Package ‘sparr’

September 17, 2018

Type Package

Title Spatial and Spatiotemporal Relative Risk

Version 2.2-13

Date 2018-09-16

Description Provides functions to estimate kernel-smoothed spatial and spatio-temporal densities and relative risk functions, and perform subsequent inference. Methodological details can be found in the accompanying tutorial: Davies et al. (2018) <DOI:10.1002/sim.7577>.

Depends R (>= 2.10.1), spatstat

Imports spatstat.utils, doParallel, parallel, foreach, misc3d

Suggests fftwtools (>= 0.9.8)

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

RoxygenNote 6.1.0

URL `http://tilmandavies.github.io/sparr`,
`https://github.com/tilmandavies/sparr`

BugReports `https://github.com/tilmandavies/sparr/issues`

Author Tilman M. Davies [aut, cre],
Jonathan C. Marshall [aut]

Maintainer Tilman M. Davies <`tdavies@maths.otago.ac.nz`>

Repository CRAN

Date/Publication 2018-09-16 22:10:15 UTC

R topics documented:

`sparr-package` *The sparr Package: Spatial and Spatiotemporal Relative Risk*

Description

Provides functions to estimate fixed and adaptive kernel-smoothed spatial relative risk surfaces via the density-ratio method and perform subsequent inference. Fixed-bandwidth spatiotemporal density and relative risk estimation is also supported.

Details

Package:	sparr
Date:	2018-09-16
Version:	2.2-13
License:	GPL (>= 2)

Kernel smoothing, and the flexibility afforded by this methodology, provides an attractive approach to estimating complex probability density functions.

The *spatial relative risk function*, constructed as a ratio of estimated case to control densities (Bithell, 1990; 1991; Kelsall and Diggle, 1995a,b), describes the variation in the 'risk' of the disease, given the underlying at-risk population. This is a technique that has been applied successfully for mainly exploratory purposes in a number of different analyses (see for example Sabel et al., 2000; Prince et al., 2001; Wheeler, 2007). It has also grown in popularity in very different fields that pose similarly styled research questions, such as ecology (e.g. Campos and Fedigan, 2014); physiology (Davies et al., 2013); and archaeology (e.g. Bevan, 2012; Smith et al. 2015).

This package provides functions for spatial (i.e. bivariate/planar/2D) kernel density estimation (KDE), implementing both fixed and 'variable' or 'adaptive' (Abramson, 1982) smoothing parameter options. A selection of bandwidth calculators for bivariate KDE and the relative risk function are provided, including one based on the maximal smoothing principle (Terrell, 1990), and others involving a leave-one-out cross-validation (see below). In addition, the ability to construct both Monte-Carlo and asymptotic *p*-value surfaces ('tolerance' contours of which signal statistically significant sub-regions of extremity in a risk surface - Hazelton and Davies, 2009; Davies and Hazelton, 2010) as well as some visualisation tools are provided.

Spatiotemporal estimation is also supported, largely following developments in Fernando and Hazelton (2014). This includes their fixed-bandwith kernel estimator of spatiotemporal densities, relative risk, and asymptotic tolerance contours.

Key content of `sparr` can be broken up as follows:

DATASETS/DATA GENERATION

`pbc` a case/control planar point pattern (`ppp.object`) concerning liver disease in northern England.

`fmd` an anonymised (jittered) case/control spatiotemporal point pattern of the 2001 outbreak of veterinary foot-and-mouth disease in Cumbria (courtesy of the Animal and Plant Health Agency, UK).

`burk` a spatiotemporal point pattern of Burkitt's lymphoma in Uganda; artificially simulated control data are also provided for experimentation.

Also available are a number of relevant additional spatial datasets built-in to the `spatstat` package (Baddeley and Turner, 2005; Baddeley et al., 2015) through `spatstat.data`, such as `chorley`, which concerns the distribution of laryngeal cancer in an area of Lancashire, UK.

`rimpoly` a wrapper function of `rpoint` to allow generated spatial point patterns based on a pixel `im`age to be returned with a polygonal `owin`.

SPATIAL

Bandwidth calculators

`OS` estimation of an isotropic smoothing parameter for fixed-bandwidth bivariate KDE, based on the oversmoothing principle introduced by Terrell (1990).

`NS` estimation of an isotropic smoothing parameter for fixed-bandwidth bivariate KDE, based on the asymptotically optimal value for a normal density (bivariate normal scale rule - see e.g. Wand and Jones, 1995).

`LSCV.density` a least-squares cross-validated (LSCV) estimate of an isotropic fixed bandwidth for bivariate, edge-corrected KDE (see e.g. Bowman and Azzalini, 1997).

`LIK.density` a likelihood cross-validated (LIK) estimate of an isotropic fixed bandwidth for bivariate, edge-corrected KDE (see e.g. Silverman, 1986).

`SLIK.adapt` an experimental likelihood cross-validation function for simultaneous global/pilot bandwidth selection for adaptive density estimates.

`BOOT.density` a bootstrap approach to optimisation of an isotropic fixed bandwidth for bivariate, edge-corrected KDE (see e.g. Taylor, 1989).

`LSCV.risk` Estimation of a jointly optimal, common isotropic case-control fixed bandwidth for the kernel-smoothed risk function based on the mean integrated squared error (MISE), a weighted MISE, or the asymptotic MISE (see respectively Kelsall and Diggle, 1995a; Hazelton, 2008; Davies, 2013).

Density and relative risk estimation

`bivariate.density` kernel density estimate of bivariate data; fixed or adaptive smoothing.

`multiscale.density` multi-scale adaptive kernel density estimates for multiple global bandwidths as per Davies and Baddeley (2018).

`multiscale.slice` a single adaptive kernel estimate based on taking a slice from a multi-scale estimate.

