Package 'ks'

June 25, 2016

Version 1.10.4

Date 2016-06-25

Title Kernel Smoothing

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

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

Depends R (>= 1.4.0), KernSmooth (>= 2.22), misc3d (>= 0.4-0), mvtnorm (>= 1.0-0), rgl (>= 0.66)

Imports grDevices, graphics, multicool, stats, utils

Suggests MASS

Description Kernel smoothers for univariate and multivariate data.

License GPL-2 | GPL-3

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-06-25 19:07:41

R topics documented:

---

ks-package *ks*

Description

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

Details

There are three main types of functions in this package:

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

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

–For kernel density estimation, `kde` computes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\hat{c}(\boldsymbol{z}) = n^{-1} \sum_{i=1}^{n} K_{\mathbf{H}}(\boldsymbol{z} - \hat{\boldsymbol{Z}}_i)$$

where $\hat{\boldsymbol{Z}}_i = (\hat{F}_1(X_{i1}), \ldots, \hat{F}_d(X_{id}))$. The bandwidth selectors are those used with `kde` for $\hat{c}$ and `kcde` for $\hat{F}_j$. Its `plot` method is `plot.kde`.

–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 derivative-based estimation code. Artur and Jaroslaw Gramacki for the binned estimation for unconstrained bandwidth matrices.

References

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

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`

`binning` *Linear binning for multivariate data*

Description

Linear binning for 1- to 4-dimensional data.

Usage

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

Arguments

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

Details

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

Value

Returns a list with 2 fields

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

References

Gramacki, A. & Gramacki, J. (2015) *FFT-based fast computation of multivariate kernel estimators with unconstrained bandwidth matrices*. URL: arxiv.org/abs/1508.02766.

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

Examples

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

contour *Contours functions*

Description

Contour levels and sizes.

Usage

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

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

Arguments

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

Details

–For contourLevels, 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 i.e. cont=(1-prob)*100%. If both prob and cont are missing then a pretty set of nlevels contours are computed.

–For contourSizes, the 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.

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

Value

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

–For 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

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

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

Description

BCV bandwidth matrix for bivariate data.

Usage

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

Arguments

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

Details

Use Hbcv for 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 binned, Hstart, see Hpi.

Value

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

References

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

See Also

Hlscv, Hpi, Hscv

Examples

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

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

Arguments

x	vector or matrix of data values
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation. 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 control truncation for numerical optimisation. Default is 4 for density.deriv>0, otherwise no truncation. For details see below.
...	parameters as above

Details

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

Truncation of the parameter space is usually required for the LSCV selector, for r > 0, to find a reasonable solution to the numerical optimisation. If a candidate matrix `H` is such that `det(H)` is not in `[1/trunc, trunc]*det(H0)` or `abs(LSCV(H)) > trunc*abs(LSCV0)` then the `LSCV(H)` is reset to `LSCV0` where `H0=Hns(x)` and `LSCV0=LSCV(H0)`.

For details about the advanced options for `binned`,`Hstart`, 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(forbes$bp)
```

`Hns` *Normal scale bandwidth*

Description

Normal scale bandwidth.

Usage

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

Arguments

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

Details

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

Value

Full normal scale bandwidth matrix.

References

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

Examples

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

Hpi *Plug-in bandwidth selector*

Description

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

Usage

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

Arguments

x	vector or matrix of data values
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"amse" = AMSE pilot bandwidths "samse" = single SAMSE pilot bandwidth "unconstr" = single unconstrained pilot bandwidth "dscalar" = single pilot bandwidth for deriv.order >= 0 "dunconstr" = single unconstrained pilot bandwidth for deriv.order >= 0
pre	"scale" = pre.scale, "sphere" = pre.sphere
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned kernel estimation. 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

Details

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

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

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

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, pilot="dscalar")
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, Hstart, binned=FALSE,
     bgridsize, amise=FALSE, deriv.order=0, verbose=FALSE, optim.fun="nlm")
Hscv.diag(x, nstage=2, pre="scale", pilot, 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)
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.

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

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

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

Value

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

References

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

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

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

See Also

Hbcv, Hlscv, Hpi

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

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

kcde *Kernel cumulative distribution/survival function estimate*

Description

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

Usage

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

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

Arguments

x	matrix of data values
H,h	bandwidth matrix/scalar bandwidth. If these are missing, then Hpi.kcde or hpi.kcde is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin,xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	points at which estimate is evaluated
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
positive	flag if 1-d data are positive. Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	not yet implemented
verbose	flag to print out progress information. Default is FALSE.
tail.flag	"lower.tail" = cumulative distribution, "upper.tail" = survival function
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"dscalar" = single pilot bandwidth (default for Hpi.diag.kcde "dunconstr" = single unconstrained pilot bandwidth (default for Hpi.kcde
Hstart	initial bandwidth matrix, used in numerical optimisation

amise	flag to return the minimal scaled PI value
optim.fun	optimiser function: one of nlm or optim
object	object of class kcde
...	other parameters

Details

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

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

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

Value

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

x	data points - same as input
eval.points	points at which the estimate is evaluated
estimate	cumulative distribution/survival function estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	weights
tail	"lower.tail"=cumulative distribution, "upper.tail"=survival function

References

Duong, T. (2015) Non-parametric smoothed estimation of multivariate cumulative distribution and survival functions, and receiver operating characteristic curves. *Journal of the Korean Statistical Society*. In press. DOI:10.1016/j.jkss.2015.06.002.

See Also

kde, plot.kcde

Examples

