Package ‘ks’

March 16, 2012

Version 1.8.7

Date 2012/02/22

Title Kernel smoothing

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

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

Depends R (>= 1.4.0), stats, KernSmooth, mvtnorm, rgl (>= 0.66),misc3d (>= 0.4-0)

Suggests MASS

Description Kernel smoothers for univariate and multivariate data

License GPL (>= 2)

URL http://www.mvstat.net/tduong

Repository CRAN

Date/Publication 2012-03-16 10:24:12

R topics documented:

binning 2
contourLevels 3
dkde 4
dmvnorm.mixt 5
dmvt.mixt 6
Hamise.mixt 7
Hbcv 9
Hlscv 10
Hpi 11
Hscv 13
kda 15
kdde 17
kde 18

kde.test 20
ks 22
plot 24
plotmixt 27
pre.scale 29
unicef 29
vec 30

Index **31**

`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)
```

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 (non-negative and sum is equal to sample size). Default is a vector of all ones.

Details

Code is used courtesy of M.P. Wand. Default `bgridsize` are d=1: 401; d=2: rep(151, 2); d=3: rep(31, 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

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))
```

`contourLevels` *Contours for kde and kda.kde objects*

Description

Contour levels and sizes for `kde` and `kda.kde` objects.

Usage

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

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

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`
`abs.cont`	vector of absolute contour levels
`nlevels`	number of pretty contour levels
`approx`	flag to compute approximate contour levels. Default is FALSE.
`...`	other parameters

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.

If `approx=FALSE`, then the exact KDE at x is computed. Otherwise the exact KDE is replaced by the weighted sum of the KDE at the nearest grid points. This can dramatically reduce computation time for large data sets.

The approximate Lebesgue measures are approximated by Riemann sums. Thsese are rough approximations and depend highly on the estimation grid, and so should be interpreted carefully.

Value

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

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

See Also

`contour`, `contourLines`

Examples

```
## contourLevels 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

## contourLevels 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))

## contourSizes kde
x <- rmvnorm.mixt(n=1000, mus=c(0,0), Sigmas=diag(2), props=1)
Hx <- Hpi(x, binned=TRUE)
fhat <- kde(x=x, H=Hx, binned=TRUE)
plot(fhat, cont=25, xlim=c(-1,1), ylim=c(-1,1), asp=1)
contourSizes(fhat, cont=25, approx=TRUE)
 ## compare to approx circle of radius=0.75, vol=pi*0.75^2=1.767146
```

---

dkde *Functions for univariate 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

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 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)
y <- rkde(fhat=fhat, n=100)
```

---

dmvnorm.mixt *Multivariate and univariate normal mixture distribution*

Description

Random generation and density values from 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)

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	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.

Details

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

Value

Normal mixture random vectors and density values.

See Also

rmvt.mixt, dmvt.mixt, dnorm.mixt, rnorm.mixt

Examples

```
## bivariate
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)

##univariate
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))
```

---

dmvt.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

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

Details

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

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)
```

Hamise.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)

Hns(x, 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. See Chacon, Duong & Wand (2011).

`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.

Value

Full 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

```
mus <- rbind(c(0,0), c(2,2))
Sigmas <- rbind(invvech(c(1, 0.7, 1)), invvech(c(1, 0.7, 1)))
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)

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

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

Description

BCV bandwidth matrix for bivariate data.

Usage

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

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=FALSE, bgridsize, amise=FALSE, deriv.order=0,
      verbose=FALSE, optim.fun="nlm", trunc=4)
Hlscv.diag(x, Hstart, binned=FALSE, bgridsize, amise=FALSE, deriv.order=0,
      verbose=FALSE, optim.fun="nlm", trunc=4)
hlscv(x, binned=TRUE, bgridsize, amise=FALSE, deriv.order=0)
```

Arguments

x	vector or matrix of data values
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation. Default is FALSE.
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 truncate numerical optimisation. Default is 4. For details see below.

