

Package ‘ks’

October 16, 2009

Version 1.6.8

Date 2009/10/08

Title Kernel smoothing

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

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

Depends R (>= 1.4.0), KernSmooth, mvtnorm

Suggests rgl (>= 0.66), misc3d (>= 0.4-0), kernlab

Description Kernel density estimators and kernel discriminant analysis for multivariate data

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-10-16 11:11:33

R topics documented:

binning	2
compare, compare.kda.diag.cv, compare.kda.cv	3
contourLevels	5
dkde, pkde, qkde, rkde	6
dmvnorm.mixt, rmvnorm.mixt	7
dmtv.mixt, rmtv.mixt	8
dnorm.mixt, rnorm.mixt	9
drvkde	10
Hamise.mixt, Hmise.mixt, Hamise.mixt.diag, Hmise.mixt.diag, amise.mixt, ise.mixt, mise.mixt	11
Hbcv, Hbcv.diag	13
Hlscv, Hlscv.diag, hlscv	14
Hpi, Hpi.diag, hpi	15
Hscv, Hscv.diag, hscv	17
kda, Hkda, Hkda.diag, hkda	18

kda.kde	20
kde	21
ks	23
plot.kda.kde	24
plot.kde	27
plotmixt	29
pre.scale, pre.sphere	31
unicef	32
vec, vech, invvec, invvech	32

Index	34
--------------	-----------

binning	<i>Linear binning for multivariate data</i>
---------	---

Description

Linear binning for 1- to 4-dimensional data.

Usage

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

Arguments

x	matrix of data values
H	bandwidth matrix
h	scalar bandwidth
xmin	vector of minimum values for grid
xmax	vector of maximum values for grid
supp	effective support for standard normal is $[-supp, supp]$
bgridsize	vector of binning grid sizes
w	vector of weights (non-negative and sum is equal to sample size)

Details

Code is used courtesy of Matt 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 (<code>d=1</code>) or list (<code>d>2</code>) of grid points in each dimension

References

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

Examples

```
data(unicef)
ubinned <- binning(x=unicef)
ubinned <- binning(x=unicef, xmin=c(0, 20), xmax=c(350, 100))
```

compare, compare.kda.diag.cv, compare.kda.cv
Comparisons for kernel discriminant analysis

Description

Comparisons for kernel discriminant analysis.

Usage

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

Arguments

x	matrix of training data values
x.group	vector of group labels for training data
est.group	vector of estimated group labels
bw	"plugin" = plug-in, "lscv" = LSCV, "scv" = SCV
Hstart	(stacked) matrix of initial bandwidth matrices
prior.prob	vector of prior probabilities
by.group	flag to give results also within each group
trace	flag for printing messages in command line to trace the execution
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes - only required if binned=TRUE
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 you have prior probabilities then set `prior.prob` to these. Otherwise `prior.prob=NULL` is the default i.e. use the sample proportions as estimates of the prior probabilities.

If `trace=TRUE`, a message is printed in the command line indicating that it's processing the *i*-th data item: cross-validated estimates may take a long time to execute.

Value

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

<code>cross</code>	cross-classification table with the rows indicating the true group and the columns the estimated group
<code>error</code>	misclassification rate (MR)

In the case where we have test data that is independent of the training data, `compare` computes

$$\text{MR} = \frac{\text{number of points wrongly classified}}{\text{total number of points}}.$$

In the case where we don't have independent test data e.g. we are classifying the training data set itself, then the cross validated estimate of MR is more appropriate. See Silverman (1986). These are implemented as `compare.kda.cv` (full 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

- Silverman, B. W. (1986) *Data Analysis for Statistics and Data Analysis*. Chapman & Hall. London.
- Simonoff, J. S. (1996) *Smoothing Methods in Statistics*. Springer-Verlag. New York
- Venables, W.N. & Ripley, B.D. (1997) *Modern Applied Statistics with S-PLUS*. Springer-Verlag. New York.

See Also

[kda.kde](#)

Examples

```
### univariate example -- independent test data
x <- c(rnorm.mixt(n=100, mus=1, sigmas=1, props=1),
      rnorm.mixt(n=100, mus=-1, sigmas=1, props=1))
x.gr <- rep(c(1,2), times=c(100,100))
y <- c(rnorm.mixt(n=100, mus=1, sigmas=1, props=1),
      rnorm.mixt(n=100, mus=-1, sigmas=1, props=1))
kda.gr <- kda(x, x.gr, hs=sqrt(c(0.09, 0.09)), y=y)
compare(x.gr, kda.gr)
compare(x.gr, kda.gr, by.group=TRUE)
```

```
### bivariate example - restricted iris dataset, dependent test data
library(MASS)
data(iris)
ir <- iris[,c(1,2)]
ir.gr <- iris[,5]
compare.kda.cv(ir, ir.gr, bw="plug-in", pilot="samse")
```

contourLevels

Contour levels for kde and kda.kde objects

Description

Contour levels for kde and kda.kde objects.

Usage

```
contourLevels(x, ...)

## S3 method for class 'kde':
contourLevels(x, prob, cont, nlevels=5, ...)

## S3 method for class 'kda.kde':
contourLevels(x, prob, cont, nlevels=5, ...)
```

Arguments

x	an object of class kde or kda.kde
prob	vector of probabilities corresponding to highest density regions
cont	vector of percentages which correspond to the complement of prob
nlevels	number of pretty contour levels
...	other parameters for contour

Details

The most straightforward is to specify `prob`. Heights of the corresponding highest density region with probability `prob` are computed.

The `cont` parameter here is consistent with `cont` parameter from `plot.kde` and `plot.kda.kde` i.e. $cont = (1 - prob) * 100\%$.

If both `prob` and `cont` are missing then a pretty set of `nlevels` contours are computed.

Value

For `kde` objects, returns vector of heights. For `kda.kde` objects, returns a list of vectors, one for each training group.

See Also

[contour](#), [contourLines](#)

Examples

```
## kde
x <- rmvnorm.mixt(n=100, mus=c(0,0), Sigmas=diag(2), props=1)
Hx <- Hpi(x)
fhatx <- kde(x=x, H=Hx)
lev1 <- contourLevels(fhatx, prob=c(0.25, 0.5, 0.75))
lev2 <- contourLevels(fhatx, cont=c(75, 50, 25))      ## lev1 == lev2

## kda.kde
library(MASS)
data(iris)
ir <- iris[,1]
ir.gr <- iris[,5]
kda.fhat <- kda.kde(ir, ir.gr, hs=sqrt(c(0.01, 0.04, 0.07)))
contourLevels(kda.fhat, prob=c(0.25, 0.5, 0.75))
```

dkde, pkde, qkde, rkde

Functions for 1-dimensional kernel density estimates

Description

Functions for 1-dimensional kernel density estimates.

