Hbcv.diag` for diagonal bandwidth matrices. These selectors are only available for bivariate data. Two types of BCV criteria are considered here. They are known as BCV1 and BCV2, from Sain, Baggerly & Scott (1994) and only differ slightly. These BCV surfaces can have multiple minima and so it can be quite difficult to locate the most appropriate minimum. Some times, there can be no local minimum at all so there may be no finite BCV selector.

For details about the advanced options for `binned`, `Hstart`, see `Hpi`.

Value

BCV bandwidth matrix. If `amise=TRUE` then the minimal BCV value is returned too.

References

Sain, S.R, Baggerly, K.A. & Scott, D.W. (1994) Cross-validation of multivariate densities. *Journal of the American Statistical Association*, **82**, 1131-1146.

See Also

`Hlscv`, `Hpi`, `Hscv`

Examples

```
data(unicef)
Hbcv(unicef)
Hbcv.diag(unicef)
```

`histde` *Histogram density estimate*

Description

Histogram density estimate for 1- and 2-dimensional data.

Usage

```
histde(x, binw, xmin, xmax, adj=0)

## S3 method for class 'histde'
predict(object, ..., x)
```

Arguments

x	matrix of data values
binw	(vector) of binwidths
xmin, xmax	vector of minimum/maximum values for grid
adj	displacement of default anchor point, in percentage of 1 bin
object	object of class histde
...	other parameters

Details

If binw is missing, the default binwidth is $\hat{b}_i = 2 \cdot 3^{1/(d+2)} \pi^{d/(2d+4)} S_i n^{-1/(d+2)}$, the normal scale selector.

If xmin is missing then it defaults to the data minimum. If xmax is missing then it defaults to the data maximum.

Value

A histogram density estimate is an object of class histde which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	density estimate at eval.points
binw	(vector of) bandwidths
nbin	(vector of) number of bins
names	variable names

See Also

plot.histde

Examples

```
## positive data example
set.seed(8192)
x <- 2^rnorm(100)
fhat <- histde(x=x)
plot(fhat, border=6)
points(c(0.5, 1), predict(fhat, x=c(0.5, 1)))

## large data example on a non-default grid
set.seed(8192)
x <- rmvnorm.mixt(10000, mus=c(0,0), Sigmas=invvech(c(1,0.8,1)))
fhat <- histde(x=x, xmin=c(-5,-5), xmax=c(5,5))
plot(fhat)

## See other examples in ? plot.histde
```

Hlscv *Least-squares cross-validation (LSCV) bandwidth matrix selector for multivariate data*

Description

LSCV bandwidth for 1- to 6-dimensional data

Usage

```
Hlscv(x, Hstart, binned, bgridsize, amise=FALSE, deriv.order=0,
      verbose=FALSE, optim.fun="optim", trunc)
Hlscv.diag(x, Hstart, binned, bgridsize, amise=FALSE, deriv.order=0,
      verbose=FALSE, optim.fun="optim", trunc)
hlscv(x, binned=TRUE, bgridsize, amise=FALSE, deriv.order=0, bw.ucv=TRUE)
Hucv(...)
Hucv.diag(...)
hucv(...)
```

Arguments

x	vector or matrix of data values
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes
amise	flag to return the minimal LSCV value. Default is FALSE.
deriv.order	derivative order
verbose	flag to print out progress information. Default is FALSE.
optim.fun	optimiser function: one of nlm or optim
trunc	parameter to control truncation for numerical optimisation. Default is 4 for density.deriv>0, otherwise no truncation. For details see below.
bw.ucv	flag to use stats::bw.ucv as minimiser function. Default is TRUE.
...	parameters as above

Details

hlscv is the univariate LSCV selector of Bowman (1984) and Rudemo (1982). Hlscv is a multivariate generalisation of this. Use Hlscv for unconstrained bandwidth matrices and Hlscv.diag for diagonal bandwidth matrices. Hucv, Hucv.diag and hucv are aliases with UCV (unbiased cross validation) instead of LSCV.

For **ks** $\geq$ 1.13.0, the default minimiser in hlscv is based on the UCV minimiser stats::bw.ucv. To reproduce prior behaviour, set bw.ucv=FALSE.

Truncation of the parameter space is usually required for the LSCV selector, for r > 0, to find a reasonable solution to the numerical optimisation. If a candidate matrix H is such that det(H) is not

in [1/trunc, trunc]*det(H0) or abs(LSCV(H)) > trunc*abs(LSCV0) then the LSCV(H) is reset to LSCV0 where H0=Hns(x) and LSCV0=LSCV(H0).

For details about the advanced options for binned,Hstart,optim.fun, see Hpi.

Value

LSCV bandwidth. If amise=TRUE then the minimal LSCV value is returned too.

References

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

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

See Also

Hbcv, Hpi, Hscv

Examples

```
data(forbes, package="MASS")
Hlscv(forbes)
hlscv(forbes$bp)
```

Hnm *Normal mixture bandwidth*

Description

Normal mixture bandwidth.

Usage

```
Hnm(x, deriv.order=0, G=1:9, subset.ind, mise.flag=FALSE, verbose, ...)
Hnm.diag(x, deriv.order=0, G=1:9, subset.ind, mise.flag=FALSE, verbose, ...)
hnm(x, deriv.order=0, G=1:9, subset.ind, mise.flag=FALSE, verbose, ... )
```

Arguments

x	vector/matrix of data values
deriv.order	derivative order
G	range of number of mixture components
subset.ind	index vector of subset of x for fitting
mise.flag	flag to use MISE or AMISE minimisation. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.
...	other parameters for Mclust

Details

The normal mixture fit is provided by the `Mclust` function in the **mclust** package. `Hnm` is then `Hmise.mixt` (if `mise.flag=TRUE`) or `Hamise.mixt` (if `mise.flag=FALSE`) with these fitted normal mixture parameters. Likewise for `Hnm.diag`, `hnm`.

Value

Normal mixture bandwidth. If `mise=TRUE` then the minimal MISE value is returned too.

References

Cwik, J. & Koronacki, J. (1997). A combined adaptive-mixtures/plug-in estimator of multivariate probability densities. *Computational Statistics and Data Analysis*, **26**, 199-218.

See Also

Hmise.mixt, Hamise.mixt

Examples

```
data(unicef)
Hnm(unicef)
```

Hns *Normal scale bandwidth*

Description

Normal scale bandwidth.

Usage

```
Hns(x, deriv.order=0)
Hns.diag(x)
hns(x, deriv.order=0)
Hns.kcde(x)
hns.kcde(x)
```

Arguments

`x`	vector/matrix of data values
`deriv.order`	derivative order

Details

`Hns` is equal to `(4/(n*(d+2*r+2)))^(2/(d+2*r+4))*var(x)`, n = sample size, d = dimension of data, r = derivative order. `hns` is the analogue of `Hns` for 1-d data. These can be used for density (derivative) estimators `kde`, `kdde`. The equivalents for distribution estimators `kcde` are `Hns.kcde` and `hns.cde`.

Value

Normal scale bandwidth.

References

Chacon J.E., Duong, T. & Wand, M.P. (2011). Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*, **21**, 807-840.

Examples

```
data(forbes, package="MASS")
Hns(forbes, deriv.order=2)
hns(forbes$bp, deriv.order=2)
```

`Hpi` *Plug-in bandwidth selector*

Description

Plug-in bandwidth for for 1- to 6-dimensional data.

Usage

```
Hpi(x, nstage=2, pilot, pre="sphere", Hstart, binned, bgridsize,
   amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
Hpi.diag(x, nstage=2, pilot, pre="scale", Hstart, binned, bgridsize,
   amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
hpi(x, nstage=2, binned=TRUE, bgridsize, deriv.order=0)
```

Arguments

`x`	vector or matrix of data values
`nstage`	number of stages in the plug-in bandwidth selector (1 or 2)
`pilot`	"amse" = AMSE pilot bandwidths "samse" = single SAMSE pilot bandwidth "unconstr" = single unconstrained pilot bandwidth "dscalar" = single pilot bandwidth for deriv.order >= 0 "dunconstr" = single unconstrained pilot bandwidth for deriv.order >= 0
`pre`	"scale" = `pre.scale`, "sphere" = `pre.sphere`
`Hstart`	initial bandwidth matrix, used in numerical optimisation
`binned`	flag for binned kernel estimation
`bgridsize`	vector of binning grid sizes
`amise`	flag to return the minimal scaled PI value
`deriv.order`	derivative order
`verbose`	flag to print out progress information. Default is FALSE.
`optim.fun`	optimiser function: one of `nlm` or `optim`

Details

`hpi(,deriv.order=0)` is the univariate plug-in selector of Wand & Jones (1994), i.e. it is exactly the same as **KernSmooth**'s `dpik`. For deriv.order>0, the formula is taken from Wand & Jones (1995). `Hpi` is a multivariate generalisation of this. Use `Hpi` for unconstrained bandwidth matrices and `Hpi.diag` for diagonal bandwidth matrices.

The default pilot is `"samse"` for d=2,r=0, and `"dscalar"` otherwise. For AMSE pilot bandwidths, see Wand & Jones (1994). For SAMSE pilot bandwidths, see Duong & Hazelton (2003). The latter is a modification of the former, in order to remove any possible problems with non-positive definiteness. Unconstrained and higher order derivative pilot bandwidths are from Chacon & Duong (2010).

For d=1, 2, 3, 4 and `binned=TRUE`, estimates are computed over a binning grid defined by `bgridsize`. Otherwise it's computed exactly. If `Hstart` is not given then it defaults to `Hns(x)`.

For **ks** $\geq$ 1.11.1, the default optimisation function is `optim.fun="optim"`. To reinstate the previous functionality, use `optim.fun="nlm"`.

Value

Plug-in bandwidth. If `amise=TRUE` then the minimal scaled PI value is returned too.

References

Chacon, J.E. & Duong, T. (2010) Multivariate plug-in bandwidth selection with unconstrained pilot matrices. *Test*, **19**, 375-398.

Duong, T. & Hazelton, M.L. (2003) Plug-in bandwidth matrices for bivariate kernel density estimation. *Journal of Nonparametric Statistics*, **15**, 17-30.

Sheather, S.J. & Jones, M.C. (1991) A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society Series B*, **53**, 683-690.

Wand, M.P. & Jones, M.C. (1994) Multivariate plug-in bandwidth selection. *Computational Statistics*, **9**, 97-116.

See Also

`Hbcv`, `Hlscv`, `Hscv`

Examples

```
data(unicef)
Hpi(unicef, pilot="dscalar")
hpi(unicef[,1])
```

hsct *Haematopoietic stem cell transplant*

Description

This data set contains the haematopoietic stem cell transplant (HSCT) measurements obtained by a flow cytometer from mouse subjects. A flow cytometer measures the spectra of fluorescent signals from biological cell samples to study their properties.

Usage

```
data(hsct)
```

Format

A matrix with 39128 rows and 6 columns. The first column is the FITC-CD45.1 fluorescence (0-1023), the second is the PE-Ly65/Mac1 fluorescence (0-1023), the third is the PI-LiveDead fluorescence (0-1023), the fourth is the APC-CD45.2 fluorescence (0-1023), the fifth is the class label of the cell type (1, 2, 3, 4, 5), the sixth the mouse subject number (5, 6, 9, 12).

Source

Aghaeepour, N., Finak, G., The FlowCAP Consortium, The DREAM Consortium, Hoos, H., Mosmann, T. R., Brinkman, R., Gottardo, R. & Scheuermann, R. H. (2013) Critical assessment of automated flow cytometry data analysis techniques, *Nature Methods* **10**, 228-238.

Hscv *Smoothed cross-validation (SCV) bandwidth selector*

Description

SCV bandwidth for 1- to 6-dimensional data.

Usage

```
Hscv(x, nstage=2, pre="sphere", pilot, Hstart, binned,
     bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
Hscv.diag(x, nstage=2, pre="scale", pilot, Hstart, binned,
     bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
hscv(x, nstage=2, binned=TRUE, bgridsize, plot=FALSE)
```

Arguments

x	vector or matrix of data values
pre	"scale" = pre.scale, "sphere" = pre.sphere
pilot	"amse" = AMSE pilot bandwidths "samse" = single SAMSE pilot bandwidth "unconstr" = single unconstrained pilot bandwidth "dscalar" = single pilot bandwidth for deriv.order>0 "dunconstr" = single unconstrained pilot bandwidth for deriv.order>0
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes
amise	flag to return the minimal scaled SCV value. Default is FALSE.
deriv.order	derivative order
verbose	flag to print out progress information. Default is FALSE.
optim.fun	optimiser function: one of nlm or optim
nstage	number of stages in the SCV bandwidth selector (1 or 2)
plot	flag to display plot of SCV(h) vs h (1-d only). Default is FALSE.