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.

Truncation of the parameter space is usually required for the LSCV selector 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 = (4/(n*(d+2*r+2)))^(2/(d+2*r+4))*var(x) and LSCV0 = LSCV(H0).

For details about the advanced options for binned,Hstart, 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

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

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, deriv.order=0, verbose=FALSE, optim.fun="nlm",
    Sdr.flag=FALSE)
Hpi.diag(x, nstage=2, pilot="samse", pre="scale", Hstart, binned=FALSE,
    bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="nlm")
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" = 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. Default is FALSE.
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.
Sdr.flag	flag to compute the symmetrizer matrix explicitly for pilot bandwidths for deriv.order>0. Default is FALSE.

Details

hpi is the univariate plug-in selector of Wand & Jones (1994), i.e. it is exactly the same as **KernSmooth**'s dpik. 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 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, deriv.order=r). If Sdr.flag=FALSE, then symmetriser matrices are not computed explicitly but indirectly in a loop of permutations. The loop can make execution times long, but avoid the memory problems for e.g., d=4, deriv.order=2.

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 plugin bandwidth selection. *Computational Statistics*, **9**, 97-116.

See Also

Hbcv, Hlscv, Hscv

Examples

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

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

Description

SCV bandwidth for 1- to 6-dimensional data.

Usage

```
Hscv(x, nstage=2, pre="sphere", pilot="samse", Hstart, binned=FALSE,
     bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="nlm",
     Sdr.flag=FALSE)
Hscv.diag(x, nstage=2, pre="scale", pilot="samse", Hstart, binned=FALSE,
     bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="nlm")
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. Default is FALSE.
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)
Sdr.flag	flag to compute the symmetrizer matrix explicitly for pilot bandwidths for deriv.order>0. Default is FALSE.
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 full bandwidth matrices and Hscv.diag for diagonal bandwidth matrices.

For SAMSE pilot bandwidths, see Duong & Hazelton (2005). Unconstrained 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, Sdr.flag, 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.diag(unicef, binned=TRUE)
hscv(unicef[,1])
```

kda *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)
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
`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`	pilot selector: see `Hpi`, `Hscv`
`pre`	`"scale"` = `pre.scale`, `"sphere"` = `pre.sphere`
`Hstart`	(stacked) matrix of initial bandwidth matrices, used in numerical optimisation
`binned`	flag for binned kernel estimation. Default is FALSE.
`bgridsize`	vector of binning grid sizes
`est.group`	vector of estimated group labels
`by.group`	flag to give results also within each group
`trace`	flag for printing messages in command line to trace the execution
`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

–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 analogous calls to diagonal selectors.

–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 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.

–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 we have test data that is independent of the training data, compare computes MR = (number of points wrongly classified)/(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. 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

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

Examples

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

### univariate example -- independent test data
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))
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")
```

---

kdde *Kernel density derivative estimate for multivariate data*

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=FALSE, bgridsize, positive=FALSE, adj.positive,
    w, deriv.vec=TRUE, verbose=FALSE)
```

Arguments

`x`	matrix of data values
`H`	bandwidth matrix
`h`	scalar bandwidth
`deriv.order`	derivative order (scalar)
`gridsize`	vector of number of grid points
`gridtype`	not yet implemented
`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 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 data `x <- log(x + adj.positive)` when `positive=TRUE`. Default is the minimum of `x`.
`w`	vector of weights (non-negative and sum is equal to sample size). 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.

Details

Binning/exact estimation behaviour is the same as for kde.

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 rows of `deriv.ind`.