`risk` estimation of a (log) spatial relative risk function, either from data or pre-existing bivariate density estimates; fixed (Kelsall and Diggle, 1995a) or both asymmetric (Davies and Hazelton, 2010) and symmetric (Davies et al., 2016) adaptive estimates are possible.

`tolerance` calculation of asymptotic or Monte-Carlo p-value surfaces.

Visualisation

`S3` methods of the `plot` function; see `plot.bivden` for visualising a single bivariate density estimate from `bivariate.density`, `plot.rrs` for visualisation of a spatial relative risk function from `risk`, or `plot.msden` for viewing animations of multi-scale density estimates from `multiscale.density`.

`tol.contour` provides more flexibility for plotting and superimposing tolerance contours upon an existing plot of spatial relative risk (i.e. given output from `tolerance`).

Printing and summarising

`S3` methods (`print.bivden`, `print.rrs`, `print.msden`, `summary.bivden`, `summary.rrs`, and `summary.msden`) are available for the bivariate density, spatial relative risk, and multi-scale adaptive density objects.

SPATIOTEMPORAL

Bandwidth calculators

`OS.spattemp` estimation of an isotropic smoothing parameter for the spatial margin and another for the temporal margin for spatiotemporal densities, based on the 2D and 1D versions, respectively, of the oversmoothing principle introduced by Terrell (1990).

`NS.spattemp` as above, based on the 2D and 1D versions of the normal scale rule (Silverman, 1986).

`LSCV.spattemp` least-squares cross-validated (LSCV) estimates of scalar spatial and temporal bandwidths for edge-corrected spatiotemporal KDE.

`LIK.spattemp` as above, based on likelihood cross-validation.

`BOOT.spattemp` bootstrap bandwidth selection for the spatial and temporal margins; for spatiotemporal, edge-corrected KDE (Taylor, 1989).

Density and relative risk estimation

`spattemp.density` fixed-bandwidth kernel density estimate of spatiotemporal data.

`spattemp.risk` fixed-bandwidth kernel density estimate of spatiotemporal relative risk, either with a time-static or time-varying control density (Fernando and Hazelton, 2014).

`spattemp.slice` extraction function of the spatial density/relative risk at prespecified time(s).

Visualisation

`S3` methods of the `plot` function; see `plot.stden` for various options (including animation) for visualisation of a spatiotemporal density, and `plot.rrst` for viewing spatiotemporal relative risk surfaces (including animation and tolerance contour superimposition).

Printing and summarising objects

`S3` methods (`print.stden`, `print.rrst`, `summary.stden`, and `summary.rrst`) are available for the spatiotemporal density and spatiotemporal relative risk objects respectively.

Dependencies

The `sparr` package depends upon `spatstat`. In particular, the user should familiarise themselves with `ppp` objects and `im` objects, which are used throughout. For spatiotemporal density estimation, `sparr` is assisted by importing from the `misc3d` package, and for the experimental capabilities involving parallel processing, `sparr` also currently imports `doParallel`, `parallel`, and `foreach`.

Citation

To cite use of current versions of `sparr` in publications or research projects please use:

Davies, T.M., Marshall, J.C. and Hazelton, M.L. (2018) Tutorial on kernel estimation of continuous spatial and spatiotemporal relative risk, *Statistics in Medicine*, **37**(7), 1191-1221. <DOI:10.1002/sim.7577>

Old versions of `sparr` (<= 2.1-09) can be referenced by Davies et al. (2011) (see reference list).

Author(s)

T.M. Davies
Dept. of Mathematics & Statistics, University of Otago, Dunedin, New Zealand.
J.C. Marshall
Institute of Fundamantal Sciences, Massey University, Palmerston North, New Zealand.

Maintainer: T.M.D. `<tdavies@maths.otago.ac.nz>`

References

Abramson, I. (1982), On bandwidth variation in kernel estimates — a square root law, *Annals of Statistics*, **10**(4), 1217-1223.

Baddeley, A. and Turner, R. (2005), spatstat: an R package for analyzing spatial point patterns, *Journal of Statistical Software*, **12**(6), 1-42.

Baddeley, A., Rubak, E. and Turner, R. (2015) *Spatial Point Patterns: Methodology and Applications with R*, Chapman and Hall/CRC Press, UK.

Bevan A. (2012), Spatial methods for analysing large-scale artefact inventories. *Antiquity*, **86**, 492-506.

Bithell, J.F. (1990), An application of density estimation to geographical epidemiology, *Statistics in Medicine*, **9**, 691-701.

Bithell, J.F. (1991), Estimation of relative risk function, *Statistics in Medicine*, **10**, 1745-1751.

Bowman, A.W. and Azzalini, A. (1997), *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations.* Oxford University Press Inc., New York. ISBN 0-19-852396-3.

Campos, F.A. and Fedigan, L.M. (2014) Spatial ecology of perceived predation risk and vigilance behavior in white-faced capuchins, *Behavioral Ecology*, **25**, 477-486.

Davies, T.M. (2013), Jointly optimal bandwidth selection for the planar kernel-smoothed density-ratio, *Spatial and Spatio-temporal Epidemiology*, **5**, 51-65.

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Davies, T.M., Cornwall, J. and Sheard, P.W. (2013) Modelling dichotomously marked muscle fibre configurations, *Statistics in Medicine*, **32**, 4240-4258.

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Hazelton, M.L. and Marshall, J.C. (2011), `sparr`: Analyzing spatial relative risk using fixed and adaptive kernel density estimation in `R`, *Journal of Statistical Software* **39**(1), 1-14.

Davies, T.M., Jones, K. and Hazelton, M.L. (2016), Symmetric adaptive smoothing regimens for estimation of the spatial relative risk function, *Computational Statistics & Data Analysis*, **101**, 12-28.

Fernando, W.T.P.S. and Hazelton, M.L. (2014), Generalizing the spatial relative risk function, *Spatial and Spatio-temporal Epidemiology*, **8**, 1-10.