Usage

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

Arguments

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

Details

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

Value

For the kernel density estimate `fhat`, `pkde` computes the cumulative probability for the quantile `q`, `qkde` computes the quantile corresponding to the probability `p`, `dkde` computes the density value at `x` and `rkde` computes a random sample of size `n`.

References

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

Examples

```
x <- rnorm.mixt(n=10000, mus=0, sigmas=1, props=1)
fhat <- kde(x=x, h=hpi(x))
p1 <- pkde(fhat=fhat, q=c(-1, 0, 0.5))
qkde(fhat=fhat, p=p1)      ## should be close to c(-1, 0, 0.5)

x1 <- rkde(fhat, n=100)
plot(fhat)
fhat1 <- kde(x=x1, h=hpi(x1))
plot(fhat1, add=TRUE, col=2)
fhat2 <- dkde(x=x1, fhat=fhat1)
points(x1, fhat2, col=3)
```

dmvnorm.mixt, rmvnorm.mixt

Multivariate normal mixture distribution

Description

Random generation and density values from multivariate normal mixture distribution.

Usage

```
dmvnorm.mixt(x, mus, Sigmas, props)
rmvnorm.mixt(n=100, mus=c(0,0), Sigmas=diag(2), props=1, mixt.label=FALSE)
```

Arguments

<code>n</code>	number of random variates
<code>x</code>	matrix of quantiles
<code>mus</code>	(stacked) matrix of mean vectors
<code>Sigmas</code>	(stacked) matrix of variance matrices
<code>props</code>	vector of mixing proportions
<code>mixt.label</code>	flag to output numeric label indicating the mixture component. Default is FALSE.

Details

`rmvnorm.mixt` and `dmvnorm.mixt` are based on the `rmvnorm` and `dmvnorm` functions from the `mvtnorm` library.

Value

Multivariate normal mixture random vectors and density values.

See Also

[rmvt.mixt](#), [dmvt.mixt](#), [dnorm.mixt](#), [rnorm.mixt](#)

Examples

```
mus <- rbind(c(-3/2,0), c(3/2,0))
Sigmas <- rbind(diag(c(1/16, 1)), rbind(c(1/16, 1/18), c(1/18, 1/16)))
props <- c(2/3, 1/3)
x <- rmvnorm.mixt(1000, mus, Sigmas, props)
dens <- dmvnorm.mixt(x, mus, Sigmas, props)
```

`dmvt.mixt, rmvt.mixt`

Multivariate t mixture distribution

Description

Random generation and density values from multivariate t mixture distribution.

Usage

```
rmvt.mixt(n=100, mus=c(0,0), Sigmas=diag(2), dfs=7, props=1)
dmvt.mixt(x, mus, Sigmas, dfs, props)
```

Arguments

<code>n</code>	number of random variates
<code>x</code>	matrix of quantiles
<code>mus</code>	(stacked) matrix of location vectors
<code>Sigmas</code>	(stacked) matrix of dispersion matrices
<code>dfs</code>	vector of degrees of freedom
<code>props</code>	vector of mixing proportions

Details

rmvt.mixt and dmvt.mixt are based on the rmvt and dmvt functions from the mvtnorm library.

The formula for a d-variate t density with location vector $\boldsymbol{\mu}$, dispersion matrix $\boldsymbol{\Sigma}$ and df degrees of freedom is

$$\frac{\Gamma((df + d)/2)}{(df\pi)^{d/2}\Gamma(df/2)|\boldsymbol{\Sigma}^{1/2}|} \left[1 + \frac{1}{df}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \right]^{-(d+df)/2}.$$

Value

Multivariate t mixture random vectors and density values.

See Also

[rmvnorm.mixt](#), [dmvnorm.mixt](#)

Examples

```
mus <- rbind(c(-3/2,0), c(3/2,0))
Sigmas <- rbind(diag(c(1/16, 1)), rbind(c(1/16, 1/18), c(1/18, 1/16)))
props <- c(2/3, 1/3)
dfs <- c(7,3)
x <- rmvt.mixt(1000, mus, Sigmas, dfs, props)
dens <- dmvt.mixt(x, mus, Sigmas, dfs, props)
```

dnorm.mixt, rnorm.mixt

Univariate normal mixture distribution

Description

Random generation and density values from univariate normal mixture distribution.

Usage

```
dnorm.mixt(x, mus=0, sigmas=1, props=1)
rnorm.mixt(n=100, mus=0, sigmas=1, props=1, mixt.label=FALSE)
```

Arguments

n	number of random variates
x	vector of quantiles
mus	vector of means
sigmas	vector of standard deviations
props	vector of mixing proportions
mixt.label	flag to output numeric label indicating the mixture component. Default is FALSE.

Value

Univariate normal mixture random vectors and density values.

Examples

```
x <- rnorm.mixt(1000, mus=c(-1,1), sigmas=c(0.5, 0.5), props=c(1/2, 1/2))
dens <- dnorm.mixt(x, mus=c(-1,1), sigmas=c(0.5, 0.5), props=c(1/2, 1/2))
```

 drvkde

Kernel density derivative estimation

Description

Compute kernel density derivative estimates and standard errors for multivariate data.

Usage

```
drvkde(x, drv, bandwidth, gridsize, range.x, binned=FALSE, se=TRUE, w)
```

Arguments

x	data matrix or matrix of binning counts
drv	vector of derivative indices
bandwidth	vector of bandwidths
gridsize	vector of grid sizes
range.x	list of vector of ranges for x
binned	TRUE if x is binned counts or FALSE if x is data matrix
se	flag for computing the standard error of kernel estimate
w	vector of weights (non-negative and sum is equal to sample size)

Details

The estimates and standard errors are computed over a grid of binned counts `x.grid`. If the binned counts are not supplied then they are computed inside this function.

If `gridsize` and `range.x` are not supplied, they are computed inside this function.

Value

Returns a list with fields

`x.grid` - grid points

`est` - kernel estimate of partial derivative of density function indicated by `drv`

`se` - estimate of standard error of `est` (if `se=TRUE`).