```
library(MASS)
data(iris)
Fhat <- kcde(iris[,1:2])
predict(Fhat, x=iris[,1:2])

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

---

kcopula *Kernel copula/copula density estimate*

Description

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

Usage

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

Arguments

x	matrix of data values
H,hs	bandwidth matrix. If these are missing, Hpi.kcde or hpi.kcde or hpi is called by default.
Hfun	bandwidth matrix function. If missing, Hpi is the default. This is called only when H is missing.
Hfun.pilot	pilot bandwidth matrix - see Hpi
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin,xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	points at which estimate is evaluated
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
w	vector of weights. Default is a vector of all ones.
verbose	flag to print out progress information. Default is FALSE.
marginal	"kernel" = kernel cdf or "empirical" = empirical cdf to calculate pseudo-uniform values. Default is "kernel".
compute.cont	flag for computing 1% to 99% probability contour levels. Default is FALSE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
boundary.supp	scaled boundary region is [0, boundary.supp*h] or [1-boundary.supp*h,1] on [0,1]. Default is 1.

Details

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

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

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

Value

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

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

References

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

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

See Also

kcde, kde

Examples

```
library(MASS)
data(fgl)
x <- fgl[,c("RI", "Na")]
Chat <- kcopula(x=x)
plot(Chat, disp="persp", thin=3, col="white", border=1)
```

kda *Kernel discriminant analysis*

Description

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

Usage

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

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

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

Arguments

x	matrix of training data values
x.group	vector of group labels for training data
Hs,hs	(stacked) matrix of bandwidth matrices/vector of scalar bandwidths. If these are missing, Hkda or hkda is called by default.
prior.prob	vector of prior probabilities
gridsize	vector of grid sizes
xmin,xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal
eval.points	points at which estimate is evaluated
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
w	vector of weights. Not yet implemented.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is FALSE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
kde.flag	flag for computing KDE on grid. Default is TRUE.
object	object of class kda

bw	bandwidth: "plugin" = plug-in, "lscv" = LSCV, "scv" = SCV
Hstart	(stacked) matrix of initial bandwidth matrices, used in numerical optimisation
est.group	vector of estimated group labels
by.group	flag to give results also within each group
verbose	flag for printing progress information. Default is FALSE.
recompute	flag for recomputing the bandwidth matrix after excluding the i-th data item
...	other optional parameters for bandwidth selection, see Hpi, Hlscv, Hscv

Details

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

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

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

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

Value

–A kernel discriminant analysis is an object of class kda which is a list with fields

x	list of data points, one for each group label
estimate	list of density estimates at eval.points, one for each group label
eval.points	points that the estimate is evaluated at, one for each group label
h	vector of bandwidths (1-d only)
H	stacked matrix of bandwidth matrices or vector of bandwidths
gridded	flag for estimation on a grid
binned	flag for binned estimation
w	weights
prior.prob	prior probabilities
x.group	group labels - same as input
x.group.estimate	estimated group labels. If the test data eval.points are given then these are classified. Otherwise the training data x are classified.