Hazelton, M.L. (2008), Letter to the editor: Kernel estimation of risk surfaces without the need for edge correction, *Statistics in Medicine*, **27**, 2269-2272.

Hazelton, M.L. and Davies, T.M. (2009), Inference based on kernel estimates of the relative risk function in geographical epidemiology, *Biometrical Journal*, **51**(1), 98-109.

Kelsall, J.E. and Diggle, P.J. (1995a), Kernel estimation of relative risk, *Bernoulli*, **1**, 3-16.

Kelsall, J.E. and Diggle, P.J. (1995b), Non-parametric estimation of spatial variation in relative risk, *Statistics in Medicine*, **14**, 2335-2342.

Prince, M. I., Chetwynd, A., Diggle, P. J., Jarner, M., Metcalf, J. V. and James, O. F. W. (2001), The geographical distribution of primary biliary cirrhosis in a well-defined cohort, *Hepatology* **34**, 1083-1088.

Sabel, C. E., Gatrell, A. C., Loytonen, M., Maasilta, P. and Jokelainen, M. (2000), Modelling exposure opportunitites: estimating relative risk for motor disease in Finland, *Social Science & Medicine* **50**, 1121-1137.

Smith, B.A., Davies, T.M. and Higham, C.F.W. (2015) Spatial and social variables in the Bronze Age phase 4 cemetery of Ban Non Wat, Northeast Thailand, *Journal of Archaeological Science: Reports*, **4**, 362-370.

Taylor, C.C. (1989) Bootstrap choice of the smoothing parameter in kernel density estimation, *Biometrika*, **76**, 705-712.

Terrell, G.R. (1990), The maximal smoothing principle in density estimation, *Journal of the American Statistical Association*, **85**, 470-477.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*, Fourth Edition, Springer, New York.

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

Wheeler, D. C. (2007), A comparison of spatial clustering and cluster detection techniques for childhood leukemia incidence in Ohio, 1996-2003, *International Journal of Health Geographics*, **6**(13).

`available.h0` *Available global bandwidth range*

Description

Gets universally available global bandwidths as represented by several multi-scale density estimate objects

Usage

```
available.h0(...)
```

Arguments

`...`	Any number of objects of class `msden`; possibly named.

Details

This simple function merely accesses and returns the maximum lower limit and minimum upper limit of all `h0range` components of the `msden` objects passed through `...`. Natural numeric error arising from any changes to the bandwidth-axis discretisation resolution in the creation of the `msden` objects (i.e. through the '`dimz`' argument) means individual global bandwidth ranges can differ slightly between affected multi-scale estimates, even if they are all applied to the same data set. Can additionally be useful when, for example, creating asymmetric relative risk surfaces based on slices of multi-scale densities with respect to the case and control data sets, because the bandwidth factors differ.

Throws an error if one or more of the `h0range` components is incompatible (i.e. all `h0range` components must overlap).

Value

A numeric vector of length 2 providing the range of available global bandwidths compatible with all supplied multi-scale density estimates.

Author(s)

T.M. Davies

See Also

`multiscale.density`, `multiscale.slice`

Examples

```
# See ?multiscale.slice
```

`bivariate.density` *Bivariate kernel density/intensity estimation*

Description

Provides an isotropic adaptive or fixed bandwidth kernel density/intensity estimate of bivariate/planar/2D data.