Author(s)

M.P. Wand

References

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

Examples

```
## univariate
x <- rnorm(100)
fhat <- drvkde(x=x, drv=0, bandwidth=0.1) ## KDE of f
fhat1 <- drvkde(x=x, drv=1, bandwidth=0.1) ## KDE of df/dx
```

Hamise.mixt, Hmise.mixt, Hamise.mixt.diag, Hmise.mixt.diag, amise.mixt, ise.mixt, mise.mixt, Hamise.mixt.diag
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)

ise.mixt(x, H, mus, Sigmas, props, h, sigmas, deriv.order=0)
mise.mixt(H, mus, Sigmas, props, samp, h, sigmas, deriv.order=0)
amise.mixt(H, mus, Sigmas, props, samp, h, sigmas, deriv.order=0)
```

Arguments

mus (stacked) matrix of mean vectors/vector of means
sigmas, Sigmas vector of standard deviations/(stacked) matrix of variance matrices
props vector of mixing proportions
samp sample size

12 *Hamise.mixt, Hmise.mixt, Hamise.mixt.diag, Hmise.mixt.diag, amise.mixt, ise.mixt, mise.mixt*

`hstart, Hstart` initial bandwidth (matrix), used in numerical optimisation
`deriv.order` derivative order
`x` matrix of data values
`h, H` bandwidth (matrix)

Details

For normal mixture densities, ISE, MISE and AMISE have exact formulas for all dimensions. See Chacón, Duong & Wand (2008).

If `Hstart` is not given then it defaults to $k \cdot \text{var}(x)$ where $k = \left[\frac{4}{n(d+2r+2)} \right]^{2/(d+2r+4)}$, n = sample size, d = dimension of data, r = derivative order. The default for `hstart` is the square root of this expression.

Value

- Full MISE- or AMISE-optimal bandwidth matrix. Diagonal forms of these matrices are not available.
- ISE, MISE or AMISE value.

Note

ISE is a random variable that depends on the data x . MISE and AMISE are non-random and don't depend on the data.

References

Chacón J.E., Duong, T. & Wand, M.P. (2009). Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*. Accepted.

Examples

```
## 1-d
mus <- c(0, 2)
sigmas <- c(1, sqrt(0.7))
props <- c(1/2, 1/2)
samp <- 1000
h <- hmise.mixt(mus, sigmas, props, samp, deriv.order=0)
x <- rnorm.mixt(n=samp, mus=mus, sigmas=sigmas, props=props)
ise.mixt(x=x, h=h, mus=mus, sigmas=sigmas, props=props)
mise.mixt(h=h, mus=mus, sigmas=sigmas, props=props, samp=samp)

## 2-d
mus <- rbind(c(0,0), c(2,2))
Sigma <- matrix(c(1, 0.7, 0.7, 1), nr=2, nc=2)
Sigmas <- rbind(Sigma, Sigma)
props <- c(1/2, 1/2)
samp <- 100
H <- Hamise.mixt(mus, Sigmas, props, samp, deriv.order=2)
```

```
x <- rmvnorm.mixt(n=samp, mus=mus, Sigmas=Sigmas, props=props)
ise.mixt(x=x, H=H, mus=mus, Sigmas=Sigmas, props=props, deriv.order=2)
amise.mixt(H=H, mus=mus, Sigmas=Sigmas, props=props, samp=samp, deriv.order=2)
```

Hbcv, Hbcv.diag	<i>Biased cross-validation (BCV) bandwidth matrix selector for bivariate data</i>
-----------------	---

Description

BCV bandwidth matrix for bivariate data.

Usage

```
Hbcv(x, whichbcv=1, Hstart)
Hbcv.diag(x, whichbcv=1, Hstart)
```

Arguments

x	matrix of data values
whichbcv	1 = BCV1, 2 = BCV2. See details below
Hstart	initial bandwidth matrix, used in numerical optimisation

Details

Use `Hbcv` for full bandwidth matrices and `Hbcv.diag` for diagonal bandwidth matrices. These selectors are only available for bivariate data.

There are two types of BCV criteria considered here. They are known as BCV1 and BCV2, from Sain, Baggerly & Scott (1994) and they only differ slightly. These BCV surfaces can have multiple minima and so it can be quite difficult to locate the most appropriate minimum.

If `Hstart` is not given then it defaults to $k * \text{var}(x)$ where $k = \left[\frac{4}{n(d+2)} \right]^{2/(d+4)}$, n = sample size, d = dimension of data.

Value

BCV bandwidth matrix.

Note

It can be difficult to find an appropriate (local) minimum of the BCV criterion. Some times, there can be no local minimum at all so there may be no finite BCV selector.

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.

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

See Also

[Hlscv](#), [Hscv](#)

Examples

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

```
Hlscv, Hlscv.diag, 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)
Hlscv.diag(x, Hstart, binned=FALSE, bgridsize)
hlscv(x, binned=TRUE, bgridsize)
```

Arguments

<code>x</code>	vector or matrix of data values
<code>Hstart</code>	initial bandwidth matrix, used in numerical optimisation
<code>binned</code>	flag for binned kernel estimation
<code>bgridsize</code>	vector of binning grid sizes - required only if <code>binned=TRUE</code>

Details

`hlscv` is the univariate SCV selector of Bowman (1984) and Rudemo (1982). `Hlscv` is a multivariate generalisation of this.

Use `Hlscv` for full bandwidth matrices and `Hlscv.diag` for diagonal bandwidth matrices.

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 $k \cdot \text{var}(x)$ where $k = \left[\frac{4}{n(d+2)} \right]^{2/(d+4)}$, n = sample size, d = dimension of data.

Value

LSCV bandwidth.

References

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

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

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

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

[Hbcv](#), [Hscv](#)

Examples

```
library(MASS)
data(forbes)
Hlscv(forbes)
Hlscv.diag(forbes, binned=TRUE)
hlscv(forbes$bp)
```

Hpi, Hpi.diag, hpi *Plug-in bandwidth selector*

Description

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

Usage