Details

hscv is the univariate SCV selector of Jones, Marron & Park (1991). Hscv is a multivariate generalisation of this, see Duong & Hazelton (2005). Use Hscv for unconstrained bandwidth matrices and Hscv.diag for diagonal bandwidth matrices.

The default pilot is "samse" for d=2, r=0, and "dscalar" otherwise. For SAMSE pilot bandwidths, see Duong & Hazelton (2005). Unconstrained and higher order derivative pilot bandwidths are from Chacon & Duong (2011).

For d=1, the selector hscv is not always stable for large sample sizes with binning. Examine the plot from hscv(, plot=TRUE) to determine the appropriate smoothness of the SCV function. Any non-smoothness is due to the discretised nature of binned estimation.

For details about the advanced options for binned, Hstart, optim.fun, see Hpi.

Value

SCV bandwidth. If amise=TRUE then the minimal scaled SCV value is returned too.

References

Chacon, J.E. & Duong, T. (2011) Unconstrained pilot selectors for smoothed cross validation. *Australian & New Zealand Journal of Statistics*, **53**, 331-351.

Duong, T. & Hazelton, M.L. (2005) Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, **32**, 485-506.

Jones, M.C., Marron, J.S. & Park, B.U. (1991) A simple root n bandwidth selector. *Annals of Statistics*, **19**, 1919-1932.

See Also

Hbcv, Hlscv, Hpi

Examples

```
data(unicef)
Hscv(unicef)
hscv(unicef[,1])
```

---

ise.mixt *Squared error bandwidth matrix selectors for normal mixture densities*

Description

The global errors ISE (Integrated Squared Error), MISE (Mean Integrated Squared Error) and the AMISE (Asymptotic Mean Integrated Squared Error) for 1- to 6-dimensional data. Normal mixture densities have closed form expressions for the MISE and AMISE. So in these cases, we can numerically minimise these criteria to find MISE- and AMISE-optimal matrices.

Usage

```
Hamise.mixt(mus, Sigmas, props, samp, Hstart, deriv.order=0)
Hmise.mixt(mus, Sigmas, props, samp, Hstart, deriv.order=0)
Hamise.mixt.diag(mus, Sigmas, props, samp, Hstart, deriv.order=0)
Hmise.mixt.diag(mus, Sigmas, props, samp, Hstart, deriv.order=0)
hamise.mixt(mus, sigmas, props, samp, hstart, deriv.order=0)
hmise.mixt(mus, sigmas, props, samp, hstart, deriv.order=0)
amise.mixt(H, mus, Sigmas, props, samp, h, sigmas, deriv.order=0)
ise.mixt(x, H, mus, Sigmas, props, h, sigmas, deriv.order=0, binned=FALSE,
         bgridsize)
mise.mixt(H, mus, Sigmas, props, samp, h, sigmas, deriv.order=0)
```

Arguments

mus	(stacked) matrix of mean vectors (>1-d), vector of means (1-d)
Sigmas, sigmas	(stacked) matrix of variance matrices (>1-d), vector of standard deviations (1-d)
props	vector of mixing proportions
samp	sample size
Hstart, hstart	initial bandwidth (matrix), used in numerical optimisation
deriv.order	derivative order
x	matrix of data values
H, h	bandwidth (matrix)
binned	flag for binned kernel estimation. Default is FALSE.
bgridsize	vector of binning grid sizes

Details

ISE is a random variable that depends on the data x. MISE and AMISE are non-random and don't depend on the data. For normal mixture densities, ISE, MISE and AMISE have exact formulas for all dimensions.

Value

MISE- or AMISE-optimal bandwidth matrix. ISE, MISE or AMISE value.

References

Chacon J.E., Duong, T. & Wand, M.P. (2011). Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*, **21**, 807-840.

Examples

```
x <- rmvnorm.mixt(100)
Hamise.mixt(samp=nrow(x), mus=rep(0,2), Sigmas=var(x), props=1, deriv.order=1)
```

---

kcde *Kernel cumulative distribution/survival function estimate*

Description

Kernel cumulative distribution/survival function estimate for 1- to 3-dimensional data.

Usage

```
kcde(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points, binned,
  bgridsize, positive=FALSE, adj.positive, w, verbose=FALSE,
  tail.flag="lower.tail")
Hpi.kcde(x, nstage=2, pilot, Hstart, binned, bgridsize, amise=FALSE,
  verbose=FALSE, optim.fun="optim", pre=TRUE)
Hpi.diag.kcde(x, nstage=2, pilot, Hstart, binned, bgridsize, amise=FALSE,
  verbose=FALSE, optim.fun="optim", pre=TRUE)
hpi.kcde(x, nstage=2, binned, amise=FALSE)

## S3 method for class 'kcde'
predict(object, ..., x)
```

Arguments

x	matrix of data values
H, h	bandwidth matrix/scalar bandwidth. If these are missing, then Hpi.kcde or hpi.kcde is called by default.
gridsize	vector of number of grid points

gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
positive	flag if 1-d data are positive. Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	not yet implemented
verbose	flag to print out progress information. Default is FALSE.
tail.flag	"lower.tail" = cumulative distribution, "upper.tail" = survival function
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"dscalar" = single pilot bandwidth (default for Hpi.diag.kcde "dunconstr" = single unconstrained pilot bandwidth (default for Hpi.kcde
Hstart	initial bandwidth matrix, used in numerical optimisation
amise	flag to return the minimal scaled PI value
optim.fun	optimiser function: one of nlm or optim
pre	flag for pre-scaling data. Default is TRUE.
object	object of class kcde
...	other parameters

Details

If tail.flag="lower.tail" then the cumulative distribution function $\mathrm{Pr}(\boldsymbol{X} \leq \boldsymbol{x})$ is estimated, otherwise if tail.flag="upper.tail", it is the survival function $\mathrm{Pr}(\boldsymbol{X} > \boldsymbol{x})$. For $d > 1$, $\mathrm{Pr}(\boldsymbol{X} \leq \boldsymbol{x}) \neq 1 - \mathrm{Pr}(\boldsymbol{X} > \boldsymbol{x})$.

If the bandwidth H is missing in kcde, then the default bandwidth is the plug-in selector Hpi.kcde. Likewise for missing h. No pre-scaling/pre-sphering is used since the Hpi.kcde is not invariant to translation/dilation.

The effective support, binning, grid size, grid range, positive, optimisation function parameters are the same as kde.

Value

A kernel cumulative distribution estimate is an object of class kcde which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	cumulative distribution/survival function estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"

gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	vector of weights
tail	"lower.tail"=cumulative distribution, "upper.tail"=survival function

References

Duong, T. (2016) Non-parametric smoothed estimation of multivariate cumulative distribution and survival functions, and receiver operating characteristic curves. *Journal of the Korean Statistical Society*, **45**, 33-50.

See Also

kde, plot.kcde

Examples

```
data(iris)
Fhat <- kcde(iris[,1:2])
predict(Fhat, x=as.matrix(iris[,1:2]))

## See other examples in ? plot.kcde
```

---

kcopula *Kernel copula (density) estimate*

Description

Kernel copula and copula density estimator for 2-dimensional data.

Usage

```
kcopula(x, H, hs, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
  binned, bgridsize, w, marginal="kernel", verbose=FALSE)
kcopula.de(x, H, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
  binned, bgridsize, w, compute.cont=TRUE, approx.cont=TRUE,
  marginal="kernel", boundary.supp, boundary.kernel="beta", verbose=FALSE)
```

Arguments

x	matrix of data values
H, hs	bandwidth matrix. If these are missing, Hpi.kcde/Hpi or hpi.kcde/hpi is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented

xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	matrix of points at which estimate is evaluated
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
marginal	"kernel" = kernel cdf or "empirical" = empirical cdf to calculate pseudo-uniform values. Default is "kernel".
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
boundary.supp	effective support for boundary region
boundary.kernel	"beta" = beta boundary kernel, "linear" = linear boundary kernel
verbose	flag to print out progress information. Default is FALSE.

Details

For kernel copula estimates, a transformation approach is used to account for the boundary effects. If H is missing, the default is Hpi.kcde; if hs are missing, the default is hpi.kcde.

For kernel copula density estimates, for those points which are in the interior region, the usual kernel density estimator (kde) is used. For those points in the boundary region, a product beta kernel based on the boundary corrected univariate beta kernel of Chen (1999) is used (kde.boundary). If H is missing, the default is Hpi.kcde; if hs are missing, the default is hpi.

The effective support, binning, grid size, grid range parameters are the same as for kde.

Value

A kernel copula estimate, output from kcopula, is an object of class kcopula. A kernel copula density estimate, output from kcopula.de, is an object of class kde. These two classes of objects have the same fields as kcde and kde objects respectively, except for

x	pseudo-uniform data points
x.orig	data points - same as input
marginal	marginal function used to compute pseudo-uniform data
boundary	flag for data points in the boundary region (kcopula.de only)

References

Duong, T. (2014) Optimal data-based smoothing for non-parametric estimation of copula functions and their densities. Submitted.

Chen, S.X. (1999). Beta kernel estimator for density functions. *Computational Statistics & Data Analysis*, **31**, 131–145.

See Also

kcde, kde

Examples

```
data(fgl, package="MASS")
x <- fgl[,c("RI", "Na")]
Chat <- kcopula(x=x)
plot(Chat, display="persp", border=1)
plot(Chat, display="filled.contour", lwd=1)
```

---

kda *Kernel discriminant analysis (kernel classification)*

Description

Kernel discriminant analysis (kernel classification) for 1- to d-dimensional data.

Usage

```
kda(x, x.group, Hs, hs, prior.prob=NULL, gridsize, xmin, xmax, supp=3.7,
  eval.points, binned, bgridsize, w, compute.cont=TRUE, approx.cont=TRUE,
  kde.flag=TRUE)
Hkda(x, x.group, Hstart, bw="plugin", ...)
Hkda.diag(x, x.group, bw="plugin", ...)
hkda(x, x.group, bw="plugin", ...)

## S3 method for class 'kda'
predict(object, ..., x)

compare(x.group, est.group, by.group=FALSE)
compare.kda.cv(x, x.group, bw="plugin", prior.prob=NULL, Hstart, by.group=FALSE,
   verbose=FALSE, recompute=FALSE, ...)
compare.kda.diag.cv(x, x.group, bw="plugin", prior.prob=NULL, by.group=FALSE,
   verbose=FALSE, recompute=FALSE, ...)
```

Arguments

x	matrix of training data values
x.group	vector of group labels for training data
Hs, hs	(stacked) matrix of bandwidth matrices/vector of scalar bandwidths. If these are missing, Hkda or hkda is called by default.
prior.prob	vector of prior probabilities
gridsize	vector of grid sizes
xmin, xmax	vector of minimum/maximum values for grid

supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
w	vector of weights. Not yet implemented.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
kde.flag	flag for computing KDE on grid. Default is TRUE.
object	object of class kda
bw	bandwidth: "plugin" = plug-in, "lscv" = LSCV, "scv" = SCV
Hstart	(stacked) matrix of initial bandwidth matrices, used in numerical optimisation
est.group	vector of estimated group labels
by.group	flag to give results also within each group
verbose	flag for printing progress information. Default is FALSE.
recompute	flag for recomputing the bandwidth matrix after excluding the i-th data item
...	other optional parameters for bandwidth selection, see Hpi, Hlscv, Hscv

Details

If the bandwidths Hs are missing from kda, then the default bandwidths are the plug-in selectors Hkda(, bw="plugin"). Likewise for missing hs. Valid options for bw are "plugin", "lscv" and "scv" which in turn call Hpi, Hlscv and Hscv.

The effective support, binning, grid size, grid range, positive parameters are the same as kde.

If prior probabilities are known then set prior.prob to these. Otherwise prior.prob=NULL uses the sample proportions as estimates of the prior probabilities.

For **ks** $\geq$ 1.8.11, kda.kde has been subsumed into kda, so all prior calls to kda.kde can be replaced by kda. To reproduce the previous behaviour of kda, the command is kda(, kde.flag=FALSE).

Value

–For kde.flag=TRUE, a kernel discriminant analysis is an object of class kda which is a list with fields

x	list of data points, one for each group label
estimate	list of density estimates at eval.points, one for each group label
eval.points	vector or list of points that the estimate is evaluated at, one for each group label
h	vector of bandwidths (1-d only)
H	stacked matrix of bandwidth matrices or vector of bandwidths
gridded	flag for estimation on a grid
binned	flag for binned estimation
w	vector of weights

`prior.prob` vector of prior probabilities

`x.group` vector of group labels - same as input

`x.group.estimate`
vector of estimated group labels. If the test data `eval.points` are given then these are classified. Otherwise the training data `x` are classified.