Usage

```
bivariate.density(pp, h0, hp = NULL, adapt = FALSE, resolution = 128,
  gamma.scale = "geometric", edge = c("uniform", "diggle", "none"),
  weights = NULL, intensity = FALSE, trim = 5, xy = NULL,
  pilot.density = NULL, leaveoneout = FALSE, parallelise = NULL,
  davies.baddeley = NULL, verbose = TRUE)
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data set to be smoothed.
`h0`	Global bandwidth for adaptive smoothing or fixed bandwidth for constant smoothing. A numeric value > 0.
`hp`	Pilot bandwidth (scalar, numeric > 0) to be used for fixed bandwidth estimation of a pilot density in the case of adaptive smoothing. If `NULL` (default), it will take on the value of `h0`. Ignored when `adapt = FALSE` or if `pilot.density` is supplied as a pre-defined pixel image.
`adapt`	Logical value indicating whether to perform adaptive kernel estimation. See ‘Details’.
`resolution`	Numeric value > 0. Resolution of evaluation grid; the density/intensity will be returned on a [`resolution` $\times$ `resolution`] grid.
`gamma.scale`	Scalar, numeric value > 0; controls rescaling of the variable bandwidths. Defaults to the geometric mean of the bandwidth factors given the pilot density (as per Silverman, 1986). See ‘Details’.
`edge`	Character string giving the type of edge correction to employ. `"uniform"` (default) corrects based on evaluation grid coordinate and `"diggle"` reweights each observation-specific kernel. Setting `edge = "none"` requests no edge correction. Further details can be found in the documentation for `density.ppp`.
`weights`	Optional numeric vector of nonnegative weights corresponding to each observation in `pp`. Must have length equal to `npoints(pp)`.

`intensity`	Logical value indicating whether to return an intensity estimate (integrates to the sample size over the study region), or a density estimate (default, integrates to 1).
`trim`	Numeric value > 0; controls bandwidth truncation for adaptive estimation. See 'Details'.
`xy`	Optional alternative specification of the evaluation grid; matches the argument of the same tag in `as.mask`. If supplied, `resolution` is ignored.
`pilot.density`	An optional pixel image (class `im`) giving the pilot density to be used for calculation of the variable bandwidths in adaptive estimation, **or** a `ppp.object` giving the data upon which to base a fixed-bandwidth pilot estimate using `hp`. If used, the pixel image *must* be defined over the same domain as the data given `resolution` or the supplied pre-set `xy` evaluation grid; **or** the planar point pattern data must be defined with respect to the same polygonal study region as in `pp`.
`leaveoneout`	Logical value indicating whether to compute and return the value of the density/intensity at each data point for an adaptive estimate. See 'Details'.
`parallelise`	Numeric argument to invoke parallel processing, giving the number of CPU cores to use when `leaveoneout = TRUE`. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`davies.baddeley`	An optional numeric vector of length 3 to control bandwidth partitioning for approximate adaptive estimation, giving the quantile step values for the variable bandwidths for density/intensity and edge correction surfaces and the resolution of the edge correction surface. May also be provided as a single numeric value. See 'Details'.
`verbose`	Logical value indicating whether to print a function progress bar to the console when `adapt = TRUE`.

Details

Given a data set $x_1, \ldots, x_n$ in 2D, the isotropic kernel estimate of its probability density function, $\hat{f}(x)$, is given by

$$\hat{f}(y) = n^{-1} \sum_{i=1}^{n} h(x_i)^{-2} K((y - x_i)/h(x_i))$$

where $h(x)$ is the bandwidth function, and $K(.)$ is the bivariate standard normal smoothing kernel. Edge-correction factors (not shown above) are also implemented.

Fixed The classic fixed bandwidth kernel estimator is used when `adapt = FALSE`. This amounts to setting $h(u)$ =`h0` for all u. Further details can be found in the documentation for `density.ppp`.

Adaptive Setting `adapt = TRUE` requests computation of Abramson's (1982) variable-bandwidth estimator. Under this framework, we have $h(u)$ =`h0`*min[$\tilde{f}(u)^{-1/2}$,$G*$`trim`]/γ, where $\tilde{f}(u)$ is a fixed-bandwidth kernel density estimate computed using the pilot bandwidth `hp`.

- Global smoothing of the variable bandwidths is controlled with the global bandwidth `h0`.

- In the above statement, G is the geometric mean of the "bandwidth factors" $\tilde{f}(x_i)^{-1/2}$; $i = 1, \ldots, n$. By default, the variable bandwidths are rescaled by $\gamma = G$, which is set with `gamma.scale = "geometric"`. This allows `h0` to be considered on the same scale as the smoothing parameter in a fixed-bandwidth estimate i.e. on the scale of the recorded data. You can use any other rescaling of `h0` by setting `gamma.scale` to be any scalar positive numeric value; though note this only affects γ – see the next bullet. When using a scale-invariant `h0`, set `gamma.scale = 1`.
- The variable bandwidths must be trimmed to prevent excessive values (Hall and Marron, 1988). This is achieved through `trim`, as can be seen in the equation for $h(u)$ above. The trimming of the variable bandwidths is universally enforced by the geometric mean of the bandwidth factors G independent of the choice of γ. By default, the function truncates bandwidth factors at five times their geometric mean. For stricter trimming, reduce `trim`, for no trimming, set `trim = Inf`.
- For even moderately sized data sets and evaluation grid `resolution`, adaptive kernel estimation can be rather computationally expensive. The argument `davies.baddeley` is used to approximate an adaptive kernel estimate by a sum of fixed bandwidth estimates operating on appropriate subsets of `pp`. These subsets are defined by "bandwidth bins", which themselves are delineated by a quantile step value $0 < \delta < 1$. E.g. setting $\delta = 0.05$ will create 20 bandwidth bins based on the 0.05th quantiles of the Abramson variable bandwidths. Adaptive edge-correction also utilises the partitioning, with pixel-wise bandwidth bins defined using the value $0 < \beta < 1$, and the option to decrease the resolution of the edge-correction surface for computation to a $[L \times L]$ grid, where $0 < L \leq$ `resolution`. If `davies.baddeley` is supplied as a vector of length 3, then the values `[1], [2], and [3]` correspond to the parameters δ, β, and $L_M = L_N$ in Davies and Baddeley (2018). If the argument is simply a single numeric value, it is used for both δ and β, with $L_M = L_N =$`resolution` (i.e. no edge-correction surface coarsening).
- Computation of leave-one-out values (when `leaveoneout = TRUE`) is done by brute force, and is therefore very computationally expensive for adaptive smoothing. This is because the leave-one-out mechanism is applied to both the pilot estimation and the final estimation stages. Experimental code to do this via parallel processing using the `foreach` routine is implemented. Fixed-bandwidth leave-one-out can be performed directly in `density.ppp`.

Value

If `leaveoneout = FALSE`, an object of class `"bivden"`. This is effectively a list with the following components:

`z`	The resulting density/intensity estimate, a pixel image object of class `im`.
`h0`	A copy of the value of `h0` used.
`hp`	A copy of the value of `hp` used.
`h`	A numeric vector of length equal to the number of data points, giving the bandwidth used for the corresponding observation in `pp`.
`him`	A pixel image (class `im`), giving the 'hypothetical' Abramson bandwidth at each pixel coordinate conditional upon the observed data. `NULL` for fixed-bandwidth estimates.
`q`	Edge-correction weights; a pixel `im`age if `edge = "uniform"`, a numeric vector if `edge = "diggle"`, and `NULL` if `edge = "none"`.

`gamma`	The value of γ used in scaling the bandwidths. `NA` if a fixed bandwidth estimate is computed.
`geometric`	The geometric mean G of the untrimmed bandwidth factors $\tilde{f}(x_i)^{-1/2}$. `NA` if a fixed bandwidth estimate is computed.
`pp`	A copy of the `ppp.object` initially passed to the `pp` argument, containing the data that were smoothed.

Else, if `leaveoneout = TRUE`, simply a numeric vector of length equal to the number of data points, giving the leave-one-out value of the function at the corresponding coordinate.

Author(s)

T.M. Davies and J.C. Marshall

References

Abramson, I. (1982). On bandwidth variation in kernel estimates — a square root law, *Annals of Statistics*, **10**(4), 1217-1223.

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Jones, K. and Hazelton, M.L. (2016), Symmetric adaptive smoothing regimens for estimation of the spatial relative risk function, *Computational Statistics & Data Analysis*, **101**, 12-28.

Diggle, P.J. (1985), A kernel method for smoothing point process data, *Journal of the Royal Statistical Society, Series C*, **34**(2), 138-147.

Hall P. and Marron J.S. (1988) Variable window width kernel density estimates of probability densities. *Probability Theory and Related Fields*, **80**, 37-49.

Marshall, J.C. and Hazelton, M.L. (2010) Boundary kernels for adaptive density estimators on regions with irregular boundaries, *Journal of Multivariate Analysis*, **101**, 949-963.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

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

Examples