```
Hpi(x, nstage=2, pilot="samse", pre="sphere", Hstart,
    binned=FALSE, bgridsize, amise=FALSE)
Hpi.diag(x, nstage=2, pilot="amse", pre="scale", Hstart,
    binned=FALSE, bgridsize)
hpi(x, nstage=2, binned=TRUE, bgridsize)
```

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" = unconstrained pilot bandwidth matrix
pre	"scale" = pre-scaling, "sphere" = pre-sphering
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes - required only if binned=TRUE
amise	flag for returning estimated AMISE

Details

hpi is the univariate plug-in selector of Wand & Jones (1994). Hpi is a multivariate generalisation of this.

Use Hpi for full bandwidth matrices and Hpi.diag for diagonal bandwidth matrices. 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 pilot bandwidths are available for $d = 1, \dots, 5$ (but are extremely computationally intensive for the latter dimensions). See Chacón & Duong (2008).

For $d = 1$, the selector hpi is exactly the same as **KernSmooth**'s `dpik`. This is always computed as binned estimator. For $d = 2, 3, 4$ and `binned=TRUE`, estimates are computed over a binning grid defined by `bgridsize`. Otherwise it's computed exactly.

For details on the pre-transformations in `pre`, see `pre.sphere` and `pre.scale`.

If `Hstart` is not given then it defaults to $k * \text{var}(x)$ where $k = \left[\frac{4}{n(d+2)} \right]^{2/(d+4)}$, n = sample size, d = dimension of data.

Value

Plug-in bandwidth. If `amise=TRUE` then the plug-in bandwidth plus the estimated AMISE is returned in a list.

References

- Chacón, J.E. & Duong, T. (2008) Multivariate plug-in bandwidth selection with unconstrained pilot matrices. *Test*. Accepted.
- 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 plugin bandwidth selection. *Computational Statistics*, **9**, 97-116.

Examples

```

data(unicef)
Hpi(unicef)
Hpi(unicef, pilot="unconstr")
Hpi.diag(unicef, binned=TRUE)
hpi(unicef[,1])

```

```
Hscv, Hscv.diag, hscv
```

Smoothed cross-validation (SCV) bandwidth selector

Description

SCV bandwidth for 1- to 6-dimensional data.

Usage

```

Hscv(x, pre="sphere", Hstart, binned=FALSE, bgridsize)
Hscv.diag(x, pre="scale", Hstart, binned=FALSE, bgridsize)
hscv(x, nstage=2, binned=TRUE, bgridsize, plot=FALSE)

```

Arguments

x	vector or matrix of data values
pre	"scale" = pre-scaling, "sphere" = pre-sphering
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes - required only if binned=TRUE
nstage	number of stages in the SCV bandwidth selector (1 or 2) (1-d only)
plot	flag to display plot of SCV(h) vs h (1-d only)

Details

hsv is the univariate SCV selector of Jones, Marron & Park (1991). Hscv is a multivariate generalisation of this.

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 $d = 1, 2, 3, 4$ and binned=TRUE, the estimates are computed over a binning grid defined by bgridsize. Otherwise it's computed exactly.

For details on the pre-transformations in pre, see [pre.sphere](#) and [pre.scale](#).

If Hstart is not given then it defaults to $k * \text{var}(x)$ where $k = \left[\frac{4}{n(d+2)} \right]^{2/(d+4)}$, n = sample size, d = dimension of data.

Value

SCV bandwidth.

References

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

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

See Also

[Hlscv](#), [Hbcv](#), [Hpi](#)

Examples

```
data(unicef)
Hscv(unicef)
Hscv.diag(unicef, binned=TRUE)
hscv(unicef[,1])
```

kda, Hkda, Hkda.diag, hkda

Kernel discriminant analysis for multivariate data

Description

Kernel discriminant analysis for 1- to 6-dimensional data.

Usage

```
Hkda(x, x.group, Hstart, bw="plugin", nstage=2, pilot="samse",
     pre="sphere", binned=FALSE, bgridsize)
Hkda.diag(x, x.group, bw="plugin", nstage=2, pilot="samse",
          pre="sphere", binned=FALSE, bgridsize)
hkda(x, x.group, bw="plugin", nstage=2, binned=TRUE, bgridsize)

kda(x, x.group, Hs, hs, y, prior.prob=NULL)
```

Arguments

x	matrix of training data values
x.group	vector of group labels for training data
y	matrix of test data
Hs	(stacked) matrix of bandwidth matrices
hs	vector of scalar bandwidths

prior.prob	vector of prior probabilities
bw	bandwidth: "plugin" = plug-in, "lscv" = LSCV, "scv" = SCV
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"amse" = AMSE pilot bandwidths, "samse" = single SAMSE pilot bandwidth
pre	"scale" = pre-scaling, "sphere" = pre-sphering
Hstart	(stacked) matrix of initial bandwidth matrices, used in numerical optimisation
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes - required only if binned=TRUE

Details

– The values that valid for `bw` are "plugin", "lscv" and "scv" for `Hkda`. These in turn call `Hpi`, `Hlscv` and `Hscv`. For plugin selectors, all of `nstage`, `pilot` and `pre` need to be set. For SCV selectors, currently `nstage=1` always but `pilot` and `pre` need to be set. For LSCV selectors, none of them are required. `Hkda.diag` makes analagous calls to diagonal selectors.

For $d = 1, 2, 3, 4$, 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`).

For $d = 1, 2, 3, 4$, and if `eval.points` is specified, then the density estimate is computed exactly at `eval.points`.

For $d > 4$, the kernel density estimate is computed exactly and `eval.points` must be specified.

For details on the pre-transformations in `pre`, see [pre.sphere](#) and [pre.scale](#).

– If you have prior probabilities then set `prior.prob` to these. Otherwise `prior.prob=NULL` is the default i.e. use the sample proportions as estimates of the prior probabilities.

Value

– 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 data group.

– The result from `kda` is a vector of group labels estimated via the kernel discriminant rule. If the test data `y` are given then these are classified. Otherwise the training data `x` are classified.

References

- Mardia, K.V., Kent, J.T. & Bibby J.M. (1979) *Multivariate Analysis*. Academic Press. London.
- Silverman, B. W. (1986) *Data Analysis for Statistics and Data Analysis*. Chapman & Hall. London.
- Simonoff, J. S. (1996) *Smoothing Methods in Statistics*. Springer-Verlag. New York
- Venables, W.N. & Ripley, B.D. (1997) *Modern Applied Statistics with S-PLUS*. Springer-Verlag. New York.

See Also

[compare](#), [compare.kda.cv](#), [kda.kde](#)

Examples

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

kda.kde	<i>Kernel density estimate for kernel discriminant analysis for multivariate data</i>
---------	---

Description

Kernel density estimate for kernel discriminant analysis for 1- to 6-dimensional data

Usage