For `kde.flag=FALSE`, which is always the case for $d > 3$, then only the vector of estimated group labels is returned.

–The result from `Hkda` and `Hkda.diag` is a stacked matrix of bandwidth matrices, one for each training data group. The result from `hkda` is a vector of bandwidths, one for each training group.

–The `compare` functions create a comparison between the true group labels `x.group` and the estimated ones. It returns a list with fields

`cross` cross-classification table with the rows indicating the true group and the columns the estimated group

`error` misclassification rate (MR)

In the case where the test data are independent of the training data, `compare` computes MR = (number of points wrongly classified)/(total number of points). In the case where the test data are not independent e.g. we are classifying the training data set itself, then the cross validated estimate of MR is more appropriate. These are implemented as `compare.kda.cv` (unconstrained bandwidth selectors) and `compare.kda.diag.cv` (for diagonal bandwidth selectors). These functions are only available for d > 1.

If `by.group=FALSE` then only the total MR rate is given. If it is set to TRUE, then the MR rates for each class are also given (estimated number in group divided by true number).

References

Simonoff, J. S. (1996) *Smoothing Methods in Statistics*. Springer-Verlag. New York

See Also

`plot.kda`

Examples

```
set.seed(8192)
x <- c(rnorm.mixt(n=100, mus=1), rnorm.mixt(n=100, mus=-1))
x.gr <- rep(c(1,2), times=c(100,100))
y <- c(rnorm.mixt(n=100, mus=1), rnorm.mixt(n=100, mus=-1))
y.gr <- rep(c(1,2), times=c(100,100))
kda.gr <- kda(x, x.gr)
y.gr.est <- predict(kda.gr, x=y)
compare(y.gr, y.gr.est)

## See other examples in ? plot.kda
```

kdcde *Deconvolution kernel density derivative estimate*

Description

Deconvolution kernel density derivative estimate for 1- to 6-dimensional data.

Usage

```
kdcde(x, H, h, Sigma, sigma, reg, bgridsize, gridsize, binned,
      verbose=FALSE, ...)
dckde(...)
```

Arguments

x	matrix of data values
H, h	bandwidth matrix/scalar bandwidth. If these are missing, Hpi or hpi is called by default.
Sigma, sigma	error variance matrix
reg	regularisation parameter
gridsize	vector of number of grid points
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
verbose	flag to print out progress information. Default is FALSE.
...	other parameters to kde

Details

A weighted kernel density estimate is utilised to perform the deconvolution. The weights w are the solution to a quadratic programming problem, and then input into kde(,w=w). This weighted estimate also requires an estimate of the error variance matrix from repeated observations, and of the regularisation parameter. If the latter is missing, it is calculated internally using a 5-fold cross validation method. See Hazelton & Turlach (2009). dckde is an alias for kdcde.

If the bandwidth H is missing from kde, then the default bandwidth is the plug-in selector Hpi. Likewise for missing h.

The effective support, binning, grid size, grid range, positive parameters are the same as kde.

Value

A deconvolution kernel density derivative estimate is an object of class kde which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated

estimate	density estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	vector of weights
cont	vector of probability contour levels

References

Hazelton, M. L. & Turlach, B. A. (2009), Nonparametric density deconvolution by weighted kernel density estimators, *Statistics and Computing*, **19**, 217-228.

See Also

kde

Examples

```
data(air)
air <- air[, c("date", "time", "co2", "pm10")]
air2 <- reshape(air, idvar="date", timevar="time", direction="wide")
air <- as.matrix(na.omit(air2[,c("co2.20:00", "pm10.20:00")]))
Sigma.air <- diag(c(var(air2["co2.19:00"] - air2["co2.21:00"], na.rm=TRUE),
   var(air2["pm10.19:00"] - air2["pm10.21:00"], na.rm=TRUE)))
fhat.air.dec <- kdcde(x=air, Sigma=Sigma.air, reg=0.00021, verbose=TRUE)
plot(fhat.air.dec, drawlabels=FALSE, display="filled.contour", lwd=1)
```

---

kdde *Kernel density derivative estimate*

Description

Kernel density derivative estimate for 1- to 6-dimensional data.

Usage

```
kdde(x, H, h, deriv.order=0, gridsize, gridtype, xmin, xmax, supp=3.7,
    eval.points, binned, bgridsize, positive=FALSE, adj.positive, w,
    deriv.vec=TRUE, verbose=FALSE)
kcurv(fhat, compute.cont=TRUE)

## S3 method for class 'kdde'
predict(object, ..., x)
```

Arguments

x	matrix of data values
H, h	bandwidth matrix/scalar bandwidth. If these are missing, Hpi or hpi is called by default.
deriv.order	derivative order (scalar)
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
positive	flag if data are positive (1-d, 2-d). Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	vector of weights. Default is a vector of all ones.
deriv.vec	flag to compute all derivatives in vectorised derivative. Default is TRUE. If FALSE then only the unique derivatives are computed.
verbose	flag to print out progress information. Default is FALSE.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
fhat	object of class kdde with deriv.order=2
object	object of class kdde
...	other parameters

Details

For each partial derivative, for grid estimation, the estimate is a list whose elements correspond to the partial derivative indices in the rows of deriv.ind. For points estimation, the estimate is a matrix whose columns correspond to the rows of deriv.ind.

If the bandwidth H is missing from kdde, then the default bandwidth is the plug-in selector Hpi. Likewise for missing h.

The effective support, binning, grid size, grid range, positive parameters are the same as kde.

The summary curvature is computed by kcurv, i.e.

$$\hat{s}(\boldsymbol{x}) = -\mathbf{1}\{\mathsf{D}^2\hat{f}(\boldsymbol{x}) < 0\}\mathrm{abs}(|\mathsf{D}^2\hat{f}(\boldsymbol{x})|)$$

where $\mathsf{D}^2\hat{f}(\boldsymbol{x})$ is the kernel Hessian matrix estimate. So $\hat{s}$ calculates the absolute value of the determinant of the Hessian matrix and whose sign is the opposite of the negative definiteness indicator.

Value

A kernel density derivative estimate is an object of class kdde which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	density derivative estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	vector of weights
deriv.order	derivative order (scalar)
deriv.ind	martix where each row is a vector of partial derivative indices

See Also

kde

Examples

```
set.seed(8192)
x <- rmvnorm.mixt(1000, mus=c(0,0), Sigmas=invvech(c(1,0.8,1)))
fhat <- kdde(x=x, deriv.order=1) ## gradient [df/dx, df/dy]
predict(fhat, x=x[1:5,])

## See other examples in ? plot.kdde
```

---

kde *Kernel density estimate*

Description

Kernel density estimate for 1- to 6-dimensional data.

Usage

```
kde(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points, binned,
    bgridsize, positive=FALSE, adj.positive, w, compute.cont=TRUE,
    approx.cont=TRUE, unit.interval=FALSE, density=FALSE, verbose=FALSE)

## S3 method for class 'kde'
predict(object, ..., x, zero.flag=TRUE)
```

Arguments

x	matrix of data values
H, h	bandwidth matrix/scalar bandwidth. If these are missing, Hpi or hpi is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation.
bgridsize	vector of binning grid sizes
positive	flag if data are positive (1-d, 2-d). Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	vector of weights. Default is a vector of all ones.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
unit.interval	flag for computing log transformation KDE on 1-d data bounded on unit interval [0,1]. Default is FALSE.
density	flag if density estimate values are forced to be non-negative function. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.
object	object of class kde
zero.flag	deprecated (retained for backwards compatibilty)
...	other parameters

Details

For d=1, if h is missing, the default bandwidth is hpi. For d>1, if H is missing, the default is Hpi.

For d=1, if positive=TRUE then x is transformed to log(x+adj.positive) where the default adj.positive is the minimum of x. This is known as a log transformation density estimate. If unit.interval=TRUE then x is transformed to qnorm(x). See kde.boundary for boundary kernel density estimates, as these tend to be more robust than transformation density estimates.

For d=1, 2, 3, and if eval.points is not specified, then the density estimate is computed over a grid defined by gridsize (if binned=FALSE) or by bgridsize (if binned=TRUE). This form is suitable for visualisation in conjunction with the plot method.

For d=4, 5, 6, and if eval.points is not specified, then the density estimate is computed over a grid defined by gridsize.

If eval.points is specified, as a vector (d=1) or as a matrix (d=2, 3, 4), then the density estimate is computed at eval.points. This form is suitable for numerical summaries (e.g. maximum likelihood), and is not compatible with the plot method. Despite that the density estimate is returned

only at eval.points, by default, a binned gridded estimate is calculated first and then the density estimate at eval.points is computed using the predict method. If this default intermediate binned grid estimate is not required, then set binned=FALSE to compute directly the exact density estimate at eval.points.

Binned kernel estimation is an approximation to the exact kernel estimation and is available for d=1, 2, 3, 4. This makes kernel estimators feasible for large samples. The default value of the binning flag binned is n>1 (d=1), n>500 (d=2), n>1000 (d>=3). Some times binned estimation leads to negative density values: if non-negative values are required, then set density=TRUE.

The default bgridsize,gridsize are d=1: 401; d=2: rep(151, 2); d=3: rep(51, 3); d=4: rep(21, 4).

The effective support for a normal kernel is where all values outside [-supp,supp]^d are zero.

The default xmin is min(x)-Hmax*supp and xmax is max(x)+Hmax*supp where Hmax is the maximum of the diagonal elements of H. The grid produced is the outer product of c(xmin[1], xmax[1]), ..., c(xmin[d], xmax[d]). For **ks** $\geq$ 1.14.0, when binned=TRUE and xmin,xmax are not missing, the data values x are clipped to the estimation grid delimited by xmin,xmax to prevent potential memory leaks.

Value

A kernel density estimate is an object of class kde which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	density estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	vector of weights
cont	vector of probability contour levels

See Also

plot.kde, kde.boundary

Examples

```
## unit interval data
set.seed(8192)
fhat <- kde(runif(10000,0,1), unit.interval=TRUE)
plot(fhat, ylim=c(0,1.2))

## positive data
data(worldbank)
```

```
wb <- as.matrix(na.omit(worldbank[,2:3]))
wb[,2] <- wb[,2]/1000
fhat <- kde(x=wb)
fhat.trans <- kde(x=wb, adj.positive=c(0,0), positive=TRUE)
plot(fhat, col=1, xlim=c(0,20), ylim=c(0,80))
plot(fhat.trans, add=TRUE, col=2)
rect(0,0,100,100, lty=2)

## large data on non-default grid
## 151 x 151 grid = [-5,-4.933,..,5] x [-5,-4.933,..,5]
set.seed(8192)
x <- rmvnorm.mixt(10000, mus=c(0,0), Sigmas=invvech(c(1,0.8,1)))
fhat <- kde(x=x, compute.cont=TRUE, xmin=c(-5,-5), xmax=c(5,5), bgridsize=c(151,151))
plot(fhat)

## See other examples in ? plot.kde
```

---

kde.boundary *Kernel density estimate for bounded data*

Description

Kernel density estimate for bounded 1- to 3-dimensional data.

Usage

```
kde.boundary(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
   binned=FALSE, bgridsize, w, compute.cont=TRUE, approx.cont=TRUE,
   boundary.supp, boundary.kernel="beta", verbose=FALSE)
```

Arguments

x	matrix of data values
H, h	bandwidth matrix/scalar bandwidth. If these are missing, Hpi or hpi is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation.
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.

boundary.supp	effective support for boundary region
boundary.kernel	"beta" = beta boundary kernel, "linear" = linear boundary kernel
verbose	flag to print out progress information. Default is FALSE.

Details

There are two forms of density estimates which are suitable for bounded data, based on the modifying the kernel function. For boundary.kernel="beta", the 2nd form of the Beta boundary kernel of Chen (1999) is employed. It is suited for rectangular data boundaries.

For boundary.kernel="linear", the linear boundary kernel of Hazelton & Marshall (2009) is employed. It is suited for arbitrarily shaped data boundaries, though it is currently only implemented for rectangular boundaries.

Value

A kernel density estimate for bounded data is an object of class kde.

References

Chen, S. X. (1999) Beta kernel estimators for density functions. *Computational Statistics and Data Analysis*, **31**, 131-145.