```
data(chorley) # Chorley-Ribble data from package 'spatstat'

# Fixed bandwidth kernel density; uniform edge correction
chden1 <- bivariate.density(chorley,h0=1.5)

# Fixed bandwidth kernel density; diggle edge correction; coarser resolution
chden2 <- bivariate.density(chorley,h0=1.5,edge="diggle",resolution=64)
```

```
# Adaptive smoothing; uniform edge correction
chden3 <- bivariate.density(chorley,h0=1.5,hp=1,adapt=TRUE)

# Adaptive smoothing; uniform edge correction; partitioning approximation
chden4 <- bivariate.density(chorley,h0=1.5,hp=1,adapt=TRUE,davies.baddeley=0.025)

par(mfrow=c(2,2))
plot(chden1);plot(chden2);plot(chden3);plot(chden4)
```

`BOOT.density` *Bootstrap bandwidth for a spatial kernel density estimate*

Description

Isotropic fixed or global (for adaptive) bandwidth selection for a standalone 2D density based on bootstrap estimation of the MISE.

Usage

```
BOOT.density(pp, hlim = NULL, eta = NULL, type = c("fixed", "adaptive"),
  hp = NULL, edge = c("uniform", "none"), ref.density = NULL,
  resolution = 64, rmdiag = TRUE, sim.adapt = list(N = 50, B = 100,
  dimz = 64, objective = FALSE), parallelise = NA, verbose = TRUE, ...)
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data to be smoothed.
`hlim`	An optional vector of length 2 giving the limits of the optimisation routine with respect to the bandwidth. If `NULL`, the function attempts to choose this automatically.
`eta`	Fixed scalar bandwidth to use for the reference density estimate; if `NULL` it is calculated as the oversmoothing bandwidth of pp using `OS`. Ignored if `ref.density` is supplied. See 'Details'.
`type`	A character string indicating selection type. Either `"fixed"` (default) for selection of a constant bandwidth for the fixed-bandwidth estimator based on theory extended from results in Taylor (1989); or `"adaptive"` for selection of the global bandwidth for an adaptive kernel density. See 'Details'.
`hp`	Pilot bandwidth used for adaptive estimates in the bootstrap; see the argument of the same tag in `bivariate.density`. Ignored when `type = "fixed"` or when `ref.density` is supplied.
`edge`	Character string dictating edge correction for the bootstrapped estimates. `"uniform"` (default) corrects based on evaluation grid coordinate. Setting `edge="none"` requests no edge correction.

`ref.density`	Optional. An object of class `bivden` giving the reference density from which data will be generated. Based on theory, this must be a fixed-bandwidth estimate if `type = "fixed"`; see 'Details'. Must be edge-corrected if `edge = "uniform"`.
`resolution`	Spatial grid size; the optimisation will be based on a [`resolution` $\times$ `resolution`] density estimate.
`rmdiag`	Logical control value for removal of mirrored evaluation points as suggested by Taylor (1989) in the theoretical expression of the fixed-bandwidth MISE estimate. See 'Details'. Ignored when `type = "adaptive"`
`sim.adapt`	List of control values for bootstrap simulation in the adaptive case; see 'Details'. Ignored when `type = "fixed"`.
`parallelise`	Optional numeric argument to reduce computation time by invoking parallel processing, by giving the number of CPU cores to use in either evaluation (fixed) or in the actual bootstrap replicate generation (adaptive). Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`verbose`	Logical value indicating whether to print function progress during execution.
`...`	Optional arguments controlling scaling to be passed to `multiscale.density` for the adaptive bootstrap; ignored when `type = "fixed"`.

Details

For a 2D kernel density estimate $\hat{f}$ defined on $W \in R^2$, the mean integrated squared error (MISE) is given by $E[\int_W (\hat{f}(x) - f(x))^2 dx]$, where f is the corresponding true density. Given an observed data set X (argument `pp`) of n observations, this function finds the bandwidth h that minimises

$$E^*[\int_W (\hat{f}^*(x) - \hat{f}(x))^2 dx],$$

where $\hat{f}(x)$ is a density estimate of X constructed with 'reference' bandwidth η (argument `eta` or `ref.density`), and $\hat{f}^*(x)$ is a density estimate using bandwidth h of n observations X^* generated from $\hat{f}(x)$. The notation E^* denotes expectation with respect to the distribution of the X^*.

Fixed When `type = "fixed"`, the function assumes you want to select a constant bandwidth for use with the fixed-bandwith density estimator. This implementation is based on extending the remarkable results of Taylor (1989) (see also Sain et al., 1994), who demonstrates that when the Gaussian kernel is being used, we can find the optimal h with respect to the aforementioned bootstrap-estimated MISE without any actual resampling. This implementation extends these results to the bivariate setting, and allows for edge-correction of both the reference and bootstrap densities.

- Taylor (1989) does not distinguish between the reference bandwidth η and the target of optimisation, h, thus allowing the reference bandwidth to vary alongside the target in the optimisation. This is not optimal, and this function always assumes a static reference bandwidth. Hall et al. (1992) indicate that a generous amount of smoothing is to be preferred in the reference density (hence the default `eta` set using `OS`).
- If `ref.density` is supplied, it **must** be a fixed-bandwidth density estimate as an object of class `bivden` for validity of the theory. Edge-correction must be present if `edge = "uniform"`; and it must be evaluated on the same spatial domain as dictated by `Window(pp)` and

resolution. If unsupplied, the function internally computes an appropriate fixed-bandwidth density estimate using eta as the reference bandwidth.