```
kda.kde(x, x.group, Hs, hs, prior.prob=NULL, gridsize, xmin, xmax,
        supp=3.7, eval.points=NULL, binned=FALSE, bgridsize, w)
```

Arguments

x	matrix of training data values
x.group	vector of group labels for training data
Hs	(stacked) matrix of bandwidth matrices
hs	vector of scalar bandwidths
prior.prob	vector of prior probabilities
gridsize	vector of number of grid points
xmin	vector of minimum values for grid
xmax	vector of maximum values for grid
supp	effective support for standard normal is [-supp, supp]
eval.points	points at which density estimate is evaluated
binned	flag for binned kernel estimation
bgridsize	vector of binning grid sizes - only required if binned=TRUE
w	vector of weights (non-negative and sum is equal to sample size)

Details

For $d = 1, 2, 3, 4$, 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`).

For $d = 1, 2, 3, 4$, and if `eval.points` is specified, then the density estimate is computed exactly at `eval.points`.

For $d > 4$, the kernel density estimate is computed exactly and `eval.points` must be specified.

If you have prior probabilities then set `prior.prob` to these. Otherwise `prior.prob=NULL` is the default i.e. use the sample proportions as estimates of the prior probabilities.

The default `xmin` is $\min(x) - H_{\max} * \text{supp}$ and `xmax` is $\max(x) + H_{\max} * \text{supp}$ where H_{\max} is the maximum of the diagonal elements of H .

The default weights `w` is a vector of all ones.

Value

The kernel density estimate for kernel discriminant analysis is based on [kde](#), one density estimate for each group.

The result from `kda.kde` is a density estimate for discriminant analysis is an object of class `kda.kde` which is a list with 6 fields

<code>x</code>	data points - same as input
<code>x.group</code>	group labels - same as input
<code>eval.points</code>	points that density estimate is evaluated at
<code>estimate</code>	density estimate at <code>eval.points</code>
<code>prior.prob</code>	prior probabilities
<code>H</code>	bandwidth matrices (>1-d only) or
<code>h</code>	bandwidths (1-d only)
<code>w</code>	weights

References

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

See Also

[plot.kda.kde](#)

Examples

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

kde

Kernel density estimate for multivariate data

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=FALSE, bgridsize, positive=FALSE, adj.positive, w)
```

Arguments

<code>x</code>	matrix of data values
<code>H</code>	bandwidth matrix
<code>h</code>	scalar bandwidth
<code>gridsize</code>	vector of number of grid points
<code>gridtype</code>	not yet implemented
<code>xmin</code>	vector of minimum values for grid
<code>xmax</code>	vector of maximum values for grid
<code>supp</code>	effective support for standard normal is $[-supp, supp]$
<code>eval.points</code>	points at which density estimate is evaluated
<code>binned</code>	flag for binned estimation (default is FALSE)
<code>bgridsize</code>	vector of binning grid sizes - required if <code>binned=TRUE</code>
<code>positive</code>	flag if 1-d data are positive (default is FALSE)
<code>adj.positive</code>	adjustment added to data i.e. when <code>positive=TRUE</code> KDE is carried out on $\log(x + adj.positive)$. Default is the minimum of <code>x</code> .
<code>w</code>	vector of weights (non-negative and sum is equal to sample size)

Details

For $d = 1, 2, 3, 4$, 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`).

For $d = 1, 2, 3, 4$, and if `eval.points` is specified, then the density estimate is computed exactly at `eval.points`.

For $d > 4$, the kernel density estimate is computed exactly and `eval.points` must be specified.

The default `xmin` is $\min(x) - H_{\max} * supp$ and `xmax` is $\max(x) + H_{\max} * supp$ where H_{\max} is the maximum of the diagonal elements of `H`.

The default weights `w` is a vector of all ones.

Value

Kernel density estimate is an object of class `kde` which is a list with 4 fields

<code>x</code>	data points - same as input
<code>eval.points</code>	points at which the density estimate is evaluated
<code>estimate</code>	density estimate at <code>eval.points</code>
<code>H</code>	bandwidth matrix
<code>h</code>	scalar bandwidth (1-d only)
<code>w</code>	weights

References

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

See Also[plot.kde](#)**Examples**

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

 ks

 ks

Description

Kernel density estimation and kernel discriminant analysis for data from 1- to 6-dimensions, with display functions.

Details

There are three main types of functions in this package: (a) computing bandwidth selectors, (b) computing kernel estimators and (c) displaying kernel estimators.

For the bandwidth matrix selectors, there are several varieties:

- (i) plug-in [hpi](#) (1-d); [Hpi](#), [Hpi.diag](#) (2- to 6-d)
- (ii) least squares (or unbiased) cross validation (LSCV or UCV) [Hlscv](#), [Hlscv.diag](#) (2- to 6-d)
- (iii) biased cross validation (BCV) [Hbcv](#), [Hbcv.diag](#) (2- to 6-d)
- (iv) smoothed cross validation (SCV) [hscv](#) (1-d); [Hscv](#), [Hscv.diag](#) (2- to 6-d)
- (v) normal scale selectors [hmise.mixt](#), [hamise.mixt](#) (1-d); and [Hmise.mixt](#), [Hamise.mixt](#) (2- to 6-d).

For kernel density estimation, the main function is [kde](#). For kernel discriminant analysis, it's [kda.kde](#).

For display, `plot` via ([plot.kde](#) and [plot.kda.kde](#)) sends to a graphics window the results of density estimation or discriminant analysis.

Binned kernel estimation is available for $d = 1, 2, 3, 4$.

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

Author(s)

Tarn Duong for most of the package. Matt Wand for the binned estimation, univariate plug-in selector and density derivative estimator code. Jose E. Chacón for the unconstrained pilot functional estimation and (A)MISE-optimal selectors for normal mixture densities code.