Value

The result is kernel density derivative estimate is an object of class `kdde`:

`x`	data points - same as input
`eval.points`	points at which the density estimate is evaluated
`estimate`	density derivative estimate at `eval.points`
`H`	bandwidth matrix
`h`	scalar bandwidth (1-d only)
`w`	weights
`deriv.order`	derivative order (scalar)
`deriv.ind`	each row is a vector of partial derivative indices

See Also

kde

Examples

```
## univariate example
x <- rnorm.mixt(n=100, mus=1, sigmas=1, props=1)
fhat2 <- kdde(x=x, h=hpi(x), deriv.order=2)      ## d^2 f/dx^2

## bivariate example
data(unicef)
H.scv <- Hscv(x=unicef)
fhat1 <- kdde(x=unicef, H=H.scv, deriv.order=1)   ## gradient vector
```

`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,
    compute.cont=FALSE, approx.cont=TRUE)

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

Arguments

`x`	matrix of data values
`x.group`	vector of group labels
`H,Hs`	bandwidth matrix(ces)
`h,hs`	scalar bandwidth(s)
`prior.prob`	vector of prior probabilities
`gridsize`	vector of number of grid points
`gridtype`	not yet implemented
`xmin`	vector of minimum values for grid
`xmax`	vector of maximum values for grid
`supp`	effective support for standard normal.
`eval.points`	points at which density 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 data `x <- log(x + adj.positive)` when `positive=TRUE`. Default is the minimum of `x`.
`w`	vector of weights (non-negative and sum is equal to sample size). Default is a vector of all ones.
`compute.cont`	flag for computing probability contour levels from 1% to 99%. Default is FALSE.
`approx.cont`	flag for computing approximate probability contour levels. Default is TRUE.

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`). 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.

`supp` is the effective support for a normal kernel, i.e. all values outside `[-supp,supp]^d` are set to 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 default weights `w` is a vector of all ones.

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

Value

–The result from kde is a kernel density estimate which is an object of class kde:

x	data points - same as input
eval.points	points at which the density estimate is evaluated
estimate	density estimate at eval.points
H	bandwidth matrix
h	scalar bandwidth (1-d only)
w	weights
cont	probability contour levels

–The result from kda.kde is a density estimate for discriminant analysis which is an object of class kda.kde:

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

See Also

plot.kde, plot.kda.kde

Examples

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

---

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

Description

Kernel density based two-sample comparison test for 2- to 6-dimensional data.

Usage

```
kde.test(x1, x2, H1, H2, psi1, psi2, fhat1, fhat2, var.fhat1, var.fhat2,
    double.loop=FALSE, binned=FALSE, bgridsize, verbose=FALSE, pre.scale=FALSE)
Hpi.kfe(x, nstage=2, Hstart, deriv.order=0, binned=FALSE, bgridsize,
    double.loop=FALSE, amise=FALSE, verbose=FALSE)
```

Arguments

`x,x1,x2`	matrices of data values
`H1,H2`	bandwidth matrices. If these are missing, `Hpi.kfe` is called by default.
`fhat1,fhat2`	gridded kde objects - output from `kde` (for 2- or 3-d data)
`psi1,psi2`	zero-th order kernel functional estimates
`var.fhat1,var.fhat2`	sample variance of KDE estimates evaluated at x1, x2
`double.loop`	flag to iterate through double loop for unbinned estimation. Default is FALSE.
`binned`	flag for binned estimation. Default is FALSE.
`bgridsize`	vector of binning grid sizes
`verbose`	flag to print out progress information. Default is FALSE.
`pre.scale`	flag to pre-scale combined data set. Default is FALSE.
`nstage`	number of stages in the plug-in bandwidth selector (1 or 2)
`Hstart`	initial bandwidth matrix, used in numerical optimisation
`amise`	flag to return the minimal scaled PI value
`deriv.order`	derivative order of kfe (kernel functional estimate). Only deriv.order=0 is currently implemented.

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 L2 error (ISE) $T = \int [f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})]^2 \, d\boldsymbol{x}$. If we rewrite this as $T = \psi_1 - \psi_{12} - \psi_{21} + \psi_2$ where $\psi_{uv} = \int f_u(\boldsymbol{x}) f_v(\boldsymbol{x}) \, d\boldsymbol{x}$, then we can use kernel functional estimators. Duong (2011) shows that this test statistic has a null distribution which is asymptotically normal, so no bootstrap resampling is required to compute an approximate p-value.