- Finally, Taylor (1989) argues it is preferable to avoid summation at identical evaluation grid points in the expression for the optimal bandwidth, which is performed when rmdiag = TRUE. Setting rmdiag = FALSE disables this correction.

Adaptive When type = "adaptive", the function assumes you want to select a global bandwidth (argument h0 in bivariate.density) for use in 2D adaptive kernel density estimation.

- An expression similar to Taylor (1989) is not possible for the adaptive estimator. Thus, in the adaptive setting, the optimal bootstrap bandwidth is calculated by brute force as was performed in Davies and Baddeley (2018) by taking advantage of the multiscale estimation theory implemented in multiscale.density. The value that minimises an interpolating cubic spline of the estimated MISE on bandwidth is identified as the optimal global bandwidth.
- The user can pass either a fixed or adaptive bivden object to ref.density. If this is the case, hp is ignored and the pilot bandwidth for each iteration of the bootstrap in estimation of the $\hat{f}^*(x)$ uses ref.density$hp (if ref.density is adaptive) or ref.density$h0 (if ref.density is fixed). When ref.density is unsupplied, the function uses a fixed-bandwidth kernel estimate with bandwidth eta as the reference density, and if additionally hp is unsupplied, the same value eta is used for the constant pilot bandwidth.
- Control over the bootstrap is achieved with four optional named arguments passed as a list to sim.adapt. N controls the number of bootstrap iterates per bandwidth; B controls the resolution of the sequence of bandwidths trialled (i.e. between hlim[1] and hlim[2]); dimz specifies the resolution of the bandwidth axis in the trivariate convolution evaluated by multiscale.density; and objective specifies whether to return the set of estimated MISEs for all bandwidths (nice to plot), or merely the optimal bandwidth (see 'Value').
- The ... are intended for any relevant optional arguments to be passed to the internal call to multiscale.density, such as gamma.scale or trim.

Value

The optimal fixed or global (for adaptive) scalar bandwidth. If simargs$objective = TRUE for the adaptive bootstrap, the return object is instead a [simargs$B $x2$] matrix, with the first column giving the trialled bandwidth and the second giving the corresponding value of the estimated bootstrap MISE.

Warning

Even with the implemented computational tricks, bootstrapping for bandwidth selection for spatial data is still computationally demanding, especially for adaptive kernel estimates. The user can reduce this time by keeping the evaluation grid at modest resolutions, and experimenting with parallelising the internal loops via parallelise. The 'Examples' section offers some rough indications of evaluation times on this author's local machine.

Author(s)

T.M. Davies

References

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Hall, P., Marron, J.S. and Park, B.U. (1992) Smoothed cross-validation, *Probability Theory and Related Fields*, **92**, 1-20.

Sain, S.R., Baggerly, K.A. and Scott, D.W. (1994) Cross-validation of multivariate densities, *Journal of the American Statistical Association*, **89**, 807-817.

Taylor, C.C. (1989) Bootstrap choice of the smoothing parameter in kernel density estimation, *Biometrika*, **76**, 705-712.

See Also

`bivariate.density`, `OS`, `multiscale.density`

Examples

```
data(pbc)

## Fixed bandwidth selection ##
BOOT.density(pbc) # ~20 secs
BOOT.density(pbc,eta=OS(pbc)/2) # halve default reference bandwidth
BOOT.density(pbc,eta=OS(pbc)*2) # double default reference bandwidth

# supplying pre-defined reference density as fixed-bandwidth 'bivden' object
pbcfix <- bivariate.density(pbc,h0=2.5,resolution=64)
system.time(hfix <- BOOT.density(pbc,ref.density=pbcfix,parallelise=4)) # parallelisation; 14 secs
hfix

## Global (for adaptive) bandwidth selection ##
# ~200 secs next line; use 'parallelise' for speedup
system.time(hada <- BOOT.density(pbc,type="adaptive")) # minimal usage for adaptive bootstrap
hada

# ~80 secs next line. Set custom h limits; increase reference bandwidth;
#    set custom pilot bandwidth; return objective function
system.time(hada <- BOOT.density(pbc,hlim=c(0.9,8),eta=3.5,type="adaptive",
                                 hp=OS(pbc)/2,parallelise=6,
                                 sim.adapt=list(objective=TRUE)))
hada[which.min(hada[,2]),1]
plot(hada);abline(v=hada[which.min(hada[,2]),1],col=2)
```

BOOT.spattemp *Bootstrap bandwidths for a spatiotemporal kernel density estimate*

Description

Bandwidth selection for standalone spatiotemporal density/intensity based on bootstrap estimation of the MISE, providing an isotropic scalar spatial bandwidth and a scalar temporal bandwidth.