References

- Bowman, A. & Azzalini, A. (1997) *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, Oxford.
- Chacón, J.E. & Duong, T. (2008) Multivariate plug-in bandwidth selection with unconstrained pilot matrices. *Test*. Accepted.
- Duong, T. (2004) *Bandwidth Matrices for Multivariate Kernel Density Estimation*. Ph.D. Thesis, University of Western Australia.
- Duong, T. & Hazelton, M.L. (2003) Plug-in bandwidth matrices for bivariate kernel density estimation. *Journal of Nonparametric Statistics*, **15**, 17-30.
- Duong, T. & Hazelton, M.L. (2005) Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, **32**, 485-506.
- Sain, S.R., Baggerly, K.A. & Scott, D.W. (1994) Cross-validation of multivariate densities. *Journal of the American Statistical Association*. **82**, 1131-1146.
- Scott, D.W. (1992) *Multivariate Density Estimation: Theory, Practice, and Visualization*. 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. (1994) Multivariate plugin bandwidth selection. *Computational Statistics*, **9**, 97-116.
- Wand, M.P. & Jones, M.C. (1995) *Kernel Smoothing*. Chapman & Hall/CRC, London.

See Also

sm, KernSmooth

plot.kda.kde

Kernel discriminant analysis plot for 1- to 3-dimensional data

Description

Kernel discriminant analysis plot for 1- to 3-dimensional data.

Usage

```
## univariate
## S3 method for class 'kda.kde':
plot(x, y, y.group, prior.prob=NULL, xlim, ylim,
      xlab="x", ylab="Weighted density function", drawpoints=FALSE,
      col, ptc, jitter=TRUE, ...)

## bivariate
## S3 method for class 'kda.kde':
```

```

plot(x, y, y.group, prior.prob=NULL, cont=c(25,50,75),
     abs.cont, xlim, ylim, xlab, ylab, drawpoints=FALSE,
     drawlabels=TRUE, col, partcol, ptc, ...)

## trivariate
## S3 method for class 'kda.kde':
plot(x, y, y.group, prior.prob=NULL, cont=c(25,50,75),
     abs.cont, colors, alphavec, xlab, ylab, zlab, drawpoints=FALSE,
     size=3, ptc="blue", ...)

```

Arguments

<code>x</code>	an object of class <code>kda.kde</code> (output from <code>kda.kde</code>)
<code>y</code>	matrix of test data points
<code>y.group</code>	vector of group labels for test data points
<code>prior.prob</code>	vector of prior probabilities
<code>cont</code>	vector of percentages for contour level curves
<code>abs.cont</code>	vector of absolute density estimate heights for contour level curves
<code>xlim,ylim</code>	axes limits
<code>xlab,ylab,zlab</code>	axes labels
<code>drawpoints</code>	if TRUE then draw data points
<code>drawlabels</code>	if TRUE then draw contour labels (2-d plot)
<code>jitter</code>	if TRUE then jitter rug plot (1-d plot)
<code>ptc</code>	vector of colours for data points of each group
<code>partcol</code>	vector of colours for partition classes (1-d, 2-d plot)
<code>col</code>	vector of colours for density estimates (1-d, 2-d plot)
<code>colors</code>	vector of colours for contours of density estimates (3-d plot)
<code>alphavec</code>	vector of transparency values - one for each contour (3-d plot)
<code>size</code>	size of plotting symbol (3-d plot)
<code>...</code>	other graphics parameters

Details

– For 1-d plots:

The partition induced by the discriminant analysis is plotted as rug plot (with the ticks inside the axes). If `drawpoints=TRUE` then the data points are plotted as a rug plot with the ticks outside the axes, their colour is controlled by `ptc`.

– For 2-d plots:

The partition classes are displayed using the colours in `partcol`. The default contours of the density estimate are 25%, 50%, 75% or `cont=c(25,50,75)` for highest density regions. See `plot.kde` for more details.

– For 3-d plots:

Default contours are `cont=c(25, 50, 75)` for highest density regions. See `plot.kde` for more details. The colour of each group is `colors`. The transparency of each contour (within each group) is `alphavec`. Default range is 0.1 to 0.5.

– If `prior.prob` is set to a particular value then this is used. The default is `NULL` which means that the sample proportions are used.

If `y` and `y.group` are missing then the training data points are plotted. Otherwise, the test data `y` are plotted.

Value

Plot of 1-d and 2-d density estimates for discriminant analysis is sent to graphics window. Plot for 3-d is sent to RGL window.

References

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

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

See Also

[kda.kde](#), [kda](#)

Examples

```
library(MASS)
data(iris)

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

## bivariate example
ir <- iris[,1:2]
ir.gr <- iris[,5]
H <- Hkda(ir, ir.gr, bw="plugin", pre="scale")
kda.fhat <- kda.kde(ir, ir.gr, Hs=H)
plot(kda.fhat, cont=0, partcol=4:6)
plot(kda.fhat, drawlabels=FALSE, drawpoints=TRUE)

## trivariate example
## colour indicates species, transparency indicates density heights

ir <- iris[,1:3]
ir.gr <- iris[,5]
H <- Hkda(ir, ir.gr, bw="plugin", pre="scale")
```

```
kda.fhat <- kda.kde(ir, ir.gr, Hs=H)
plot(kda.fhat, cont=50, alpha=0.5)
```

plot.kde

*Kernel density estimate plot for 1- to 3-dimensional data***Description**

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

Usage

```
## univariate
## S3 method for class 'kde':
plot(x, xlab, ylab="Density function", add=FALSE,
     drawpoints=FALSE, ptc="blue", col="black", jitter=TRUE, ...)

## bivariate
## S3 method for class 'kde':
plot(x, display="slice", cont=c(25,50,75), abs.cont,
     xlab, ylab, zlab="Density function", add=FALSE, drawpoints=FALSE,
     drawlabels=TRUE, theta=-30, phi=40, d=4,
     ptc="blue", col="black", ...)

## trivariate
## S3 method for class 'kde':
plot(x, cont=c(25,50,75), abs.cont, colors, add=FALSE,
     drawpoints=FALSE, alpha, alphavec, xlab, ylab, zlab, size=3,
     ptc="blue", ...)
```

Arguments

x	an object of class kde (output from kde function)
display	type of display, "slice" for contour plot, "persp" for perspective plot, "image" for image plot, "filled" for filled contour plot (2-d plot)
cont	vector of percentages for contour level curves
abs.cont	vector of absolute density estimate heights for contour level curves
ptc	plotting colour for data points
col	plotting colour for density estimate (1-d, 2-d plot)
colors	vector of colours for each contour (3-d plot)
jitter	if TRUE then jitter rug plot (1-d plot)
xlab, ylab, zlab	axes labels
add	if TRUE then add to current plot

<code>theta, phi, d</code>	graphics parameters for perspective plots (2-d plot)
<code>drawpoints</code>	if TRUE then draw data points on density estimate
<code>drawlabels</code>	if TRUE then draw contour labels (2-d plot)
<code>alpha</code>	transparency value of plotting symbol (3-d plot)
<code>alphavec</code>	vector of transparency values for contours (3-d plot)
<code>size</code>	size of plotting symbol (3-d plot)
<code>...</code>	other graphics parameters

Details

- The 1-d plot is a standard plot of a 1-d curve. If `drawpoints=TRUE` then a rug plot is added.
- There are three types of plotting displays for 2-d data available, controlled by the `display` parameter.