Hazelton, M. L. & Marshall, J. C. (2009) Linear boundary kernels for bivariate density estimation. *Statistics and Probability Letters*, **79**, 999-1003.

See Also

kde

Examples

```
data(worldbank)
wb <- as.matrix(na.omit(worldbank[,c("internet", "ag.value")]))
fhat <- kde(x=wb)
fhat.beta <- kde.boundary(x=wb, xmin=c(0,0), xmax=c(100,100), boundary.kernel="beta")
fhat.LB <- kde.boundary(x=wb, xmin=c(0,0), xmax=c(100,100), boundary.kernel="linear")

plot(fhat, col=1, xlim=c(0,100), ylim=c(0,100))
plot(fhat.beta, add=TRUE, col=2)
rect(0,0,100,100, lty=2)
plot(fhat, col=1, xlim=c(0,100), ylim=c(0,100))
plot(fhat.LB, add=TRUE, col=3)
rect(0,0,100,100, lty=2)
```

kde.local.test *Kernel density based local two-sample comparison test*

Description

Kernel density based local two-sample comparison test for 1- to 6-dimensional data.

Usage

```
kde.local.test(x1, x2, H1, H2, h1, h2, fhat1, fhat2, gridsize, binned,
   bgridsize, verbose=FALSE, supp=3.7, mean.adj=FALSE, signif.level=0.05,
   min.ESS, xmin, xmax)
```

Arguments

x1, x2	vector/matrix of data values
H1, H2, h1, h2	bandwidth matrices/scalar bandwidths. If these are missing, Hpi or hpi is called by default.
fhat1, fhat2	objects of class kde
binned	flag for binned estimation
gridsize	vector of grid sizes
bgridsize	vector of binning grid sizes
verbose	flag to print out progress information. Default is FALSE.
supp	effective support for normal kernel
mean.adj	flag to compute second order correction for mean value of critical sampling distribution. Default is FALSE. Currently implemented for d<=2 only.
signif.level	significance level. Default is 0.05.
min.ESS	minimum effective sample size. See below for details.
xmin, xmax	vector of minimum/maximum values for grid

Details

The null hypothesis is $H_0(\boldsymbol{x}) : f_1(\boldsymbol{x}) = f_2(\boldsymbol{x})$ where f_1, f_2 are the respective density functions. The measure of discrepancy is $U(\boldsymbol{x}) = [f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})]^2$. Duong (2013) shows that the test statistic obtained, by substituting the KDEs for the true densities, has a null distribution which is asymptotically chi-squared with 1 d.f.

The required input is either x1, x2 and H1, H2, or fhat1, fhat2, i.e. the data values and bandwidths or objects of class kde. In the former case, the kde objects are created. If the H1, H2 are missing then the default are the plug-in selectors Hpi. Likewise for missing h1, h2.

The mean.adj flag determines whether the second order correction to the mean value of the test statistic should be computed. min.ESS is borrowed from Godtliebsen et al. (2002) to reduce spurious significant results in the tails, though by it is usually not required for small to moderate sample sizes.

Value

A kernel two-sample local significance is an object of class `kde.loctest` which is a list with fields:

`fhat1`, `fhat2`	kernel density estimates, objects of class `kde`
`chisq`	chi squared test statistic
`pvalue`	matrix of local p-values at each grid point
`fhat.diff`	difference of KDEs
`mean.fhat.diff`	mean of the test statistic
`var.fhat.diff`	variance of the test statistic
`fhat.diff.pos`	binary matrix to indicate locally significant fhat1 > fhat2
`fhat.diff.neg`	binary matrix to indicate locally significant fhat1 < fhat2
`n1`, `n2`	sample sizes
`H1`, `H2`, `h1`, `h2`	bandwidth matrices/scalar bandwidths

References

Duong, T. (2013) Local significant differences from non-parametric two-sample tests. *Journal of Nonparametric Statistics*, **25**, 635-645.

Godtliebsen, F., Marron, J.S. & Chaudhuri, P. (2002) Significance in scale space for bivariate density estimation. *Journal of Computational and Graphical Statistics*, **11**, 1-22.

See Also

`kde.test`, `plot.kde.loctest`

Examples

```
data(crabs, package="MASS")
x1 <- crabs[crabs$sp=="B", 4]
x2 <- crabs[crabs$sp=="O", 4]
loct <- kde.local.test(x1=x1, x2=x2)
plot(loct, ylim=c(-0.08,0.12))
cols <- hcl.colors(palette="Dark2",2)
plot(loct$fhat1, add=TRUE, col=cols[1])
plot(loct$fhat2, add=TRUE, col=cols[2])

## see examples in ? plot.kde.loctest
```

kde.test *Kernel density based global two-sample comparison test*

Description

Kernel density based global two-sample comparison test for 1- to 6-dimensional data.

Usage

```
kde.test(x1, x2, H1, H2, h1, h2, psi1, psi2, var.fhat1, var.fhat2,
    binned=FALSE, bgridsize, verbose=FALSE)
```

Arguments

x1, x2	vector/matrix of data values
H1, H2, h1, h2	bandwidth matrices/scalar bandwidths. If these are missing, Hpi.kfe, hpi.kfe is called by default.
psi1, psi2	zero-th order kernel functional estimates
var.fhat1, var.fhat2	sample variance of KDE estimates evaluated at x1, x2
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
verbose	flag to print out progress information. Default is FALSE.

Details

The null hypothesis is $H_0 : f_1 \equiv f_2$ where f_1, f_2 are the respective density functions. The measure of discrepancy is the integrated squared error (ISE) $T = \int [f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})]^2 \, d\boldsymbol{x}$. If we rewrite this as $T = \psi_{0,1} - \psi_{0,12} - \psi_{0,21} + \psi_{0,2}$ where $\psi_{0,uv} = \int f_u(\boldsymbol{x}) f_v(\boldsymbol{x}) \, d\boldsymbol{x}$, then we can use kernel functional estimators. This test statistic has a null distribution which is asymptotically normal, so no bootstrap resampling is required to compute an approximate p-value.

If H1,H2 are missing then the plug-in selector Hpi.kfe is automatically called by kde.test to estimate the functionals with kfe(, deriv.order=0). Likewise for missing h1,h2.

For **ks** $\geq$ 1.8.8, kde.test(,binned=TRUE) invokes binned estimation for the computation of the bandwidth selectors, and not the test statistic and p-value.

Value

A kernel two-sample global significance test is a list with fields:

Tstat	T statistic
zstat	z statistic - normalised version of Tstat
pvalue	p-value of the double sided test
mean, var	mean and variance of null distribution

`var.fhat1`, `var.fhat2`
: sample variances of KDE values evaluated at data points

`n1`, `n2`
: sample sizes

`H1`, `H2`
: bandwidth matrices

`psi1`, `psi12`, `psi21`, `psi2`
: kernel functional estimates

References

Duong, T., Goud, B. & Schauer, K. (2012) Closed-form density-based framework for automatic detection of cellular morphology changes. *PNAS*, **109**, 8382-8387.

See Also

`kde.local.test`

Examples

```
set.seed(8192)
samp <- 1000
x <- rnorm.mixt(n=samp, mus=0, sigmas=1, props=1)
y <- rnorm.mixt(n=samp, mus=0, sigmas=1, props=1)
kde.test(x1=x, x2=y)$pvalue   ## accept H0: f1=f2

data(crabs, package="MASS")
x1 <- crabs[crabs$sp=="B", c(4,6)]
x2 <- crabs[crabs$sp=="O", c(4,6)]
kde.test(x1=x1, x2=x2)$pvalue  ## reject H0: f1=f2
```

---

kde.truncate *Truncated kernel density derivative estimate*

Description

Truncated kernel density derivative estimate for 2-dimensional data.

Usage

```
kde.truncate(fhat, boundary)
kdde.truncate(fhat, boundary)
```

Arguments

`fhat`
: object of class `kde` or `kdde`

`boundary`
: two column matrix delimiting the boundary for truncation

Details

A simple truncation is performed on the kernel estimator. All the points in the estimation grid which are outside of the regions delimited by boundary are set to 0, and their probability mass is distributed proportionally to the remaining density (derivative) values.

Value

A truncated kernel density (derivative) estimate inherits the same object class as the input estimate.

See Also

kde, kdde

Examples

```
data(worldbank)
wb <- as.matrix(na.omit(worldbank[,c("internet", "ag.value")]))
fhat <- kde(x=wb)
rectb <- cbind(x=c(0,100,100,0,0), y=c(0,0,100,100,0))
fhat.b <- kde.truncate(fhat, boundary=rectb)
plot(fhat, col=1, xlim=c(0,100), ylim=c(0,100))
plot(fhat.b, add=TRUE, col=4)
rect(0,0,100,100, lty=2)

library(oz)
data(grevillea)
wa.coast <- ozRegion(section=1)
wa.polygon <- cbind(wa.coast$lines[[1]]$x, wa.coast$lines[[1]]$y)
fhat1 <- kdde(x=grevillea, deriv.order=1)
fhat1 <- kdde.truncate(fhat1, wa.polygon)
oz(section=1, xlim=c(113,122), ylim=c(-36,-29))
plot(fhat1, add=TRUE, display="filled.contour")
```

---

kdr *Kernel density ridge estimation*

Description

Kernel density ridge estimation for 2- to 3-dimensional data.

Usage

```
kdr(x, y, H, p=1, max.iter=400, tol.iter, segment=TRUE, k, kmax, min.seg.size,
    keep.path=FALSE, gridsize, xmin, xmax, binned, bgridsize, w, fhat,
    density.cutoff, pre=TRUE, verbose=FALSE)
kdr.segment(x, k, kmax, min.seg.size, verbose=FALSE)

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

Arguments

x	matrix of data values or an object of class kdr
y	matrix of initial values
p	dimension of density ridge
H	bandwidth matrix/scalar bandwidth. If missing, Hpi(x,deriv,order=2) is called by default.
max.iter	maximum number of iterations. Default is 400.
tol.iter	distance under which two successive iterations are considered convergent. Default is 0.001*min marginal IQR of x.
segment	flag to compute segments of density ridge. Default is TRUE.
k	number of segments to partition density ridge
kmax	maximum number of segments to partition density ridge. Default is 30.
min.seg.size	minimum length of a segment of a density ridge. Default is round(0.001*nrow(y),0).
keep.path	flag to store the density gradient ascent paths. Default is FALSE.
gridsize	vector of number of grid points
xmin, xmax	vector of minimum/maximum values for grid
binned	flag for binned estimation.
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
fhat	kde of x. If missing kde(x=x,w=w) is executed.
density.cutoff	density threshold under which the y are excluded from the density ridge estimation. Default is contourLevels(fhat, cont=99).
pre	flag for pre-scaling data. Default is TRUE.
verbose	flag to print out progress information. Default is FALSE.
...	other graphics parameters

Details

Kernel density ridge estimation is based on reduced dimension kernel mean shift. See Ozertem & Erdogmus (2011).

If y is missing, then it defaults to the grid of size gridsize spanning from xmin to xmax.

If the bandwidth H is missing, then the default bandwidth is the plug-in selector for the density gradient Hpi(x,deriv.order=2). Any bandwidth that is suitable for the density Hessian is also suitable for the kernel density ridge.

kdr(, segment=TRUE) or kdr.segment() carries out the segmentation of the density ridge points in end.points. If k is set, then k segments are created. If k is not set, then the optimal number of segments is chosen from 1:kmax, with kmax=30 by default. The segments are created via a hierarchical clustering with single linkage. *Experimental: following the segmentation, the points within each segment are ordered to facilitate a line plot in plot(, type="l"). The optimal ordering is not always achieved in this experimental implementation, though a scatterplot plot(, type="p") always suffices, regardless of this ordering.*

Value

A kernel density ridge set is an object of class `kdr` which is a list with fields:

`x`, `y`	data points - same as input
`end.points`	matrix of final iterates starting from `y`
`H`	bandwidth matrix
`names`	variable names
`tol.iter`, `tol.clust`, `min.seg.size`	tuning parameter values - same as input
`binned`	flag for binned estimation
`names`	variable names
`w`	vector of weights
`path`	list of density gradient ascent paths where `path[[i]]` is the path of `y[i,]` (only if `keep.path=TRUE`)

References

Ozertem, U. & Erdogmus, D. (2011) Locally defined principal curves and surfaces, *Journal of Machine Learning Research*, **12**, 1249-1286.

Examples

```
data(cardio)
set.seed(8192)
cardio.train.ind <- sample(1:nrow(cardio), round(nrow(cardio)/4,0))
cardio2 <- cardio[cardio.train.ind,c(8,18)]
cardio.dr2 <- kdr(x=cardio2, gridsize=c(21,21))
## gridsize=c(21,21) is for illustrative purposes only
plot(cardio2, pch=16, col=3)
plot(cardio.dr2, cex=0.5, pch=16, col=6, add=TRUE)