–`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). This is automatically called by `kde.test` to estimate the ψ functionals with $r = 0$.

Value

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

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

Duong, T. (2011) Multivariate kernel density based two-sample comparisons without bootstrap resampling. Submitted.

Examples

```
mus1 <- rbind(c(1,-1), c(-1,1))
Sigmas1 <- rbind(invvech(c(4/9, 4/15, 4/9)), invvech(c(4/9, 4/15, 4/9)))
props1 <- c(1,1)/2
mus2 <- rbind(c(1,-1), c(-1,1))
Sigmas2 <- rbind(invvech(c(4/9, 14/45, 4/9)), 4/9*diag(2))
props2 <- c(1,1)/2

set.seed(8192)
samp <- 1000
x <- rmvnorm.mixt(n=samp, mus=mus1, Sigmas=Sigmas1, props=props1)
y <- rmvnorm.mixt(n=samp, mus=mus2, Sigmas=Sigmas2, props=props2)
y2 <- rmvnorm.mixt(n=samp, mus=mus2, Sigmas=Sigmas2, props=props2)
kde.test(x1=x, x2=y)$pvalue    ## reject H0: f1=f2
kde.test(x1=y2, x2=y)$pvalue   ## accept H0: f1=f2
```

---

ks *ks*

Description

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

Details

There are three main types of functions in this package: (a) computing kernel estimators, (b) computing bandwidth selectors and (c) displaying kernel estimators. The kernel used throughout is the normal (Gaussian) kernel.

–For kernel density estimation, `kde` computes

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

For display, its `plot` method calls `plot.kde`.

–For kernel discriminant analysis, `kda.kde` computes density estimates for each the groups in the training data, and the discriminant surface. Its `plot` method is `plot.kda.kde`. The wrapper function `Hkda` computes bandwidth matrices for each group in the training data, by calling the above selectors.

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

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

–There are several varieties of bandwidth matrix 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 hmise.mixt, hamise.mixt (1-d); and Hmise.mixt, Hamise.mixt, Hmise.mixt.diag, Hamise.mixt.diag (2- to 6-d).

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 functional estimation, kfe computes the r-th order integrated density functional

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

The plug-in selector is Hpi.kfe.

–Binned kernel estimation 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").

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. Jose E. Chacon for the unconstrained pilot functional estimation and fast implementation of density derivative estimation code.

References

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

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

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. (1995) *Kernel Smoothing*. Chapman & Hall/CRC, London.

See Also

sm, KernSmooth

---

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

Description

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

Usage

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

Arguments

x	an object of class kde (output from kde) or an object of class kda.kde (output from kda.kde)
y	matrix of test data points
y.group	vector of group labels for test data points
...	other graphics parameters

Details

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

```
## univariate
plot(x, xlab, ylab="Density function", add=FALSE, drawpoints=FALSE,
ptcol="blue", col="black", cont.col=1, cont.lwd, jitter=FALSE, cont,
abs.cont, approx.cont=FALSE, ...)

## bivariate
plot(x, display="slice", cont=c(25,50,75), abs.cont,
  approx.cont=FALSE, xlab, ylab, zlab="Density function", add=FALSE,
  drawpoints=FALSE, drawlabels=TRUE, theta=-30, phi=40, d=4,
  ptcol="blue", col="black", ...)

## trivariate
plot(x, cont=c(25,50,75), abs.cont, approx.cont=FALSE, colors,
```

```
  add=FALSE, drawpoints=FALSE, alpha, alphavec, xlab, ylab, zlab,
  size=3, ptcol="blue", ...)
```

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

```
## univariate
plot(x, y, y.group, prior.prob=NULL, xlim, ylim,
  xlab="x", ylab="Weighted density function", drawpoints=FALSE,
  col, ptcol, jitter=TRUE, ...)

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

## trivariate
plot(x, y, y.group, prior.prob=NULL, cont=c(25,50,75),
  abs.cont,  approx.cont=FALSE, colors, alphavec, xlab, ylab, zlab,
  drawpoints=FALSE, size=3, ptcol="blue", ...)
```

The arguments are

`display` type of display, "slice" for contour plot, "persp" for perspective plot, "image" for image plot, "filled.contour" for filled contour plot (1st form), "filled.contour2" (2nd form) (2-d plot)

`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.