If `display="slice"` then a slice/contour plot is generated using `contour`. The default contours are at 25%, 50%, 75% or `cont=c(25, 50, 75)` which are upper percentages of highest density regions. See below for alternative contour levels.

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.

If `display="image"` then an image plot is generated. Default colours are the default from the usual `image` command.

- For 3-dimensional data, the interactive plot is a series of nested 3-d contours. The default contours are `cont=c(25, 50, 75)`. See below for alternative contour levels. The default `colors` are `heat.colors` and the default opacity `alphavec` ranges from 0.1 to 0.5.

- To specify contours, either one of `cont` or `abs.cont` is required. `cont` specifies upper percentages which correspond to highest density regions. If `abs.cont=NULL` then a pretty set of contours is drawn. If `abs.cont` is set to particular values, then contours at these levels are drawn. This third option is useful for plotting multiple density estimates with common contour levels. See [contourLevels](#) for details on computing contour levels for `kde` objects.

Value

Plot of 1-d and 2-d kernel density estimates are sent to graphics window. Plot for 3-d is generated by the `misc3d` and `rgl` libraries and is sent to RGL window.

References

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

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

See Also

[kde](#)

Examples

```

## univariate example
x <- rnorm.mixt(n=100, mus=1, sigmas=1, props=1)
fhat <- kde(x=x, h=hpi(x))
plot(fhat)

## bivariate example
data(unicef)
H.scv <- Hscv(x=unicef)
fhat <- kde(x=unicef, H=H.scv)

plot(fhat)
plot(fhat, drawpoints=TRUE, drawlabels=FALSE, col=3, lwd=2)
plot(fhat, display="persp", border=NA, col="grey96", shade=0.75)
plot(fhat, display="image", col=rev(heat.colors(100)))
plot(fhat, display="filled")

## pair of densities with same absolute contour levels
x <- rmvnorm.mixt(n=100, mus=c(0,0), Sigmas=diag(2), props=1)
Hx <- Hpi(x)
fhatx <- kde(x=x, H=Hx, xmin=c(-4,-4), xmax=c(4,4))
y <- rmvnorm.mixt(n=100, mus=c(0.5,0.5), Sigmas=0.5*diag(2), props=1)
Hy <- Hpi(y)
fhaty <- kde(x=y, H=Hy, xmin=c(-4,-4), xmax=c(4,4))
lev <- contourLevels(fhatx, prob=c(0.25, 0.5, 0.75))
plot(fhatx, abs.cont=lev)
plot(fhaty, abs.cont=lev, col=3, add=TRUE)

## large sample - 10000 sample from bivariate standard normal
x <- rmvnorm.mixt(10000, c(0,0), diag(2))
H.pi <- Hpi.diag(x, binned=TRUE)
fhat <- kde(x, H=H.pi, binned=TRUE)
plot(fhat, drawpoints=FALSE, cont=seq(10,90, by=20))

## trivariate example
mus <- rbind(c(0,0,0), c(-1,1,1))
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)

x <- rmvnorm.mixt(n=100, mus=mus, Sigmas=Sigmas, props=props)
H.pi <- Hpi(x, pilot="samse")
fhat <- kde(x, H=H.pi)
plot(fhat)
plot(fhat, axes=FALSE, box=FALSE, drawpoints=TRUE); axes3d(c('x','y','z'))

```

Description

Plot for 2- and 3-dimensional normal and t-mixture density functions.

Usage

```
plotmixt(mus, Sigmas, props, dfs, dist="normal", ...)
```

Arguments

mus	(stacked) matrix of mean vectors
Sigmas	(stacked) matrix of variance matrices
props	vector of mixing proportions
dfs	vector of degrees of freedom
dist	"normal" - normal mixture, "t" - t-mixture
...	other graphics parameters

Details

See the graphics parameter options in `?plot.kde`.

Value

2-d plot is sent to graphics window; 3-d plot to RGL window.

Examples

```
## bivariate example
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, Sigmas, props)
plotmixt(mus, Sigmas, props, dfs=c(3,8), dist="t")

## trivariate example
mus <- rbind(c(0,0,0), c(-1,1,1))
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, Sigmas, props)
plotmixt(mus, Sigmas, props, dfs=c(3,8), dist="t")
```

```
pre.scale, pre.sphere
```

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` if TRUE then centre the data values to have zero mean

Details

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

If \mathbf{H}^* is the bandwidth matrix for the pre-transformed data and \mathbf{H} is the bandwidth matrix for the original data, then $\mathbf{H} = \mathbf{S}^{1/2}\mathbf{H}^*\mathbf{S}^{1/2}$ or $\mathbf{H} = \mathbf{S}_D^{1/2}\mathbf{H}^*\mathbf{S}_D^{1/2}$ as appropriate.

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 <- as.matrix(unicef)
unicef.sp <- pre.sphere(unicef)
unicef.sc <- pre.scale(unicef, mean.centred=TRUE)
var(unicef.sp)
var(unicef.sc)
```

 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.

 vec, vech, invvec, invvech

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)

References

Magnus, J.R. & Neudecker H.M. (1999) *Matrix Differential Calculus with Applications in Statistics and Econometrics (revised edition)*, Wiley & Sons. Chichester.

Examples