## Not run: cardio3 <- cardio[cardio.train.ind,c(8,18,11)]
cardio.dr3 <- kdr(x=cardio3)
plot(cardio.dr3, pch=16, col=6, xlim=c(10,90), ylim=c(70,180), zlim=c(0,40))
plot3D::points3D(cardio3[,1], cardio3[,2], cardio3[,3], pch=16, col=3, add=TRUE)

library(maps)
data(quake)
quake <- quake[quake$prof==1,]  ## Pacific Ring of Fire
quake$long[quake$long<0] <- quake$long[quake$long<0] + 360
quake <- quake[, c("long", "lat")]
data(plate)                     ## tectonic plate boundaries
plate <- plate[plate$long < -20 | plate$long > 20,]
plate$long[plate$long<0 & !is.na(plate$long)] <- plate$long[plate$long<0
& !is.na(plate$long)] + 360

quake.dr <- kdr(x=quake, xmin=c(70,-70), xmax=c(310, 80))
map(wrap=c(0,360), lty=2)
lines(plate[,1:2], col=4, lwd=2)
```

```
plot(quake.dr, type="p", cex=0.5, pch=16, col=6, add=TRUE)
## End(Not run)
```

---

kfe *Kernel functional estimate*

Description

Kernel functional estimate for 1- to 6-dimensional data.

Usage

```
kfe(x, G, deriv.order, inc=1, binned, bin.par, bgridsize, deriv.vec=TRUE,
    add.index=TRUE, verbose=FALSE)
Hpi.kfe(x, nstage=2, pilot, pre="sphere", Hstart, binned=FALSE,
    bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
Hpi.diag.kfe(x, nstage=2, pilot, pre="scale", Hstart, binned=FALSE,
    bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="optim")
hpi.kfe(x, nstage=2, binned=FALSE, bgridsize, amise=FALSE, deriv.order=0)
```

Arguments

x	vector/matrix of data values
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"dscalar" = single pilot bandwidth (default) "dunconstr" = single unconstrained pilot bandwidth
pre	"scale" = pre.scale, "sphere" = pre.sphere
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
amise	flag to return the minimal scaled PI value
deriv.order	derivative order
verbose	flag to print out progress information. Default is FALSE.
optim.fun	optimiser function: one of nlm or optim
G	pilot bandwidth matrix
inc	0=exclude diagonal, 1=include diagonal terms in kfe calculation
bin.par	binning parameters - output from binning
deriv.vec	flag to compute duplicated partial derivatives in the vectorised form. Default is FALSE.
add.index	flag to output derivative indices matrix. Default is true.

Details

`Hpi.kfe` is the optimal plug-in bandwidth for r-th order kernel functional estimator based on the unconstrained pilot selectors of Chacon & Duong (2010). `hpi.kfe` is the 1-d equivalent, using the formulas from Wand & Jones (1995, p.70).

`kfe` does not usually need to be called explicitly by the user.

Value

Plug-in bandwidth matrix for r-th order kernel functional estimator.

References

Chacon, J.E. & Duong, T. (2010) Multivariate plug-in bandwidth selection with unconstrained pilot matrices. *Test*, **19**, 375-398.

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

See Also

`kde.test`

---

kfs *Kernel feature significance*

Description

Kernel feature significance for 1- to 6-dimensional data.

Usage

```
kfs(x, H, h, deriv.order=2, gridsize, gridtype, xmin, xmax, supp=3.7,
    eval.points, binned, bgridsize, positive=FALSE, adj.positive, w,
    verbose=FALSE, signif.level=0.05)
```

Arguments

`x`	matrix of data values
`H`, `h`	bandwidth matrix/scalar bandwidth. If these are missing, `Hpi` or `hpi` is called by default.
`deriv.order`	derivative order (scalar)
`gridsize`	vector of number of grid points
`gridtype`	not yet implemented
`xmin`, `xmax`	vector of minimum/maximum values for grid
`supp`	effective support for standard normal
`eval.points`	vector or matrix of points at which estimate is evaluated

`binned`	flag for binned estimation
`bgridsize`	vector of binning grid sizes
`positive`	flag if 1-d data are positive. Default is FALSE.
`adj.positive`	adjustment applied to positive 1-d data
`w`	vector of weights. Default is a vector of all ones.
`verbose`	flag to print out progress information. Default is FALSE.
`signif.level`	overall level of significance for hypothesis tests. Default is 0.05.

Details

Feature significance is based on significance testing of the gradient (first derivative) and curvature (second derivative) of a kernel density estimate. Only the latter is currently implemented, and is also known as significant modal regions.

The hypothesis test at a grid point $\boldsymbol{x}$ is $H_0(\boldsymbol{x}) : \mathsf{H} f(\boldsymbol{x}) < 0$, i.e. the density Hessian matrix $\mathsf{H} f(\boldsymbol{x})$ is negative definite. The p-values are computed for each $\boldsymbol{x}$ using that the test statistic is approximately chi-squared distributed with $d(d+1)/2$ d.f. We then use a Hochberg-type simultaneous testing procedure, based on the ordered p-values, to control the overall level of significance to be `signif.level`. If $H_0(\boldsymbol{x})$ is rejected then $\boldsymbol{x}$ belongs to a significant modal region.

The computations are based on `kdde(x, deriv.order=2)` so `kfs` inherits its behaviour from `kdde`. If the bandwidth `H` is missing, then the default bandwidth is the plug-in selector `Hpi(,deriv.order=2)`. Likewise for missing `h`. The effective support, binning, grid size, grid range, positive parameters are the same as `kde`.

This function is similar to the `featureSignif` function in the **feature** package, except that it accepts unconstrained bandwidth matrices.

Value

A kernel feature significance estimate is an object of class `kfs` which is a list with fields

`x`	data points - same as input
`eval.points`	vector or list of points at which the estimate is evaluated
`estimate`	binary matrix for significant feature at `eval.points`: 0 = not signif., 1 = signif.
`h`	scalar bandwidth (1-d only)
`H`	bandwidth matrix
`gridtype`	"linear"
`gridded`	flag for estimation on a grid
`binned`	flag for binned estimation
`names`	variable names
`w`	vector of weights
`deriv.order`	derivative order (scalar)
`deriv.ind`	martix where each row is a vector of partial derivative indices.

This is the same structure as a `kdde` object, except that `estimate` is a binary matrix rather than real-valued.

References

Chaudhuri, P. & Marron, J.S. (1999) SiZer for exploration of structures in curves. *Journal of the American Statistical Association*, **94**, 807-823.

Duong, T., Cowling, A., Koch, I. & Wand, M.P. (2008) Feature significance for multivariate kernel density estimation. *Computational Statistics and Data Analysis*, **52**, 4225-4242.

Godtliebsen, F., Marron, J.S. & Chaudhuri, P. (2002) Significance in scale space for bivariate density estimation. *Journal of Computational and Graphical Statistics*, **11**, 1-22.

See Also

kdde, plot.kfs

Examples

```
data(geyser, package="MASS")
geyser.fs <- kfs(geyser$duration, binned=TRUE)
plot(geyser.fs, xlab="duration")

## see example in ? plot.kfs
```

---

kms *Kernel mean shift clustering*

Description

Kernel mean shift clustering for 2- to 6-dimensional data.

Usage

```
kms(x, y, H, max.iter=400, tol.iter, tol.clust, min.clust.size, merge=TRUE,
    keep.path=FALSE, verbose=FALSE)

## S3 method for class 'kms'
plot(x, display="splom", col, col.fun, alpha=1, xlab, ylab, zlab, theta=-30,
    phi=40, add=FALSE, ...)
## S3 method for class 'kms'
summary(object, ...)
```

Arguments

x	matrix of data values or object of class kms
y	matrix of candidate data values for which the mean shift will estimate their cluster labels. If missing, y=x.
H	bandwidth matrix/scalar bandwidth. If missing, Hpi(x,deriv.order=1,nstage=2-(d>2)) is called by default.
max.iter	maximum number of iterations. Default is 400.

tol.iter	distance under which two successive iterations are considered convergent. Default is 0.001*min marginal IQR of x.
tol.clust	distance under which two cluster modes are considered to form one cluster. Default is 0.01*max marginal IQR of x.
min.clust.size	minimum cluster size (cardinality). Default is 0.01*nrow(y).
merge	flag to merge clusters which are smaller than min.clust.size. Default is TRUE.
keep.path	flag to store the density gradient ascent paths. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.
object	object of class kms
display	type of display, "splom" (>=2-d) or "plot3D" (3-d)
col, col.fun	vector or colours (one for each group) or colour function
alpha	colour transparency. Default is 1.
xlab, ylab, zlab	axes labels
theta, phi	graphics parameters for perspective plots (3-d)
add	flag to add to current plot. Default is FALSE.
...	other (graphics) parameters

Details

Mean shift clustering belongs to the class of modal or density-based clustering methods. The mean shift recurrence of the candidate point $\boldsymbol{x}$ is $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \mathbf{H}\mathsf{D}\hat{f}(\boldsymbol{x}_j)/\hat{f}(\boldsymbol{x}_j)$ where $j \geq 0$ and $\boldsymbol{x}_0 = \boldsymbol{x}$. The sequence $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots\}$ follows the density gradient ascent paths to converge to a local mode of the density estimate $\hat{f}$. Hence $\boldsymbol{x}$ is iterated until it converges to its local mode, and this determines its cluster label.

The mean shift recurrence is terminated if successive iterations are less than tol.iter or the maximum number of iterations max.iter is reached. Final iterates which are less than tol.clust distance apart are considered to form a single cluster. If merge=TRUE then the clusters whose cardinality is less than min.clust.size are iteratively merged with their nearest cluster.

If the bandwidth H is missing, then the default bandwidth is the plug-in selector for the density gradient Hpi(x,deriv.order=1). Any bandwidth that is suitable for the density gradient is also suitable for the mean shift.

Value

A kernel mean shift clusters set is an object of class kms which is a list with fields:

x, y	data points - same as input
end.points	matrix of final iterates starting from y
H	bandwidth matrix
label	vector of cluster labels
nclust	number of clusters
nclust.table	frequency table of cluster labels

mode	matrix of cluster modes
names	variable names
tol.iter, tol.clust, min.clust.size	tuning parameter values - same as input
path	list of density gradient ascent paths where path[[i]] is the path of y[i,] (only if keep.path=TRUE)

References

Chacon, J.E. & Duong, T. (2013) Data-driven density estimation, with applications to nonparametric clustering and bump hunting. *Electronic Journal of Statistics*, **7**, 499-532.

Comaniciu, D. & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 603-619.

See Also

kde

Examples

```
data(crabs, package="MASS")
kms.crabs <- kms(x=crabs[,c("FL","CW")])
plot(kms.crabs, pch=16)
summary(kms.crabs)

kms.crabs <- kms(x=crabs[,c("FL","CW","RW")])
plot(kms.crabs, pch=16)
plot(kms.crabs, display="plot3D", pch=16)
```

---

kroc *Kernel receiver operating characteristic (ROC) curve*

Description

Kernel receiver operating characteristic (ROC) curve for 1- to 3-dimensional data.

Usage

```
kroc(x1, x2, H1, h1, hy, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
   binned, bgridsize, positive=FALSE, adj.positive, w, verbose=FALSE)