`ptcol` 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` flag to jitter rug plot (1-d plot). Default is TRUE.

`xlim, ylim` axes limits

`xlab,ylab,zlab` axes labels

`add` flag to add to current plot. Default is FALSE.

`theta,phi,d` graphics parameters for perspective plots (2-d plot)

`drawpoints` flag to draw data points on density estimate. Default is FALSE.

`drawlabels` flag to draw contour labels (2-d plot). Default is TRUE.

`alpha` transparency value of plotting symbol (3-d plot)

`alphavec` vector of transparency values for contours (3-d plot)

`size` size of plotting symbol (3-d plot)

`prior.prob` vector of prior probabilities

`partcol` vector of colours for partition classes (1-d, 2-d plot)

The 1-d 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.

There are different types of plotting displays for 2-d data available, controlled by the `display` parameter. (a) If `display="slice"` then a slice/contour plot is generated using `contour`. (b) If `display` is `"filled.contour"` or `"filled.contour2"` 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. See below for alternative contour levels. (c) `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. 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 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 at x is computed. Otherwise the exact KDE is replaced by the KDE at the nearest grid point. This can dramatically reduce computation time for large data sets.

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.

See Also

`kde`, `kda.kde`

Examples

```
## kde examples

## univariate example
x <- rnorm.mixt(n=100, mus=1, sigmas=1, props=1)
fhat <- kde(x=x, h=hpi(x))
plot(fhat)
plot(fhat, cont=50, cont.col="blue", cont.lwd=2)

## bivariate example
data(unicef)
H.scv <- Hscv(x=unicef)
fhat <- kde(x=unicef, H=H.scv, compute.cont=TRUE)
plot(fhat, drawpoints=TRUE, drawlabels=FALSE, col=3, lwd=3, cex=0.1)
plot(fhat, display="persp", border=NA, shade=0.3)
plot(fhat, display="image", col=rev(heat.colors(100)))
plot(fhat, display="filled.contour2", cont=seq(10,90,by=10))
```

```
## 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 from bivariate normal
x <- rmvnorm.mixt(10000, c(0,0), invvech(c(1, 0.8, 1)))
H <- Hpi(x, binned=TRUE)
fhat <- kde(x, H=H, binned=TRUE)
plot(fhat, display="persp", border=NA, shade=0.1)

## trivariate example
library(MASS)
x <- iris[,1:3]
H.pi <- Hpi(x, pilot="samse")
fhat <- kde(x, H=H.pi, compute.cont=TRUE)
plot(fhat, drawpoints=TRUE)

## kda.kde examples

## 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, abs.cont=0)

## trivariate example
## colour indicates species, transparency indicates density heights
ir <- iris[,1:3]
ir.gr <- iris[,5]
H <- Hkda(ir, ir.gr, bw="plugin")
kda.fhat <- kda.kde(ir, ir.gr, Hs=H, compute.cont=TRUE)
plot(kda.fhat)
```

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

Description

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

Usage

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

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
`draw`	flag to draw plot. Default is TRUE.
`...`	other graphics parameters

Details

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

Value

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

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)