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

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

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

`error` misclassification rate (MR)

In the case where the test data is independent of the training data, `compare` computes MR = (number of points wrongly classified)/(total number of points). In the case where the test data are not independent e.g. we are classifying the training data set itself, then the cross validated estimate of MR is more appropriate. These are implemented as `compare.kda.cv` (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

See Also

`plot.kda`

Examples

```
set.seed(8192)
x <- c(rnorm.mixt(n=100, mus=1), rnorm.mixt(n=100, mus=-1))
x.gr <- rep(c(1,2), times=c(100,100))
y <- c(rnorm.mixt(n=100, mus=1), rnorm.mixt(n=100, mus=-1))
kda.gr <- kda(x, x.gr, eval.points=y)
compare(kda.gr$x.group, kda.gr$x.group.est, by.group=TRUE)
predict(kda.gr, x=0)

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

---

kdde *Kernel density derivative estimate*

Description

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

Usage

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

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

Arguments

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

Details

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

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

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

Value

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

x	data points - same as input
eval.points	points at which the estimate is evaluated
estimate	density derivative estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid

binned	flag for binned estimation
names	variable names
w	weights
deriv.order	derivative order (scalar)
deriv.ind	each row is a vector of partial derivative indices

See Also

kde

Examples

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

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

---

kde *Kernel density estimate*

Description

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

Usage

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

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

Arguments

x	matrix of data values
H,h	bandwidth matrix/scalar bandwidth. If these are missing, Hpi or hpi is called by default.
gridsize	vector of number of grid points
gridtype	not yet implemented
xmin,xmax	vector of minimum/maximum values for grid
supp	effective support for standard normal

eval.points	points at which estimate is evaluated
binned	flag for binned estimation. Default is FALSE.
bgridsize	vector of binning grid sizes
positive	flag if 1-d data are positive. Default is FALSE.
adj.positive	adjustment applied to positive 1-d data
w	vector of weights. Default is a vector of all ones.
compute.cont	flag for computing 1% to 99% probability contour levels. Default is FALSE.
approx.cont	flag for computing approximate probability contour levels. Default is TRUE.
unit.interval	flag if 1-d data are bounded on unit interval [0,1]. Default is FALSE.
verbose	flag to print out progress information. Default is FALSE.
object	object of class kde
zero.flag	interploted values object$estimate of when x outside of object$eval.points=0 (if TRUE), = nearest object$estimate (if FALSE)
...	other parameters

Details

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

For d=1, if positive=TRUE then x<-log(x+adj.positive) where the default adj.positive is the minimum of x.

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.

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

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

The default xmin is min(x)-Hmax*supp and xmax is max(x)+Hmax*supp where Hmax is the maximum of the diagonal elements of H. The grid produced is the outer product of c(xmin[1], xmax[1]), ..., c(xmin[d], xmax[d]).

Value

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

x	data points - same as input
eval.points	points at which the estimate is evaluated
estimate	density estimate at eval.points
h	scalar bandwidth (1-d only)
H	bandwidth matrix
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation

names	variable names
w	weights
cont	probability contour levels (if compute.cont=TRUE)

See Also

plot.kde

Examples

```
## positive data example
set.seed(8192)
x <- 2^rnorm(100)
fhat <- kde(x=x, positive=TRUE)
plot(fhat, col=3)
points(c(0.5, 1), predict(fhat, x=c(0.5, 1)))

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

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

---

kde.1d *Functions for univariate kernel density estimates*

Description

Functions for 1-dimensional kernel density estimates.

Usage

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

Arguments

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

Details

pkde uses Simpson's rule 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, binned=TRUE)
p1 <- pkde(fhat=fhat, q=c(-1, 0, 0.5))
qkde(fhat=fhat, p=p1)
y <- rkde(fhat=fhat, n=100)
```

---

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

Description

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

Usage

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

Arguments

x1,x2	vector/matrix of data values
H1,H2,h1,h2	bandwidth matrices/scalar bandwidths. If these are missing, Hpi or hpi is called by default.
fhat1,fhat2	objects of class kde
binned	flag for binned estimation. Default is FALSE.
gridsize	vector of grid sizes
bgridsize	vector of binning grid sizes
verbose	flag to print out progress information. Default is FALSE.

supp	effective support for normal kernel
mean.adj	flag to compute second order correction for mean value of critical sampling distribution. Default is FALSE. Currently implemented for d<=2 only.
signif.level	significance level. Default is 0.05.
min.ESS	minimum effective sample size. See below for details.

Details

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

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

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

Value

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

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

References

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

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

See Also

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

Examples

```
library(MASS)
x1 <- crabs[crabs$sp=="B", 4]
x2 <- crabs[crabs$sp=="O", 4]
loct <- kde.local.test(x1=x1, x2=x2)
plot(loct)

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

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

Description

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

Usage

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

Arguments

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

Details

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

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

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

Value

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

`Tstat`	T statistic
`zstat`	z statistic - normalised version of Tstat
`pvalue`	p-value of the double sided test
`mean,var`	mean and variance of null distribution
`var.fhat1,var.fhat2`	sample variances of KDE values evaluated at data points
`n1,n2`	sample sizes
`H1,H2`	bandwidth matrices
`psi1,psi12,psi21,psi2`	kernel functional estimates