```
x <- matrix(1:9, nrow=3, ncol=3)
y <- (x + t(x))/2
vec(x)
vech(y)
invvec(vec(x))
invvech(vech(y))
```

Index

- *Topic **algebra**
 - pre.scale, pre.sphere, 31
 - vec, vech, invvec, invvech, 32
- *Topic **datasets**
 - unicef, 32
- *Topic **distribution**
 - dmvnorm.mixt, rmvnorm.mixt, 7
 - dmvt.mixt, rmvt.mixt, 8
 - dnorm.mixt, rnorm.mixt, 9
- *Topic **hplot**
 - contourLevels, 4
 - plot.kda.kde, 24
 - plot.kde, 27
 - plotmixt, 29
- *Topic **package**
 - ks, 23
- *Topic **smooth**
 - binning, 1
 - compare,
 - compare.kda.diag.cv,
 - compare.kda.cv, 2
 - dkde, pkde, qkde, rkde, 6
 - drvkde, 10
 - Hamise.mixt, Hmise.mixt,
 - Hamise.mixt.diag,
 - Hmise.mixt.diag,
 - amise.mixt, ise.mixt,
 - mise.mixt, 11
 - Hbcv, Hbcv.diag, 13
 - Hlscv, Hlscv.diag, hlscv, 14
 - Hpi, Hpi.diag, hpi, 15
 - Hscv, Hscv.diag, hscv, 17
 - kda, Hkda, Hkda.diag, hkda, 18
 - kda.kde, 20
 - kde, 21
- amise.mixt (*Hamise.mixt*,
Hmise.mixt,
Hamise.mixt.diag,
Hmise.mixt.diag,
amise.mixt, *ise.mixt*,
mise.mixt), 11
- binning, 1
- compare, 19
- compare (*compare*,
compare.kda.diag.cv,
compare.kda.cv), 2
- compare, compare.kda.diag.cv,
compare.kda.cv, 2
- compare.kda.cv, 19
- compare.kda.cv (*compare*,
compare.kda.diag.cv,
compare.kda.cv), 2
- compare.kda.diag.cv (*compare*,
compare.kda.diag.cv,
compare.kda.cv), 2
- contour, 5
- contourLevels, 4, 28
- contourLevels.kda.kde
(*contourLevels*), 4
- contourLevels.kde
(*contourLevels*), 4
- contourLines, 5
- dkde (*dkde*, *pkde*, *qkde*, *rkde*), 6
- dkde, pkde, qkde, rkde, 6
- dmvnorm.mixt, 8
- dmvnorm.mixt (*dmvnorm.mixt*,
rmvnorm.mixt), 7
- dmvnorm.mixt, rmvnorm.mixt, 7
- dmvt.mixt, 7
- dmvt.mixt (*dmvt.mixt*, *rmvt.mixt*),
8
- dmvt.mixt, rmvt.mixt, 8
- dnorm.mixt, 7

- dnorm.mixt (*dnorm.mixt*,
 rnorm.mixt), 9
 dnorm.mixt, *rnorm.mixt*, 9
 drvkde, 10

 Hamise.mixt, 23
 Hamise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 hamise.mixt, 23
 hamise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 Hamise.mixt, *Hmise.mixt*,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt, 11
 Hamise.mixt.diag (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 Hbcv, 15, 18, 23
 Hbcv (*Hbcv*, *Hbcv.diag*), 13
 Hbcv, *Hbcv.diag*, 13
 Hbcv.diag, 23
 Hkda (*kda*, *Hkda*, *Hkda.diag*,
 hkda), 18
 hkda (*kda*, *Hkda*, *Hkda.diag*,
 hkda), 18
 Hkda.diag (*kda*, *Hkda*, *Hkda.diag*,
 hkda), 18
 Hlscv, 3, 14, 18, 19, 23
 Hlscv (*Hlscv*, *Hlscv.diag*, *hlscv*),
 14
 hlscv (*Hlscv*, *Hlscv.diag*, *hlscv*),
 14
 Hlscv, *Hlscv.diag*, *hlscv*, 14
 Hlscv.diag, 23
 Hlscv.diag (*Hlscv*, *Hlscv.diag*,
 hlscv), 14

 Hmise.mixt, 23
 Hmise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 hmise.mixt, 23
 hmise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 Hmise.mixt.diag (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11
 Hpi, 3, 18, 19, 23, 31
 Hpi (*Hpi*, *Hpi.diag*, *hpi*), 15
 hpi, 23
 hpi (*Hpi*, *Hpi.diag*, *hpi*), 15
 Hpi, *Hpi.diag*, *hpi*, 15
 Hpi.diag, 23
 Hpi.diag (*Hpi*, *Hpi.diag*, *hpi*), 15
 Hscv, 3, 14, 15, 19, 23, 31
 Hscv (*Hscv*, *Hscv.diag*, *hscv*), 17
 hscv, 23
 hscv (*Hscv*, *Hscv.diag*, *hscv*), 17
 Hscv, *Hscv.diag*, *hscv*, 17
 Hscv.diag, 23
 Hscv.diag (*Hscv*, *Hscv.diag*,
 hscv), 17

 invvec (*vec*, *vech*, *invvec*,
 invvech), 32
 invvech (*vec*, *vech*, *invvec*,
 invvech), 32
 ise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11

 kda, 26

kda (*kda*, *Hkda*, *Hkda.diag*, *hkda*),
 18
 kda, *Hkda*, *Hkda.diag*, *hkda*, 18
 kda.kde, 4, 19, 20, 23, 25, 26
 kde, 21, 21, 23, 27, 28
 ks, 23

 mise.mixt (*Hamise.mixt*,
 Hmise.mixt,
 Hamise.mixt.diag,
 Hmise.mixt.diag,
 amise.mixt, *ise.mixt*,
 mise.mixt), 11

 pkde (*dkde*, *pkde*, *qkde*, *rkde*), 6
 plot.kda.kde, 21, 23, 24
 plot.kde, 23, 25, 26, 27
 plotmixt, 29
 pre.scale, 16, 17, 19
 pre.scale (*pre.scale*,
 pre.sphere), 31
 pre.scale, *pre.sphere*, 31
 pre.sphere, 16, 17, 19
 pre.sphere (*pre.scale*,
 pre.sphere), 31

 qkde (*dkde*, *pkde*, *qkde*, *rkde*), 6

 rkde (*dkde*, *pkde*, *qkde*, *rkde*), 6
 rmvnorm.mixt, 8
 rmvnorm.mixt (*dmvnorm.mixt*,
 rmvnorm.mixt), 7
 rmvt.mixt, 7
 rmvt.mixt (*dmvt.mixt*, *rmvt.mixt*),
 8
 rnorm.mixt, 7
 rnorm.mixt (*dnorm.mixt*,
 rnorm.mixt), 9

 unicef, 32

 vec (*vec*, *vech*, *invvec*, *invvech*),
 32
 vec, *vech*, *invvec*, *invvech*, 32
 vech (*vec*, *vech*, *invvec*,
 invvech), 32