## S3 method for class 'kroc'
predict(object, ..., x)
## S3 method for class 'kroc'
summary(object, ...)
```

Arguments

x, x1, x2	vector/matrix of data values
H1, h1, hy	bandwidth matrix/scalar bandwidths. If these are missing, Hpi.kcde, hpi.kcde is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	not yet implemented
binned	flag for binned estimation
bgridsize	vector of binning grid sizes
positive	flag if 1-d data are positive. Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	vector of weights. Default is a vector of all ones.
verbose	flag to print out progress information. Default is FALSE.
object	object of class kroc, output from kroc
...	other parameters

Details

In this set-up, the values in the first sample x1 should be larger in general that those in the second sample x2. The usual method for computing 1-d ROC curves is not valid for multivariate data. Duong (2014), based on Lloyd (1998), develops an alternative formulation $(F_{Y_1}(z), F_{Y_2}(z))$ based on the cumulative distribution functions of $Y_j = \bar{F}_1(\boldsymbol{X}_j), j = 1, 2$.

If the bandwidth H1 is missing from kroc, then the default bandwidth is the plug-in selector Hpi.kcde. Likewise for missing h1,hy. A bandwidth matrix H1 is required for x1 for d>1, but the second bandwidth hy is always a scalar since Y_j are 1-d variables.

The effective support, binning, grid size, grid range, positive parameters are the same as kde.

–The summary method for kroc objects prints out the summary indices of the ROC curve, as contained in the indices field, namely the AUC (area under the curve) and Youden index.

Value

A kernel ROC curve is an object of class kroc which is a list with fields:

x	list of data values x1, x2 - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	ROC curve estimate at eval.points
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names

w	vector of weights
tail	"lower.tail"
h1	scalar bandwidth for first sample (1-d only)
H1	bandwidth matrix for first sample
hy	scalar bandwidth for ROC curve
indices	summary indices of ROC curve.

References

Duong, T. (2016) Non-parametric smoothed estimation of multivariate cumulative distribution and survival functions, and receiver operating characteristic curves. *Journal of the Korean Statistical Society*, **45**, 33-50.

Lloyd, C. (1998) Using smoothed receiver operating curves to summarize and compare diagnostic systems. *Journal of the American Statistical Association*, **93**, 1356-1364.

See Also

kcde

Examples

```
samp <- 1000
x <- rnorm.mixt(n=samp, mus=0, sigmas=1, props=1)
y <- rnorm.mixt(n=samp, mus=0.5, sigmas=1, props=1)
Rhat <- kroc(x1=x, x2=y)
summary(Rhat)
predict(Rhat, x=0.5)
```

---

ksupp *Kernel support estimate*

Description

Kernel support estimate for 2 and 3-dimensional data.

Usage

```
ksupp(fhat, cont=95, abs.cont, convex.hull=TRUE)

## S3 method for class 'ksupp'
plot(x, display="plot3D", ...)
```

Arguments

fhat	object of class kde
cont	percentage for contour level curve. Default is 95.
abs.cont	absolute density estimate height for contour level curve
convex.hull	flag to compute convex hull of contour level curve. Default is TRUE.
x	object of class ksupp
display	one of "plot3D", "rgl" (required for 3-d only)
...	other graphics parameters

Details

The kernel support estimate is the level set of the density estimate that exceeds the cont percent contour level. If this level set is a simply connected region, then this can suffice to be a conservative estimate of the density support. Otherwise, the convex hull of the level set is advised. For 2-d data, the convex hull is computed by chull; for 3-d data, it is computed by geometry::convhulln.

Value

A kernel support estimate is an object of class ksupp, i.e. a 2- or 3-column matrix which delimits the (convex hull of the) level set of the density estimate fhat.

See Also

kde

Examples

```
data(grevillea)
fhat <- kde(x=grevillea)
fhat.supp <- ksupp(fhat)
plot(fhat, display="filled.contour", cont=seq(10,90,by=10))
plot(fhat, cont=95, add=TRUE, col=1)
plot(fhat.supp, lty=2)

data(iris)
fhat <- kde(x=iris[,1:3])
fhat.supp <- ksupp(fhat)
plot(fhat)
plot(fhat.supp, add=TRUE, col=3, alpha=0.1)
```

mixt *Normal and t-mixture distributions*

Description

Random generation and density values from normal and t-mixture distributions.

Usage

```
dnorm.mixt(x, mus=0, sigmas=1, props=1)
rnorm.mixt(n=100, mus=0, sigmas=1, props=1, mixt.label=FALSE)
dmvnorm.mixt(x, mus, Sigmas, props=1, verbose=FALSE)
rmvnorm.mixt(n=100, mus=c(0,0), Sigmas=diag(2), props=1, mixt.label=FALSE)
rmvt.mixt(n=100, mus=c(0,0), Sigmas=diag(2), dfs=7, props=1)
dmvt.mixt(x, mus, Sigmas, dfs, props)
mvnorm.mixt.mode(mus, Sigmas, props=1, verbose=FALSE)
```

Arguments

n	number of random variates
x	matrix of quantiles
mus	(stacked) matrix of mean vectors (>1-d) or vector of means (1-d)
Sigmas	(stacked) matrix of variance matrices (>1-d)
sigmas	vector of standard deviations (1-d)
props	vector of mixing proportions
mixt.label	flag to output numeric label indicating mixture component. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.
dfs	vector of degrees of freedom

Details

`rmvnorm.mixt` and `dmvnorm.mixt` are based on the `rmvnorm` and `dmvnorm` functions from the **mvtnorm** package. Likewise for `rmvt.mixt` and `dmvt.mixt`.

For the normal mixture densities, `mvnorm.mixt.mode` computes the local modes: these are usually very close but not exactly equal to the component means.

Value

Normal and t-mixture random vectors and density values.

Examples

```
## univariate normal mixture
x <- rnorm.mixt(1000, mus=c(-1,1), sigmas=c(0.5, 0.5), props=c(1/2, 1/2))

## bivariate mixtures
mus <- rbind(c(-1,0), c(1, 2/sqrt(3)), c(1,-2/sqrt(3)))
Sigmas <- 1/25*rbind(invvech(c(9, 63/10, 49/4)), invvech(c(9,0,49/4)), invvech(c(9,0,49/4)))
props <- c(3,3,1)/7
dfs <- c(7,3,2)
x <- rmvnorm.mixt(1000, mus=mus, Sigmas=Sigmas, props=props)
y <- rmvt.mixt(1000, mus=mus, Sigmas=Sigmas, dfs=dfs, props=props)

mvnorm.mixt.mode(mus=mus, Sigmas=Sigmas, props=props)
```

---

plot.histde *Plot for histogram density estimate*

Description

Plot for histogram density estimate for 1- and 2-dimensional data.

Usage

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

Arguments

x	object of class histde (output from histde)
...	other graphics parameters:
	col plotting colour for density estimate
	col.fun plotting colour function for levels
	col.pt plotting colour for data points
	jitter flag to jitter rug plot (1-d). Default is TRUE.
	xlim,ylim axes limits
	xlab,ylab axes labels
	add flag to add to current plot. Default is FALSE.
	drawpoints flag to draw data points on density estimate. Default is FALSE.
	breaks vector of break values of density estimate. Default is an nbreaks equi-linear sequence over the data range.
	nbreaks number of breaks in breaks sequence
	lty.rect,lwd.rect line type/width for histogram box lines (2-d)
	border colour of histogram box lines (2-d)
	col.rect colour of histogram bars (1-d)
	add.grid flag to add histogram grid (2-d). Default is TRUE.

Details

For histde objects, the function headers for the different dimensional data are

```
## univariate
plot(fhat, xlab, ylab="Density function", add=FALSE, drawpoints=FALSE,
   col.pt=4, jitter=FALSE, border=1, alpha=1, ...)

## bivariate
plot(fhat, breaks, nbreaks=11, xlab, ylab, zlab="Density function", cex=1,
   pch=1, add=FALSE, drawpoints=FALSE, col, col.fun, alpha=1, col.pt=4,
   lty.rect=2, cex.text=1, border, lwd.rect=1, col.rect="transparent",
   add.grid=TRUE, ...)
```

The 1-d plot is a standard plot of a histogram generated by hist. If drawpoints=TRUE then a rug plot is added.

The 2-d plot is similar to the display="filled.contour" option from plot.kde with the default nbreaks=11 contour levels.

Value

Plots for 1-d and 2-d are sent to graphics window.

See Also

plot.kde

Examples

```
data(iris)

## univariate example
fhat <- histde(x=iris[,2])
plot(fhat, xlab="Sepal length")

## bivariate example
fhat <- histde(x=iris[,2:3])
plot(fhat, drawpoints=TRUE)
box()
```

---

plot.kcde *Plot for kernel cumulative distribution estimate*

Description

Plot for kernel cumulative distribution estimate 1- to 3-dimensional data.

Usage

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

Arguments

x	object of class kcde (output from kcde)
...	other graphics parameters used in plot.kde

Details

For kcde objects, the function headers for the different dimensional data are

```
## univariate
plot(Fhat, xlab, ylab="Distribution function", add=FALSE, drawpoints=FALSE,
     col.pt=4, jitter=FALSE, alpha=1, ...)

## bivariate
plot(Fhat, display="persp", cont=seq(10,90, by=10), abs.cont, xlab, ylab,
     zlab="Distribution function", cex=1, pch=1, add=FALSE, drawpoints=FALSE,
    drawlabels=TRUE, theta=-30, phi=40, d=4, col.pt=4, col, col.fun, alpha=1,
     lwd=1, border=NA, thin=3, lwd.fc=5, ...)

## trivariate
plot(Fhat, display="plot3D", cont=c(25,50,75), colors, col, alphavec,
     size=3, cex=1, pch=1, theta=-30, phi=40, d=4, ticktype="detailed",
     bty="f", col.pt=4, add=FALSE, xlab, ylab, zlab, drawpoints=FALSE,
     alpha, box=TRUE, axes=TRUE, ...)
```

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

See Also

plot.kde

Examples

```
data(iris)
Fhat <- kcde(x=iris[,1])
plot(Fhat, xlab="Sepal.Length")
Fhat <- kcde(x=iris[,1:2])
plot(Fhat)
Fhat <- kcde(x=iris[,1:3])
plot(Fhat, alpha=0.3)
```

`plot.kda` *Plot for kernel discriminant analysis*

Description

Plot for kernel discriminant analysis for 1- to 3-dimensional data.

Usage

```
## S3 method for class 'kda'
plot(x, y, y.group, ...)
```

Arguments

`x`	object of class `kda` (output from `kda`)
`y`	matrix of test data points
`y.group`	vector of group labels for test data points
`...`	other graphics parameters: `rugsize` height of rug-like plot for partition classes (1-d) `prior.prob` vector of prior probabilities `col.part` vector of colours for partition classes (1-d, 2-d) and those used in `plot.kde`

Details

For `kda` objects, the function headers for the different dimensional data are

```
## univariate
plot(x, y, y.group, prior.prob=NULL, xlim, ylim, xlab,
     ylab="Weighted density function", drawpoints=FALSE, col, col.fun,
     col.part, col.pt, lty, jitter=TRUE, rugsize, add=FALSE, alpha=1, ...)

## bivariate
plot(x, y, y.group, prior.prob=NULL, display.part="filled.contour",
     cont=c(25,50,75), abs.cont, approx.cont=TRUE, xlim, ylim, xlab, ylab,
     drawpoints=FALSE, drawlabels=TRUE, cex=1, pch, lty, part=TRUE, col,
     col.fun, col.part, col.pt, alpha=1, lwd=1, lwd.part=0, add=FALSE, ...)

## trivariate
plot(x, y, y.group, prior.prob=NULL, display="plot3D", cont=c(25,50,75),
     abs.cont, approx.cont=TRUE, colors, col, col.fun, col.pt, alpha=0.5,
     alphavec, xlab, ylab, zlab, drawpoints=FALSE, size=3, cex=1, pch,
     theta=-30, phi=40, d=4, ticktype="detailed", bty="f", add=FALSE, ...)
```

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

See Also

kda, kde

Examples

```
data(iris)

## univariate example
ir <- iris[,1]
ir.gr <- iris[,5]
kda.fhat <- kda(x=ir, x.group=ir.gr, xmin=3, xmax=9)
plot(kda.fhat, xlab="Sepal length")

## bivariate example
ir <- iris[,1:2]
ir.gr <- iris[,5]
kda.fhat <- kda(x=ir, x.group=ir.gr)
plot(kda.fhat, alpha=0.2, drawlabels=FALSE)

## trivariate example
ir <- iris[,1:3]
ir.gr <- iris[,5]
kda.fhat <- kda(x=ir, x.group=ir.gr)
plot(kda.fhat)
   ## colour=species, transparency=density heights
```

---

plot.kdde *Plot for kernel density derivative estimate*

Description

Plot for kernel density derivative estimate for 1- to 3-dimensional data.

Usage

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

Arguments

x	object of class kdde (output from kdde)
...	other graphics parameters:
	which.deriv.ind index of the partial derivative to be plotted (>1-d)
	and those used in plot.kde

Details

For kdde objects, the function headers for the different dimensional data are

```
## univariate
plot(fhat, ylab="Density derivative function", cont=50, abs.cont, alpha=1, ...)

## bivariate
plot(fhat, which.deriv.ind=1, cont=c(25,50,75), abs.cont, display="slice",
    zlab="Density derivative function", col, col.fun, alpha=1, kdde.flag=TRUE,
     thin=3, transf=1, neg.grad=FALSE, ...)