References

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

See Also

`kde.local.test`

Examples

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

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

`kfe` *Kernel functional estimate*

Description

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

Usage

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

Arguments

x	vector/matrix of data values
nstage	number of stages in the plug-in bandwidth selector (1 or 2)
pilot	"dscalar" = single pilot bandwidth (default) "dunconstr" = single unconstrained pilot bandwidth
pre	"scale" = pre.scale, "sphere" = pre.sphere
Hstart	initial bandwidth matrix, used in numerical optimisation
binned	flag for binned estimation. 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
G	pilot bandwidth matrix
inc	0=exclude diagonal, 1=include diagonal terms in kfe calculation
bin.par	binning parameters - output from binning
deriv.vec	flag to compute duplicated partial derivatives in the vectorised form. Default is FALSE.
add.index	flag to ouput derivative indices matrix. Default is true.

Details

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

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

Value

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

References

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

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

See Also

kde.test

---

kroc *Kernel receiver operating characteristic (ROC) curve*

Description

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

Usage

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

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

Arguments

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

Details

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

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

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

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

Value

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

x	list of data values x1, x2 - same as input
eval.points	points at which the estimate is evaluated
estimate	ROC curve estimate at eval.points
gridtype	"linear"
gridded	flag for estimation on a grid
binned	flag for binned estimation
names	variable names
w	weights
tail	"lower.tail"
h1	scalar bandwidth for first sample (1-d only)
H1	bandwidth matrix for first sample
hy	scalar bandwidth for ROC curve
indices	summary indices of ROC curve.

References

Duong, T. (2015) Non-parametric smoothed estimation of multivariate cumulative distribution and survival functions, and receiver operating characteristic curves. *Journal of the Korean Statistical Society*. In press. DOI:10.1016/j.jkss.2015.06.002.

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

See Also

kcde

Examples

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

---

mixt *Normal and t-mixture distributions*

Description

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

Usage

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

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.
dfs	vector of degrees of freedom

Details

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

Value

Normal and t-mixture random vectors and density values.

Examples

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

## bivariate mixtures
mus <- rbind(c(-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 <- rmvnorm.mixt(1000, mus=mus, Sigmas=Sigmas, props=props)
y <- rmvt.mixt(1000, mus=mus, Sigmas=Sigmas, dfs=dfs, props=props)
```

---

plot.kcde *Plot for kernel cumulative distribution estimate*

Description

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

Usage

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

Arguments

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

Details

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

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

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

Value

Plots for 1-d and 2-d are sent to graphics window. Plot for 3-d is sent to RGL window (not yet implemented).

See Also

plot.kde

Examples

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

---

plot.kda *Plot for kernel discriminant analysis*

Description

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

Usage

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

Arguments

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

Details

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

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

## 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, col.part, col.pt, ...)

## 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, col.pt="blue", ...)
```

Value

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

See Also

kda, kde

Examples

```
library(MASS)
data(iris)

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

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

## trivariate example
ir <- iris[,1:3]
ir.gr <- iris[,5]
H <- Hkda(x=ir, x.group=ir.gr, bw="plugin", pilot="dscalar")
kda.fhat <- kda(x=ir, x.group=ir.gr, Hs=H)
plot(kda.fhat, drawpoints=TRUE, col.pt=c(2,3,4))
   ## colour=species, transparency=density heights
```

---

plot.kdde *Plot for kernel density derivative estimate*

Description

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

Usage

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

Arguments

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

Details

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

```
## univariate
plot(fhat, ylab="Density derivative function", ...)

## bivariate
plot(fhat, which.deriv.ind=1, cont=c(25,50,75), abs.cont, display="slice",
   zlab="Density derivative function", ...)
```

Value

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

See Also

plot.kde

Examples

```
## univariate example
data(unicef)
fhat1 <- kdde(x=unicef[,1], deriv.order=1)  ## gradient [df/dx, df/dy]
plot(fhat1, xlab="Under-5")                 ## df/dx
points(30,predict(fhat1, x=30))

## bivariate example
fhat2 <- kdde(x=unicef, deriv.order=2)
plot(fhat2, which.deriv.ind=2, display="persp", phi=20)
plot(fhat2, which.deriv.ind=2, display="filled.contour2", col.fun=topo.colors)
  ## d^2 f/(dx dy): purple=-ve, green=zero, beige=+ve
```

