Package: GenKern (via r-universe)

November 16, 2026

Version 1.2.60.9001

Title Functions for Generating and Manipulating Binned Kernel Density Estimates

Author David Lucy [aut, cre], Robert Aykroyd [aut]

Maintainer David Lucy <d.lucy@lancaster.ac.uk>

Description Computes generalised kernel density estimates. Compatibility updates by Renova.

Depends KernSmooth, utils

License GPL-2

NeedsCompilation yes

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

Date/Publication 2026-11-16 17:34:43 UTC

RemoteUrl https://github.com/renova-r/GenKern

RemoteRef renova-1.2.60.9001

RemoteSha 4b99c49f3a990c311c12adc688fdc9abcddab9a7

Contents

KernSec 2
KernSur 4
nearest 6
per 7

Index **10**

KernSec *Univariate kernel density estimate*

Description

Computes univariate kernel density estimate using Gaussian kernels which can also use non-equally spaced ordinates and adaptive bandwidths and local bandwidths

Usage

```
KernSec(x, xgridsize=100, xbandwidth, range.x)
```

Arguments

x	vector of x values
xgridsize	integer for number of ordinates at which to calculate the smoothed estimate: default=100
xbandwidth	value of x window width, or vector of local window widths, one for each x, or one for each range.x, or a vector of length xgridsize: default=dpik(x)
range.x	total range of the estimate in the x dimension, or a vector giving the x ordinates: default=range +- 1.5 * mean bandwidth

Value

returns two vectors:

xords	vector of ordinates
yden	vector of density estimates corresponding to each x ordinate

Acknowledgements

Written in collaboration with A.M.Pollard <<mark.pollard@rlaha.ox.ac.uk>> with the financial support of the Natural Environment Research Council (NERC) grant GR3/11395

Note

Slow code suitable for visualisation and display of p.d.f where highly generalised k.p.d.fs are needed - bkde is faster when uniformly grided, single bandwidth, k.p.d.fs are required, although in the univariate case you won't notice the difference.

This function doesn't use bins as such, it calculates the density at a set of points. These points can be thought of as 'bin centres' but in reality they're not.

For version 1.10 on local kernel density estimates can now be sent, so that a vector of bandwidths can be send which is the same length as that of the observations. This will give a density which is has a unique bandwidth for each observation. Or a vector of bandwidths can be sent which is the same length as that of the number of bins. This will give a unique bandwidth for each ordinate, and

is described in Wand & Jones (1995) *Kernal Smoothing*. It is for the user to supply this vector of bandwidths, possibly with some form of *pilot estimation*.

It should be noted that multi-element vectors which approximate the bin centres, can be sent rather than the extreme limits of the range; which means that the points at which the density is to be calculated need not be uniformly spaced.

If the default `xbandwidth` is to be used there **must** be at least five unique values for in the `x` vector. If not the function will return an error. If you don't have five unique values in the vector then send a value, or vector for `xbandwidth`

The number of ordinates defaults to the length of `range.x` if `range.x` is a vector of ordinates, otherwise it is `xgridsize`, or 100 if that isn't specified.

The option `na.rm` is no longer supported. The function will automatically remove NAs where appropriate and possible, and will return a warning.

Finally, the various modes of sending parameters can be mixed, ie: the extremes of the range can be sent to define the range for `x`, but a multi-element vector could be sent to define the ordinates in the `y` dimension, or, a vector could be sent to describe the bandwidth for each case in `x`.

Author(s)

David Lucy <<d.lucy@lancaster.ac.uk>> http://www.maths.lancs.ac.uk/~lucy/
Robert Aykroyd <<r.g.aykroyd@leeds.ac.uk>>http://www.amsta.leeds.ac.uk/~robert/

References

Lucy, D. Aykroyd, R.G. & Pollard, A.M.(2002) Non-parametric calibration for age estimation . *Applied Statistics* **51**(2): 183-196

See Also

`KernSur` `per` `density` `hist` `bkde` `bkde2D` `dpik`

Examples

```
x <- c(2,4,6,8,10)

z <- KernSec(x)    # simplest invocation
plot(z$xords, z$yden, type="l")

z <- KernSec(x, xbandwidth=2, range.x=c(0,8))
plot(z$xords, z$yden, type="l")

# local bandwidths
ords <- seq(from=0, to=14, length=100)
bands <- x/15
z <- KernSec(x, xbandwidth=bands, range.x=ords)
plot(z$xords, z$yden, type="l")         # should plot a wriggly line

bands <- seq(from=1, to=4, length=100) # improvise a pilot estimate
z <- KernSec(x, xbandwidth=bands, range.x=ords)
plot(z$xords, z$yden, type="l")
```

KernSur *Bivariate kernel density estimation*

Description

Compute bivariate kernel density estimate using five parameter Gaussian kernels which can also use non equally spaced and adaptive bandwidths

Usage

```
KernSur(x, y, xgridsize=100, ygridsize=100, correlation, xbandwidth,
ybandwidth, range.x, range.y, na.rm=FALSE)
```