## trivariate
plot(fhat, which.deriv.ind=1, display="plot3D", cont=c(25,50,75), abs.cont,
     colors, col, col.fun, ...)
```

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

In addition to the display options inherited from `plot.kde`, the first derivative has `display="quiver"`. This is a quiver plot where the size and direction of the arrow indicates the magnitude/direction of the density gradient. See `quiver` from the **pracma** package for more details.

See Also

`plot.kde`

Examples

```
## univariate example
data(tempb)
fhat1 <- kde(x=tempb[,"tmin"], deriv.order=1)   ## gradient [df/dx, df/dy]
plot(fhat1, xlab="Min. temp.", col.cont=4)      ## df/dx
points(20,predict(fhat1, x=20))

## bivariate example
fhat1 <- kde(x=tempb[,c("tmin", "tmax")], deriv.order=1)
plot(fhat1, display="quiver")
  ## gradient [df/dx, df/dy]

fhat2 <- kde(x=tempb[,c("tmin", "tmax")], deriv.order=2)
plot(fhat2, which.deriv.ind=2, display="persp", phi=10)
plot(fhat2, which.deriv.ind=2, display="filled.contour")
  ## d^2 f/(dx dy): blue=-ve, red=+ve
s2 <- kcurv(fhat2)
plot(s2, display="filled.contour", alpha=0.5, lwd=1)
  ## summary curvature

## trivariate example
data(iris)
fhat1 <- kde(iris[,2:4], deriv.order=1)
plot(fhat1)
```

`plot.kde` *Plot for kernel density estimate*

Description

Plot for kernel density estimate for 1- to 3-dimensional data.

Usage

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

Arguments

`x`	object of class `kde` (output from `kde`)
`...`	other graphics parameters:
	`display` type of display, "slice" for contour plot, "persp" for perspective plot, "image" for image plot, "filled.contour" for filled contour plot (2-d); "plot3D", "rgl" (3-d)
	`cont` vector of percentages for contour level curves
	`abs.cont` vector of absolute density estimate heights for contour level curves
	`approx.cont` flag to compute approximate contour levels. Default is FALSE.
	`col` plotting colour for density estimate (1-d, 2-d)
	`col.cont` plotting colour for contours
	`col.fun` plotting colour function for contours
	`col.pt` plotting colour for data points
	`colors` vector of colours for each contour (3-d)
	`jitter` flag to jitter rug plot (1-d). Default is TRUE.
	`lwd.fc` line width for filled contours (2-d)
	`xlim,ylim,zlim` axes limits
	`xlab,ylab,zlab` axes labels
	`add` flag to add to current plot. Default is FALSE.
	`theta,phi,d,border` graphics parameters for perspective plots (2-d)
	`drawpoints` flag to draw data points on density estimate. Default is FALSE.
	`drawlabels` flag to draw contour labels (2-d). Default is TRUE.
	`alpha` transparency value of plotting symbol
	`alphavec` vector of transparency values for contours (3-d)
	`size` size of plotting symbol (3-d).

Details

For kde objects, the function headers for the different dimensional data are

```
## univariate
plot(fhat, xlab, ylab="Density function", add=FALSE, drawpoints=FALSE, col=1,
     col.pt=4, col.cont=1, cont.lwd=1, jitter=FALSE, cont, abs.cont,
     approx.cont=TRUE, alpha=1, ...)

## bivariate
plot(fhat, display="slice", cont=c(25,50,75), abs.cont, approx.cont=TRUE,
     xlab, ylab, zlab="Density function", cex=1, pch=1, add=FALSE,
     drawpoints=FALSE, drawlabels=TRUE, theta=-30, phi=40, d=4, col.pt=4,
     col, col.fun, alpha=1, lwd=1, border=1, thin=3, kdde.flag=FALSE,
     ticktype="detailed", ...)

## trivariate
plot(fhat, display="plot3D", cont=c(25,50,75), abs.cont, approx.cont=TRUE,
     colors, col, col.fun, alphavec, size=3, cex=1, pch=1, theta=-30, phi=40,
     d=4, ticktype="detailed", bty="f", col.pt=4, add=FALSE, xlab, ylab,
     zlab, drawpoints=FALSE, alpha, box=TRUE, axes=TRUE, ...)
```

For 1-dimensional data, the plot is a standard plot of a 1-d curve. If `drawpoints=TRUE` then a rug plot is added. If `cont` is specified, the horizontal line on the x-axis indicates the `cont`% highest density level set.

For 2-dimensional data, the different types of plotting displays are controlled by the `display` parameter. (a) If `display="slice"` then a slice/contour plot is generated using `contour`. (b) If `display` is `"filled.contour"` then a filled contour plot is generated. The default contours are at 25%, 50%, 75% or `cont=c(25,50,75)` which are upper percentages of highest density regions. (c) If `display="persp"` then a perspective/wire-frame plot is generated. The default z-axis limits `zlim` are the default from the usual `persp` command. (d) If `display="image"` then an image plot is generated.

For 3-dimensional data, the plot is a series of nested 3-d contours. The default contours are `cont=c(25,50,75)`. The default opacity `alphavec` ranges from 0.1 to 0.5. For **ks** $\geq$ 1.12.0, base R graphics becomes the default plotting engine: to create an **rgl** plot like in previous versions, set `display="rgl"`.

To specify contours, either one of `cont` or `abs.cont` is required. `cont` specifies upper percentages which correspond to probability contour regions. If `abs.cont` is set to particular values, then contours at these levels are drawn. This second option is useful for plotting multiple density estimates with common contour levels. See `contourLevels` for details on computing contour levels. If `approx=FALSE`, then the exact KDE is computed. Otherwise it is interpolated from an existing KDE grid, which can dramatically reduce computation time for large data sets.

If a colour function is specified in `col.fun`, it should have the number of colours as a single argument, e.g. `function(n){hcl.colors(n, ...)}`. The transparent background colour is automatically concatenated before this colour function. If `col` is specified, it overrides `col.fun`. There should be one more colour than the number of contours, i.e. background colour plus one for each contour.

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

Examples

```
data(iris)

## univariate example
fhat <- kde(x=iris[,2])
plot(fhat, cont=50, col.cont=4, cont.lwd=2, xlab="Sepal length")

## bivariate example
fhat <- kde(x=iris[,2:3])
plot(fhat, display="filled.contour", cont=seq(10,90,by=10), lwd=1, alpha=0.5)
plot(fhat, display="persp", border=1, alpha=0.5)

## trivariate example
fhat <- kde(x=iris[,2:4])
plot(fhat)
if (interactive()) plot(fhat, display="rgl")
```

---

plot.kde.loctest *Plot for kernel local significant difference regions*

Description

Plot for kernel local significant difference regions for 1- to 3-dimensional data.

Usage

```
## S3 method for class 'kde.loctest'
plot(x, ...)
```

Arguments

x	object of class kde.loctest (output from kde.local.test)
...	other graphics parameters:
	lcol colour for KDE curve (1-d)
	col vector of 2 colours. First colour: sample 1>sample 2, second colour: sample 1<sample2.
	add flag to add to current plot. Default is FALSE.
	rugsize height of rug-like plot (1-d)
	add.legend flag to add legend. Default is TRUE.
	pos.legend position label for legend (1-d, 2-d)
	alphavec vector of transparency values for contour (3-d)
	and those used in plot.kde

Details

For kde.loctest objects, the function headers are

```
## univariate
plot(x, lcol, col, add=FALSE, xlab="x", ylab, rugsize, add.legend=TRUE,
     pos.legend="topright", alpha=1, ...)

## bivariate
plot(x, col, add=FALSE, add.legend=TRUE, pos.legend="topright", alpha=1,
     ...)

## trivariate
plot(x, col, color, add=FALSE, box=TRUE, axes=TRUE, alphavec=c(0.5, 0.5),
     add.legend=TRUE, ...)
```

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

See Also

kde.local.test

Examples

```
## bivariate
data(air)
air.var <- c("co2","pm10","no")
air <- air[, c("date","time",air.var)]
air2 <- reshape(air, idvar="date", timevar="time", direction="wide")
a1 <- as.matrix(na.omit(air2[, paste0(air.var, ".08:00")]))
a2 <- as.matrix(na.omit(air2[, paste0(air.var, ".20:00")]))
colnames(a1) <- air.var
colnames(a2) <- air.var
loct <- kde.local.test(x1=a1[,c("co2","pm10")], x2=a2[,c("co2","pm10")])
plot(loct, lwd=1)

## trivariate
loct <- kde.local.test(x1=a1, x2=a2)
plot(loct, xlim=c(0,800), ylim=c(0,300), zlim=c(0,300))
```

---

plot.kde.part *Partition plot for kernel density clustering*

Description

Plot of partition for kernel density clustering for 2-dimensional data.

Usage

```
mvnorm.mixt.part(mus, Sigmas, props=1, xmin, xmax, gridsize, max.iter=100,
   verbose=FALSE)
kms.part(x, H, xmin, xmax, gridsize, verbose=FALSE, ...)

## S3 method for class 'kde.part'
plot(x, display="filled.contour", col, col.fun, alpha=1, add=FALSE, ...)
```

Arguments

mus	(stacked) matrix of mean vectors
Sigmas	(stacked) matrix of variance matrices
props	vector of mixing proportions
xmin, xmax	vector of minimum/maximum values for grid
gridsize	vector of number of grid points
max.iter	maximum number of iterations
verbose	flag to print out progress information. Default is FALSE.
x	matrix of data values or an object of class kde.part
H	bandwidth matrix. If missing, Hpi(x,deriv,order=1) is called by default.
display	type of display, "filled.contour" for filled contour plot
col, col.fun	vector of plotting colours or colour function
alpha	colour transparency. Default is 1.
add	flag to add to current plot. Default is FALSE.
...	other parameters

Details

For 2-d data, kms.part and mvnorm.mixt.part produce a kde.part object whose values are the class labels, rather than probability density values.

Value

A kernel partition is an object of class kde.part which is a list with fields:

x	data points - same as input
eval.points	vector or list of points at which the estimate is evaluated
estimate	density estimate at eval.points
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names

w	vector of weights
cont	vector of probability contour levels
end.points	matrix of final iterates starting from x
label	vector of cluster labels
mode	matrix of cluster modes
nclust	number of clusters
nclust.table	frequency table of cluster labels
tol.iter, tol.clust, min.clust.size	tuning parameter values - same as input

Plot is sent to graphics window.

See Also

plot.kde, kms

Examples

```
## normal mixture partition
mus <- rbind(c(-1,0), c(1, 2/sqrt(3)), c(1,-2/sqrt(3)))
Sigmas <- 1/25*rbind(invvech(c(9, 63/10, 49/4)), invvech(c(9,0,49/4)), invvech(c(9,0,49/4)))
props <- c(3,3,1)/7
gridsize <- c(11,11) ## small gridsize illustrative purposes only
nmixt.part <- mvnorm.mixt.part(mus=mus, Sigmas=Sigmas, props=props, gridsize=gridsize)
plot(nmixt.part, asp=1, xlim=c(-3,3), ylim=c(-3,3), alpha=0.5)

## kernel mean shift partition
set.seed(81928192)
x <- rmvnorm.mixt(n=10000, mus=mus, Sigmas=Sigmas, props=props)
msize <- round(prod(gridsize)*0.1)
kms.nmixt.part <- kms.part(x=x, min.clust.size=msize, gridsize=gridsize)
plot(kms.nmixt.part, asp=1, xlim=c(-3,3), ylim=c(-3,3), alpha=0.5)
```

---

plot.kfs *Plot for kernel feature significance*

Description

Plot for kernel significant regions for 1- to 3-dimensional data.

Usage

```
## S3 method for class 'kfs'
plot(x, display="filled.contour", col=7, colors, abs.cont,
   alpha=1, alphavec=0.4, add=FALSE, ...)
```

Arguments

x	object of class kfs (output from kfs)
display	type of display, "slice" for contour plot, "persp" for perspective plot, "image" for image plot, "filled.contour" for filled contour plot (2-d); "plot3D", "rgl" (3-d)
col, colors	colour for contour region
abs.cont	absolute contour height. Default is 0.5.
alpha	transparency value for contour (2-d)
alphavec	vector of transparency values for contour (3-d)
add	flag to add to current plot. Default is FALSE.
...	other graphics parameters used in plot.kde

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

See Also

plot.kde

Examples

```
data(geyser, package="MASS")
geyser.fs <- kfs(geyser, binned=TRUE)
plot(geyser.fs)
```

---

plot.kroc *Plot for kernel receiver operating characteristic curve (ROC) estimate*

Description

Plot for kernel receiver operating characteristic curve (ROC) estimate 1- to 3-dimensional data.

Usage