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

Description

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

Usage

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

Arguments

`x`	an object of class `kde` (output from `kde`)
`...`	other graphics parameters:
	`display` type of display, "slice" for contour plot, "persp" for perspective plot, "image" for image plot, "filled.contour" for filled contour plot (1st form), "filled.contour2" (2nd form) (2-d)
	`cont` vector of percentages for contour level curves
	`abs.cont` vector of absolute density estimate heights for contour level curves
	`approx.cont` flag to compute approximate contour levels. Default is FALSE.
	`col` plotting colour for density estimate (1-d, 2-d)
	`col.cont` plotting colour for contours
	`col.fun` plotting colour function for contours
	`col.pt` plotting colour for data points
	`colors` vector of colours for each contour (3-d)
	`jitter` flag to jitter rug plot (1-d). Default is TRUE.
	`lwd.fc` line width for filled contours (2-d)
	`xlim,ylim,zlim` axes limits
	`xlab,ylab,zlab` axes labels
	`add` flag to add to current plot. Default is FALSE.
	`theta,phi,d,border` graphics parameters for perspective plots (2-d)
	`drawpoints` flag to draw data points on density estimate. Default is FALSE.
	`drawlabels` flag to draw contour labels (2-d). Default is TRUE.
	`alpha` transparency value of plotting symbol (3-d)
	`alphavec` vector of transparency values for contours (3-d)
	`size` size of plotting symbol (3-d).

Details

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

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

## bivariate
plot(fhat, display="slice", cont=c(25,50,75), abs.cont, approx.cont=TRUE,
xlab, ylab, zlab="Density function", cex=1, pch=1, add=FALSE,
drawpoints=FALSE, drawlabels=TRUE, theta=-30, phi=40, d=4, col.pt="blue",
col, col.fun, lwd=1, border=1, thin=3, lwd.fc=5, ...)

## trivariate
plot(fhat, cont=c(25,50,75), abs.cont, approx.cont=FALSE, colors,
  add=FALSE, drawpoints=FALSE, alpha, alphavec, xlab, ylab, zlab,
  size=3, col.pt="blue", ...)
```

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. (c) If `display="persp"` then a perspective/wire-frame plot is generated. The default z-axis limits `zlim` are the default from the usual `persp` command. (d) If `display="image"` then an image plot is generated. 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)`. 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 is computed. Otherwise it is interpolated from an existing KDE grid. This can dramatically reduce computation time for large data sets.

Value

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

Examples

```
library(MASS)
data(iris)
```

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

## bivariate example
fhat <- kde(x=iris[,2:3])
plot(fhat, display="filled.contour2", cont=seq(10,90,by=10))
plot(fhat, display="persp", thin=3, border=1, col="white")

## trivariate example
fhat <- kde(x=iris[,2:4])
plot(fhat, drawpoints=TRUE)
```

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

Description

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

Usage

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

Arguments

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

Details

For `kde.loctest` objects, the function headers are

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

```
## bivariate
plot(x, col, add=FALSE, xlab="x", ylab="y", add.contour=FALSE,
  add.legend=TRUE, pos.legend="topright", ...)

## trivariate
plot(x, col, add=FALSE, xlab="x", ylab="y", zlab="z", box=TRUE, axes=TRUE,
  alphavec=c(0.5, 0.5), ...)
```

Value

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

See Also

kde.local.test

Examples

```
library(MASS)
data(crabs)
x1 <- crabs[crabs$sp=="B", c(4,6)]
x2 <- crabs[crabs$sp=="O", c(4,6)]
loct <- kde.local.test(x1=x1, x2=x2)
plot(loct)
```

---

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

Description

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

Usage

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

Arguments

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

Value

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

See Also

plot.kde

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Value

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

Examples

```
## bivariate
mus <- rbind(c(0,0), c(-1,1))
Sigma <- matrix(c(1, 0.7, 0.7, 1), nr=2, nc=2)
Sigmas <- rbind(Sigma, Sigma)
props <- c(1/2, 1/2)
plotmixt(mus=mus, Sigmas=Sigmas, props=props, display="filled.contour2")

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

---

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

Description

Pre-sphered or pre-scaled version of data.

Usage

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

Arguments

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

Details

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

Value

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

Examples

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

`unicef` *Unicef child mortality - life expectancy data*

Description

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

Usage

```
data(unicef)
```

Format

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

Source

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

`vector` *Vector and vector half operators*

Description

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

Usage

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

Arguments

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

References

Magnus, J.R. & Neudecker H.M. (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