Arguments

x	vector of x values
y	vector of y values
xgridsize	integer for number of ordinates at which to calculate the smoothed estimate: default=100
ygridsize	integer for number of ordinates at which to calculate the smoothed estimate: default=100
correlation	x,y correlation, or vector of local correlations: default=cor(x,y)
xbandwidth	value of x window width, or vector of local window widths: default=dpik(x)
ybandwidth	value of y window width, or vector of local window widths: default=dpik(y)
range.x	total range of the estimate in the x dimension, or a vector giving the x ordinates: default=range +- 1.5 * mean bandwidth
range.y	total range of the estimate in the y dimension, or a vector giving the y ordinates: default=range +- 1.5 * mean bandwidth
na.rm	NA behaviour: TRUE drops cases with NA's, FALSE stops function with a warning if NA's are detected: default=FALSE

Value

returns two vectors and a matrix:

xords	vector of ordinates at which the density has been estimated in the x dimension
yords	vector of ordinates at which the density has been estimated in the y dimension
zden	matrix of density for f(x,y) with dimensions xgridsize, ygridsize

Acknowledgements

Written in collaboration with A.M.Pollard <<mark.pollard@rlaha.ox.ac.uk>> with the financial support of the Natural Environment Research Council (NERC) grant GR3/11395

Note

Slow code suitable for visualisation and display of correlated p.d.f, where highly generalised k.p.d.fs are needed - bkde2D is much faster when uncorrelated, uniformly grided, single bandwidth, k.p.d.fs are required.

This function doesn't use bins as such, it calculates the density at a set of points in each dimension. These points can be thought of as 'bin centres' but in reality they're not.

From version 1.00 onwards a number of improvements have been made: NA's are now handled semi-convincingly by dropping if required. A multi-element vector of bandwidths associated with each case can be sent for either dimension, so it is possible to accept the default, give a fixed bandwidth, or a bandwidth associated with each case. A multi-element vector of correlations can be sent, rather than a single correlation.

It should be noted that if a vector is sent for correlation, or either bandwidth, they must be of the same length as the data vectors. Furthermore, vectors which approximate the bin centres, can be sent rather than the extreme limits in the range; which means that the points at which the density is to be calculated need not be uniformly spaced.

Unlike KernSec this function does not yet support local bandwidths.

If the default bandwidth is to be used there **must** be at least five unique values for in the x and y vectors. If not the function will return an error. If you don't have five unique values in the vector then send a value, or vector for bandwidth

The number of ordinates defaults to the length of range.x if range.x is a vector of ordinates, otherwise it is xgridsize, or 100 if that isn't specified.

Finally, the various modes of sending parameters can be mixed, ie: the extremes of the range can be sent to define the range for x, but a multi-element vector could be sent to define the ordinates in the y dimension, or, a vector could be sent to describe the bandwidth for each case in the x direction, and a single-element vector defines all bandwidths in the y.

Version 1.1-0 has a bugfix in that it now outputs the magnetude of the density function at the specified bi-variate points, not an approximation to the volumes.

Author(s)

David Lucy <<d.lucy@lancaster.ac.uk>> http://www.maths.lancs.ac.uk/~lucy/
Robert Aykroyd <<r.g.aykroyd@leeds.ac.uk>>http://www.amsta.leeds.ac.uk/~robert/

References

Lucy, D. Aykroyd, R.G. & Pollard, A.M.(2002) Non-parametric calibration for age estimation. *Applied Statistics* **51**(2): 183-196

See Also

KernSec per density hist bkde bkde2D dpik

Examples

```
x <- c(2,4,6,8,10)                    # make up some x-y data
y <- x
```

```
# calculate and plot a surface with zero correlation based on above data
op <- KernSur(x,y, xgridsize=50, ygridsize=50, correlation=0,
        xbandwidth=1, ybandwidth=1, range.x=c(0,13), range.y=c(0,13))
image(op$xords, op$yords, op$zden, col=terrain.colors(100), axes=TRUE)
contour(op$xords, op$yords, op$zden, add=TRUE)
box()

# re-calculate and re-plot the above using a 0.8 correlation
op <- KernSur(x,y, xgridsize=50, ygridsize=50, correlation=0.8,
         xbandwidth=1, ybandwidth=1, range.x=c(0,13), range.y=c(0,13))
image(op$xords, op$yords, op$zden, col=terrain.colors(100), axes=TRUE)
contour(op$xords, op$yords, op$zden, add=TRUE)
box()

# calculate and plot a surface of the above data with an ascending
# correlation and bandwidths and a vector of equally spaced ordinates
bands <- c(1,1.1,1.2,1.3,1.0)
cors <- c(0,-0.2,-0.4,-0.6, -0.7)
rnge.x <- seq(from=0, to=13, length=100)

op <- KernSur(x,y, xgridsize=50, ygridsize=50, correlation=cors,
        xbandwidth=bands, ybandwidth=bands, range.x=rnge.x, range.y=c(0,13))
image(op$xords, op$yords, op$zden, col=terrain.colors(100), axes=TRUE)
contour(op$xords, op$yords, op$zden, add=TRUE)
box()
```