Usage

```
BOOT.spattemp(pp, tt = NULL, tlim = NULL, eta = NULL, nu = NULL,
  sedge = c("uniform", "none"), tedge = sedge, ref.density = NULL,
  sres = 64, tres = sres, start = NULL, verbose = TRUE)
```

Arguments

`pp`	An object of class `ppp` giving the spatial coordinates of the observations to be smoothed. Possibly marked with the time of each event; see argument `tt`.
`tt`	A numeric vector of equal length to the number of points in `pp`, giving the time corresponding to each spatial observation. If unsupplied, the function attempts to use the values in the `marks` attribute of the `ppp.object` in `pp`.
`tlim`	A numeric vector of length 2 giving the limits of the temporal domain over which to smooth. If supplied, all times in `tt` must fall within this interval (equality with limits allowed). If unsupplied, the function simply uses the range of the observed temporal values.
`eta`	Fixed scalar bandwidth to use for the spatial margin of the reference density estimate; if `NULL` it is calculated as the oversmoothing bandwidth of `pp` using `OS`. Ignored if `ref.density` is supplied. See 'Details'.
`nu`	Fixed scalar bandwidth to use for the temporal margin of the reference density estimate; if `NULL` it is calculated from `tt` using the univariate version of Terrell's (1990) oversmoothing principle. Ignored if `ref.density` is supplied. See 'Details'.
`sedge`	Character string dictating spatial edge correction. `"uniform"` (default) corrects based on evaluation grid coordinate. Setting `sedge="none"` requests no edge correction.
`tedge`	As `sedge`, for temporal edge correction.
`ref.density`	Optional. An object of class `stden` giving the reference density from which data is assumed to originate in the bootstrap. Must be spatially edge-corrected if `sedge = "uniform"`.
`sres`	Numeric value > 0. Resolution of the [`sres` $\times$ `sres`] evaluation grid in the spatial margin.
`tres`	Numeric value > 0. Resolution of the evaluation points in the temporal margin as defined by the `tlim` interval. If unsupplied, the density is evaluated at integer values between `tlim[1]` and `tlim[2]`.

`start`	Optional positive numeric vector of length 2 giving starting values for the internal call to `optim`, in the order of (<spatial bandwidth>, <temporal bandwidth>).
`verbose`	Logical value indicating whether to print a function progress bar to the console during evaluation.

Details

For a spatiotemporal kernel density estimate $\hat{f}$ defined on $WxT \in R^3$, the mean integrated squared error (MISE) is given by $E[\int_W \int_T (\hat{f}(x,t) - f(x,t))^2 dtdx]$, where f is the corresponding true density. Given observed spatiotemporal locations X (arguments `pp` and `tt`) of n observations, this function finds the scalar spatial bandwidth h and scalar temporal bandwidth λ that jointly minimise

$$E^*[\int_W \int_T (\hat{f}^*(x,t) - \hat{f}(x,t))^2 dtdx],$$

where $\hat{f}(x,t)$ is a density estimate of X constructed with ‘reference’ bandwidths η (spatial; argument `eta`) and ν (temporal; argument `nu`); $\hat{f}^*(x,t)$ is a density estimate using bandwidths h and λ of n observations X^* generated from $\hat{f}(x,t)$. The notation E^* denotes expectation with respect to the distribution of the X^*. The user may optionally supply `ref.density` as an object of class `stden`, which must be evaluated on the same spatial and temporal domains W and T as the data (arguments `pp`, `tt`, and `tlim`). In this case, the reference bandwidths are extracted from this object, and `eta` and `nu` are ignored.

This function is based on an extension of the theory of Taylor (1989) to the spatiotemporal domain and to cope with the inclusion of edge-correction factors. No resampling is necessary due to the theoretical properties of the Gaussian kernel.

Value

A numeric vector of length 2 giving the jointly optimised spatial and temporal bandwidths (named `h` and `lambda` respectively).

Warning

Bootstrapping for spatiotemporal bandwidth selection for spatiotemporal data is very computationally demanding. Keeping `verbose = TRUE` offers an indication of the computational burden by printing each pair of bandwidths at each iteration of the `optim`isation routine. The ‘Examples’ section also offers some rough indications of evaluation times on this author’s local machine.

Author(s)

T. M. Davies

References

Taylor, C.C. (1989) Bootstrap choice of the smoothing parameter in kernel density estimation, *Biometrika*, **76**, 705-712.

See Also

`LSCV.spattemp`, `spattemp.density`

Examples

```
data(burk) # Burkitt's Uganda lymphoma data
burkcas <- burk$cases

#~85 secs
hlam1 <- BOOT.spattemp(burkcas)

#~75 secs. Widen time limits, reduce ref. bw.
hlam2 <- BOOT.spattemp(burkcas,tlim=c(400,5800),eta=8,nu=450)

#~150 secs. Increase ref. bw., custom starting vals
hlam3 <- BOOT.spattemp(burkcas,eta=20,nu=800,start=c(7,400))

rbind(hlam1,hlam2,hlam3)
```

---

burk *Burkitt's lymphoma in Uganda*

Description

Data of the spatiotemporal locations of Burkitt's lymphoma in the Western Nile district of Uganda from 1960 to 1975.

Format

`burk` is a named list with three members:

`$cases` An object of class `ppp` giving the spatial locations (eastings/northings) of the 188 cases of Burkitt's lymphoma recorded in individuals of various ages (mostly children); the spatial study region as a polygonal `owin`; as well as the time (in days since 1/1/1960) of each observation stored as the `marks` of the points.

`$cases.age` A numeric vector of length 188 giving the age of each individual in `$cases`.

`$controls` An object of class `ppp` giving 500 **artificially simulated** spatial-only observations to pose as a ‘control’ data set representing the at-risk population. The data were generated from a smooth kernel estimate of the spatial margin of the cases. The similarity between the case point distribution and the true at-risk population dispersion can be seen in e.g. Figure 2 of Middleton and Greenland (1954).

Source

The case data were extracted from the `burkitt` object of the `splancs` R package; see

Rowlingson B. and Diggle P.J. (2017), splancs: Spatial and Space-Time Point Pattern Analysis, R package version 2.01-40; `https://CRAN.R-project.org/package=splancs`.

References

Bailey, T.C. and Gatrell, A.C. (1995), *Interactive spatial data analysis*, Longman; Harlow.

Middleton, J.F.M. and Greenland, D.J. (1954), Land and population in West Nile District, Uganda, *The Geographical Journal*, **120**, 446–455.