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

pre.scale — *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 <- as.matrix(unicef)
unicef.sp <- pre.sphere(unicef)
var(unicef.sp)
```

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 — *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. (1999) *Matrix Differential Calculus with Applications in Statistics and Econometrics (revised edition)*, Wiley & Sons. Chichester.

Examples

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

Index

∗Topic **algebra**
pre.scale, 29
vec, 30
∗Topic **datasets**
unicef, 29
∗Topic **distribution**
dmvnorm.mixt, 5
dmvt.mixt, 6
∗Topic **hplot**
contourLevels, 3
plot, 24
plotmixt, 27
∗Topic **package**
ks, 22
∗Topic **smooth**
binning, 2
dkde, 4
Hamise.mixt, 7
Hbcv, 9
Hlscv, 10
Hpi, 11
Hscv, 13
kda, 15
kde, 17
kde, 18
∗Topic **test**
kde.test, 20

amise.mixt *(*Hamise.mixt*)*, 7

binning, 2

compare *(*kda*)*, 15
contour, *4*
contourLevels, 3, *26*
contourLines, *4*
contourSizes *(*contourLevels*)*, 3

dkde, 4
dmvnorm.mixt, 5, *7*
dmvt.mixt, 6, *6*
dnorm.mixt, *6*
dnorm.mixt *(*dmvnorm.mixt*)*, 5

Hamise.mixt, 7, *23*
hamise.mixt, *23*
hamise.mixt *(*Hamise.mixt*)*, 7
Hamise.mixt.diag, *23*
Hamise.mixt.diag *(*Hamise.mixt*)*, 7
Hbcv, 9, *11*, *13, 14*, *23*
Hbcv.diag, *23*
Hkda, *22*
Hkda *(*kda*)*, 15
hkda *(*kda*)*, 15
Hkda.diag *(*kda*)*, 15
Hlscv, 10, *10*, *13–16*, *23*
hlscv, *23*
hlscv *(*Hlscv*)*, 10
Hlscv.diag, *23*
Hlscv.diag *(*Hlscv*)*, 10
Hmise.mixt, *23*
Hmise.mixt *(*Hamise.mixt*)*, 7
hmise.mixt, *23*
hmise.mixt *(*Hamise.mixt*)*, 7
Hmise.mixt.diag, *23*
Hmise.mixt.diag *(*Hamise.mixt*)*, 7
Hns *(*Hamise.mixt*)*, 7
Hpi, *9, 10*, 11, *11*, *14–16*, *23*, *29*
hpi, *23*
hpi *(*Hpi*)*, 11
Hpi.diag, *23*
Hpi.diag *(*Hpi*)*, 11
Hpi.kfe, *23*
Hpi.kfe *(*kde.test*)*, 20
Hscv, *10, 11*, 13, *13*, *15, 16*, *23*, *29*
hscv, *23*
hscv *(*Hscv*)*, 13
Hscv.diag, *23*
Hscv.diag *(*Hscv*)*, 13

invvec *(*vec*)*, 30
invvech *(*vec*)*, 30
ise.mixt *(*Hamise.mixt*)*, 7

kda, 15
kda.kde, *22*, *24*, *26*
kda.kde *(*kde*)*, 18
kdde, 17, *23*
kde, 18, *18*, *21, 22*, *24*, *26*
kde.test, 20
ks, 22

mise.mixt *(*Hamise.mixt*)*, 7

nlm, *10*, *12*, *14*

optim, *10*, *12*, *14*

pkde *(*dkde*)*, 4
plot, 24
plot.kda.kde, *20*, *22*
plot.kde, *20*, *22*
plotmixt, 27
pre.scale, *12, 13*, *15*, 29
pre.sphere, *12, 13*, *15*
pre.sphere *(*pre.scale*)*, 29

qkde *(*dkde*)*, 4

rkde *(*dkde*)*, 4
rmvnorm.mixt, *7*
rmvnorm.mixt *(*dmvnorm.mixt*)*, 5
rmvt.mixt, *6*
rmvt.mixt *(*dmvt.mixt*)*, 6
rnorm.mixt, *6*
rnorm.mixt *(*dmvnorm.mixt*)*, 5

unicef, 29

vec, 30
vech *(*vec*)*, 30