`nearest` *Index of a vector nearest in value to a supplied value*

Description

Returns the index of a vector which contains the value closest to an arbitary value

Usage

```
nearest(x, xval, outside=FALSE, na.rm=FALSE)
```

Arguments

`x`	vector of values
`xval`	value to find the nearest value in `x` to
`outside`	if not set to `TRUE` the function returns an error if `xval` is outside the range of `x` - default `FALSE`
`na.rm`	NA behaviour: `TRUE` drops cases with NA's, `FALSE` stops function with a warning if NA's are detected: default=`FALSE`

Value

returns an integer:

index	the index of x with the value nearest to xval

Acknowledgements

Written in collaboration with A.M.Pollard <<mark.pollard@rlaha.ox.ac.uk>> with the financial support of the Natural Environment Research Council (NERC) grant GR3/11395

Note

The vector doesn't have to be in any particular order - this routine will just give the index of the nearest number. The only inconsistancy is that if the value of xval are not strictly within the range of the vector the function will return an error. To prevent this call with the outside=TRUE flag enabled. If there are many values which match the 'nearest' value then the function will return a vector of their indicies.

Author(s)

David Lucy <<d.lucy@lancaster.ac.uk>> http://www.maths.lancs.ac.uk/~lucy/
Robert Aykroyd <<r.g.aykroyd@leeds.ac.uk>>http://www.amsta.leeds.ac.uk/~robert/

Examples

```
# make up a vector
x <- c(1,2,2,2,2,2,3,4,5,6,7,8,9,10)
# conventional useage - xval within range should return 9
nearest(x, 4.7)
# xval - outside the range of x should return 14
nearest(x, 12.7, outside=TRUE)
# many 'nearest' values in x - should return - 2 3 4 5 6
nearest(x, 1.7)
# make x[3] an NA
x[3] <- NA
# returns - 2 4 5 6 - by enabling na.rm
nearest(x, 1.7, na.rm=TRUE)
```

---

per *Locate value for ith percentage point in a binned distribution*

Description

Calculates the value for the ith point in a binned distribution

Usage

```
per(den, vals, point, na.rm=FALSE, neg.rm=FALSE)
```

Arguments

den	vector of frequency or density values
vals	vector of values corresponding to the centres of the bins in den, or the bin break points
point	percentage point of the distribution ie: 0.50 is median
na.rm	behaviour for NA's in the vector of density values: FALSE (default) per() will fail with warning if NA's are detected, TRUE per() will assume that these values are really zeros
neg.rm	per() will also fail if any member of the density vector is negative (which can happen occasionally from density functions based on FFT), set this to TRUE to treat these values as zeros

Value

returns a value:

x	value of vals corresponding to the point position

Acknowledgements

Written in collaboration with A.M.Pollard <<mark.pollard@rlaha.ox.ac.uk>> with the financial support of the Natural Environment Research Council (NERC) grant GR3/11395

Note

Not restricted to uniform bin widths but due to linear interpolation gets less accurate as bin widths deviate from uniformity. The vectors must be in ascending order of bin centres bin break points. The density can be a frequency in that it doesn't have to sum to unity.

Out of character for the rest of the GenKern package this function does assume proper bins rather than ordinates, although if a density estimate has been generated using KernSec then the ordinate vector can be used as a first order approximation to bin centres.

Author(s)

David Lucy <<d.lucy@lancaster.ac.uk>> http://www.maths.lancs.ac.uk/~lucy/
Robert Aykroyd <<r.g.aykroyd@leeds.ac.uk>>http://www.amsta.leeds.ac.uk/~robert/

See Also

KernSur per density hist bkde bkde2D dpik

Examples

```
# make up some x-y data
x <- seq(1,100)
y <- dnorm(x, mean=40, sd=10)
plot(x,y)
# mark the median, 0.1 and 0.9 positions with vertical lines
```

```
abline(v=per(y,x,0.5))
abline(v=per(y,x,0.9))
abline(v=per(y,x,0.1))
# for a bimodal distribution which doesn't sum to one
x <- c(1:5)
y <- c(2,3,4,3,4)
per(y,x,0.5) # should return 3.25
# change the previous example to bin extremes
x <- c(1:6)
per(y,x,0.5) # should return 3.75
```

Index

∗ **arith**
nearest, 6
∗ **distribution**
KernSec, 2
KernSur, 4
∗ **nonparametric**
per, 7
∗ **smooth**
KernSec, 2
KernSur, 4

bkde, *2*, *3*, *5*, *8*
bkde2D, *3*, *5*, *8*

density, *3*, *5*, *8*
dpik, *3*, *5*, *8*

hist, *3*, *5*, *8*

KernSec, 2, *5*, *8*
KernSur, *3*, 4, *8*

nearest, 6

per, *3*, *5*, 7, *8*