```
## S3 method for class 'kroc'
plot(x, add=FALSE, add.roc.ref=FALSE, xlab, ylab,
   alpha=1, col=1, ...)
```

Arguments

x	object of class kroc (output from kroc)
add	flag to add to current plot. Default is FALSE.
add.roc.ref	flag to add reference ROC curve. Default is FALSE.
xlab	x-axis label. Default is "False positive rate (bar(specificity))".
ylab	y-axis label. Default is "True positive rate (sensitivity)".
alpha, col	transparency value and colour of line
...	other graphics parameters used in plot.kde.

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to graphics/RGL window.

See Also

plot.kde

Examples

```
data(fgl, package="MASS")
x1 <- fgl[fgl[,"type"]=="WinF",c("RI", "Na")]
x2 <- fgl[fgl[,"type"]=="Head",c("RI", "Na")]
Rhat <- kroc(x1=x1, x2=x2)
plot(Rhat, add.roc.ref=TRUE)
```

---

plotmixt *Plot for 1- to 3-dimensional normal and t-mixture density functions*

Description

Plot for 1- to 3-dimensional normal and t-mixture density functions.

Usage

```
plotmixt(mus, sigmas, Sigmas, props, dfs, dist="normal", draw=TRUE,
   deriv.order=0, which.deriv.ind=1, binned=TRUE, ...)
```

Arguments

mus	(stacked) matrix of mean vectors
sigmas	vector of standard deviations (1-d)
Sigmas	(stacked) matrix of variance matrices (2-d, 3-d)
props	vector of mixing proportions
dfs	vector of degrees of freedom
dist	"normal" - normal mixture, "t" - t-mixture
draw	flag to draw plot. Default is TRUE.
deriv.order	derivative order
which.deriv.ind	index of which partial derivative to plot
binned	flag for binned estimation of contour levels. Default is TRUE.
...	other graphics parameters, see plot.kde

Value

If draw=TRUE, the 1-d, 2-d plot is sent to graphics window, 3-d plot to graphics/RGL window. If draw=FALSE, then a kdde-like object is returned.

Examples

```
## bivariate
mus <- rbind(c(0,0), c(-1,1))
Sigma <- matrix(c(1, 0.7, 0.7, 1), nr=2, nc=2)
Sigmas <- rbind(Sigma, Sigma)
props <- c(1/2, 1/2)
plotmixt(mus=mus, Sigmas=Sigmas, props=props, display="filled.contour", lwd=1)

## trivariate
mus <- rbind(c(0,0,0), c(-1,0.5,1.5))
Sigma <- matrix(c(1, 0.7, 0.7, 0.7, 1, 0.7, 0.7, 0.7, 1), nr=3, nc=3)
Sigmas <- rbind(Sigma, Sigma)
props <- c(1/2, 1/2)
plotmixt(mus=mus, Sigmas=Sigmas, props=props, dfs=c(11,8), dist="t")
```

---

pre.transform *Pre-sphering and pre-scaling*

Description

Pre-sphered or pre-scaled version of data.

Usage

```
pre.sphere(x, mean.centred=FALSE)
pre.scale(x, mean.centred=FALSE)
```

Arguments

x	matrix of data values
mean.centred	flag to centre the data values to have zero mean. Default is FALSE.

Details

For pre-scaling, the data values are pre-multiplied by $\mathbf{S}^{-1/2}$ and for pre-scaling, by $\mathbf{S}_D^{-1/2}$ where $\mathbf{S}$ is the sample variance and $\mathbf{S}_D$ is $\mathrm{diag}\,(S_1^2, S_2^2, \ldots, S_d^2)$ where S_i^2 is the i-th marginal sample variance.

Value

Pre-sphered or pre-scaled version of data. These pre-transformations are required for implementing the plug-in Hpi selectors and the smoothed cross validation Hscv selectors.

Examples

```
data(unicef)
unicef.sp <- pre.sphere(as.matrix(unicef))
```

quake *Geographical locations of earthquakes and tectonic plates*

Description

The quake data set contains the geographical locations of severe earthquakes in the years 100 and 2016 inclusive. The plate data set contains the geographical locations of the tectonic plate boundaries.

Usage

```
data(quake)
data(plate)
data(quakesf)
data(platesf)
```

Format

–For quake, a matrix with 5871 rows and 5 columns. Each row corresponds to an earthquake. The first column is the year (negative years indicate B.C.E.), the second is the longitude (decimal degrees), the third is the latitude (decimal degrees), the fourth is the depth beneath the Earth's surface (km), the fifth is a flag for the location inside the circum-Pacific belt (aka Pacific Ring of Fire). quakesf is a WGS84 sf version with a point geometry.

–For plate, a matrix with 6276 rows and 3 columns. Each row corresponds to an location of the tectonic plate boundaries. The first is the longitude, the second is the latitude, the third is the label of the tectonic plate. platesf is a WGS84 sf spatial version with a multipolygon geometry, where the individual plate line segments have been merged into a single multipolygon.

Source

Alhenius, H., Nordpil and Bird, P. (2014). World Tectonic Plates and Boundaries. https://github.com/fraxen/tectonicplates. Accessed 2021-03-11.

Bird, P. (2003) An updated digital model of plate boundaries, *Geochemistry, Geophysics, Geosystems* **4(3)**, 1-52. 1027.

NGDC/WDS (2017) Global significant earthquake database, National Geophysical Data Center, NOAA, doi:10.7289/V5TD9V7K. National Geophysical Data Center/World Data Service. Accessed 2017-03-30.

rkde *Derived quantities from kernel density estimates*

Description

Derived quantities from kernel density estimates.

Usage

```
dkde(x, fhat)
pkde(q, fhat)
qkde(p, fhat)
rkde(n, fhat, positive=FALSE)
```

Arguments

x, q	vector of quantiles
p	vector of probabilities
n	number of observations
positive	flag to compute KDE on the positive real line. Default is FALSE.
fhat	kernel density estimate, object of class kde

Details

pkde uses the trapezoidal rule for the numerical integration. rkde uses Silverman (1986)'s method to generate a random sample from a KDE.

Value

For the 1-d kernel density estimate fhat, pkde computes the cumulative probability for the quantile q, qkde computes the quantile corresponding to the probability p.

For any kernel density estimate, dkde computes the density value at x (it is an alias for predict.kde), rkde computes a random sample of size n.

References

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

Examples

```
set.seed(8192)
x <- rnorm.mixt(n=10000, mus=0, sigmas=1, props=1)
fhat <- kde(x=x)
p1 <- pkde(fhat=fhat, q=c(-1, 0, 0.5))
qkde(fhat=fhat, p=p1)
y <- rkde(fhat=fhat, n=100)
```

```
x <- rmvnorm.mixt(n=10000, mus=c(0,0), Sigmas=invvech(c(1,0.8,1)))
fhat <- kde(x=x)
y <- rkde(fhat=fhat, n=1000)
fhaty <- kde(x=y)
plot(fhat, col=1)
plot(fhaty, add=TRUE, col=2)
```

---

tempb *Daily temperature*

Description

This data set contains the daily minimum and maximum temperatures from the weather station in Badajoz, Spain, from 1 January 1955 to 31 December 2015.

Usage

```
data(tempb)
```

Format

A matrix with 21908 rows and 5 columns. Each row corresponds to a daily measurement. The first column is the year (yyyy), the second is the month (mm), the third is the day (dd), the fourth is the minimum temperature (degrees Celsius), the fifth is the maximum temperature (degrees Celsius).

Source

Menne, M. J., Durre, I., Vose, R. S., Gleason, B. E. & Houston, T. (2012) An overview of the global historical climatology network-daily database, *Journal of Atmospheric and Oceanic Technology* **429**, 897 - 910. https://climexp.knmi.nl/selectdailyseries.cgi. Accessed 2016-10-20.

---

unicef *Unicef child mortality - life expectancy data*

Description

This data set contains the number of deaths of children under 5 years of age per 1000 live births and the average life expectancy (in years) at birth for 73 countries with GNI (Gross National Income) less than 1000 US dollars per annum per capita.

Usage

```
data(unicef)
```

Format

A matrix with 2 columns and 73 rows. Each row corresponds to a country. The first column is the under 5 mortality rate and the second is the average life expectancy.

Source

Unicef (2003). *State of the World's Children Report 2003*, Oxford University Press, for Unicef.

`vector` *Vector and vector half operators*

Description

The vec (vector) operator takes a $d \times d$ matrix and stacks the columns into a single vector of length d^2. The vech (vector half) operator takes a symmetric $d \times d$ matrix and stacks the lower triangular half into a single vector of length $d(d+1)/2$. The functions invvec and invvech are the inverses of vec and vech i.e. they form matrices from vectors.

Usage

```
vec(x, byrow=FALSE)
vech(x)
invvec(x, ncol, nrow, byrow=FALSE)
invvech(x)
```

Arguments

`x`	vector or matrix
`ncol`, `nrow`	number of columns and rows for inverse of vech
`byrow`	flag for stacking row-wise or column-wise. Default is FALSE.

References

Magnus, J.R. & Neudecker H.M. (2007) *Matrix Differential Calculus with Applications in Statistics and Econometrics (3rd edition)*, Wiley & Sons. Chichester.

Examples

```
x <- matrix(1:9, nrow=3, ncol=3)
vec(x)
invvec(vec(x))
```

vkde *Variable kernel density estimate.*

Description

Variable kernel density estimate for 2-dimensional data.

Usage

```
kde.balloon(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
   binned, bgridsize, w, compute.cont=TRUE, approx.cont=TRUE, verbose=FALSE)
kde.sp(x, H, h, gridsize, gridtype, xmin, xmax, supp=3.7, eval.points,
   binned, bgridsize, w, compute.cont=TRUE, approx.cont=TRUE, verbose=FALSE)
```

Arguments

x	matrix of data values
H	bandwidth matrix. If this missing, Hns is called by default.
h	not yet implemented
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin, xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	vector or matrix of points at which estimate is evaluated
binned	flag for binned estimation.
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is TRUE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
verbose	flag to print out progress information. Default is FALSE.

Details

The balloon density estimate kde.balloon employs bandwidths which vary at each estimation point (Loftsgaarden & Quesenberry, 1965). There are as many bandwidths as there are estimation grid points. The default bandwidth is Hns(,deriv.order=2) and the subsequent bandwidths are derived via a minimal MSE formula.

The sample point density estimate kde.sp employs bandwidths which vary for each data point (Abramson, 1982). There are as many bandwidths as there are data points. The default bandwidth is Hns(,deriv.order=4) and the subsequent bandwidths are derived via the Abramson formula.

Value

A variable kernel density estimate for bounded data is an object of class kde.

References

Abramson, I. S. (1982) On bandwidth variation in kernel estimates - a square root law. *Annals of Statistics*, **10**, 1217-1223.

Loftsgaarden, D. O. & Quesenberry, C. P. (1965) A nonparametric estimate of a multivariate density function. *Annals of Mathematical Statistics*, **36**, 1049-1051.

See Also

kde, plot.kde

Examples

```
data(worldbank)
wb <- as.matrix(na.omit(worldbank[,4:5]))
xmin <- c(-70,-35); xmax <- c(35,70)
fhat <- kde(x=wb, xmin=xmin, xmax=xmax)
fhat.sp <- kde.sp(x=wb, xmin=xmin, xmax=xmax)
zmax <- max(fhat.sp$estimate)
plot(fhat, display="persp", box=TRUE, phi=20, thin=1, border=grey(0,0.2), zlim=c(0,zmax))
plot(fhat.sp, display="persp", box=TRUE, phi=20, thin=1, border=grey(0,0.2), zlim=c(0,zmax))
## Not run:
fhat.ball <- kde.balloon(x=wb, xmin=xmin, xmax=xmax)
plot(fhat.ball, display="persp", box=TRUE, phi=20, zlim=c(0,zmax))
## End(Not run)
```

---

worldbank *Development indicators from the World Bank Group*

Description

This data set contains six development indicators for national entities for the year 2011, which is the latest year for which they are consistently available.

Usage

```
data(worldbank)
```

Format

A matrix with 7 columns and 218 rows. Each row corresponds to a country. The first column is the country, the second is the per capita carbon dioxide emissions (thousands Kg), the third is the per capita GDP (thousands of current USD), the fourth is the annual GDP growth rate (%), the fifth is the annual inflation rate (%), the sixth is the percentage of internet users in the population (%), the seventh is the added value agricultural production as a ratio of the total GDP (%).

Source

World Bank Group (2016) World development indicators. http://databank.worldbank.org/data/reports.aspx?source=world-development-indicators. Accessed 2016-10-03.

Index