Examples

```
data(burk)
summary(burk$cases)

par(mfrow=c(1,3))
plot(burk$cases)
plot(burk$controls)
plot(density(marks(burk$cases)),xlim=range(marks(burk$cases)))
```

---

fft2d *2D fast-Fourier wrapper around 'fftwtools' or 'stats' package*

Description

Utilises the Fastest Fourier Transform in the West (FFTW) via the 'fftwtools' package if available, else reverts to built-in functionality

Usage

```
fft2d(x, inverse = FALSE, fftw = sparr:::fftw_available())
```

Arguments

`x`	A numeric matrix to be transformed.
`inverse`	Whether it should compute the inverse transform (defaults to `FALSE`).
`fftw`	Whether the `fftwtools` R package is available.

Details

This function is called wherever `sparr` seeks to perform a 2D fast-Fourier transform. Where available, computational expense is noticeably reduced by appealing to routines in the independent 'FFTW' toolbox. The user is encouraged to install the corresponding R package `fftwtools` from CRAN; this function will automatically detect and use the faster option, otherwise will defer to the built-in `fft`.

Value

The fast-Fourier (inverse) transform. A complex-valued matrix of the same size as `x`.

Author(s)

J.C. Marshall

Examples

```
# System check
sparr:::fftw_available()

system.time(fft(matrix(1:2000^2,2000)))
system.time(fft2d(matrix(1:2000^2,2000)))
```

---

fmd *Veterinary foot-and-mouth disease outbreak data*

Description

Data of the spatial locations and time of farms infected by veterinary foot-and-mouth disease in the county of Cumbria, UK, over a course of nearly 250 days between February and August in 2001. There are 410 infected farms (the cases), and 1866 uninfected farms (the controls). The data have been jittered and randomly thinned by an unspecified amount to preserve anonymity.

Format

`fmd` is a named list with two members:

`$cases` An object of class `ppp` giving the spatial locations of the 410 infected farms within a polygonal study region representing the county of Cumbria. The `marks` component of this object contain the integer day of infection (from beginning of study period).

`$controls` An object of class `ppp` defined over the same spatial study region with the locations of the 1866 uninfected farms.

Acknowledgements

The Animal and Plant Health Agency (APHA), UK, provided permission to use this dataset.

References

Fernando, W.T.P.S. and Hazelton, M.L. (2014), Generalizing the spatial relative risk function, *Spatial and Spatio-temporal Epidemiology*, **8**, 1-10.

Keeling M, Woolhouse M, Shaw D, Matthews L, Chase-Topping M, Haydon D, et al. (2001), Dynamics of the 2001 UK foot and mouth epidemic: stochastic dispersal in a heterogeneous landscape, *Science*, **294**, 813-817.

Lawson A, Zhou H. (2005), Spatial statistical modeling of disease outbreaks with particular reference to the UK foot and mouth disease (FMD) epidemic of 2001, *Preventative Veterinary Medicine*, **71**, 141-156.

Examples

```
data(fmd)
summary(fmd$cases)
summary(fmd$controls)

par(mfrow=c(1,2))
plot(fmd$cases)
plot(fmd$controls)
```

`LIK.density` *Cross-validation bandwidths for spatial kernel density estimates*

Description

Isotropic fixed or global (for adaptive) bandwidth selection for standalone 2D density/intensity based on either unbiased least squares cross-validation (LSCV) or likelihood (LIK) cross-validation.

Usage

```
LIK.density(pp, hlim = NULL, hseq = NULL, resolution = 64,
  edge = TRUE, auto.optim = TRUE, type = c("fixed", "adaptive"),
  seqres = 30, parallelise = NULL, zero.action = 0, verbose = TRUE,
  ...)

LSCV.density(pp, hlim = NULL, hseq = NULL, resolution = 64,
  edge = TRUE, auto.optim = TRUE, type = c("fixed", "adaptive"),
  seqres = 30, parallelise = NULL, zero.action = 0, verbose = TRUE,
  ...)
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data to be smoothed.
`hlim`	An optional vector of length 2 giving the limits of the optimisation routine with respect to the bandwidth. If unspecified, the function attempts to choose this automatically.
`hseq`	An optional increasing sequence of bandwidth values at which to manually evaluate the optimisation criterion. Used only in the case `(!auto.optim && is.null(hlim))`.
`resolution`	Spatial grid size; the optimisation will be based on a [`resolution` $\times$ `resolution`] density estimate.
`edge`	Logical value indicating whether to edge-correct the density estimates used.
`auto.optim`	Logical value indicating whether to automate the numerical optimisation using `optimise`. If `FALSE`, the optimisation criterion is evaluated over `hseq` (if supplied), or over a seqence of values controlled by `hlim` and `seqres`.

`type`	A character string; `"fixed"` (default) performs classical leave-one-out cross-validation for the fixed-bandwidth estimator. Alternatively, `"adaptive"` utilises multiscale adaptive kernel estimation (Davies & Baddeley, 2018) to run the cross-validation in an effort to find a suitable global bandwidth for the adaptive estimator. Note that data points are not 'left out' of the pilot density estimate when using this option (this capability is currently in development). See also the entry for `...`.
`seqres`	Optional resolution of an increasing sequence of bandwidth values. Only used if `(!auto.optim && is.null(hseq))`.
`parallelise`	Numeric argument to invoke parallel processing, giving the number of CPU cores to use when `!auto.optim` **and** `type = "fixed"`. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`zero.action`	A numeric integer, either `-1`, `0` (default), `1` or `2` controlling how the function should behave in response to numerical errors at very small bandwidths, when such a bandwidth results in one or more zero or negative density values during the leave-one-out computations. See 'Details'.
`verbose`	Logical value indicating whether to provide function progress commentary.
`...`	Additional arguments controlling pilot density estimation and multi-scale bandwidth-axis resolution when `type = "adaptive"`. Relevant arguments are `hp`, `pilot.density`, `gamma.scale`, and `trim` from `bivariate.density`; and `dimz` from `multiscale.density`. If `hp` is missing and required, the function makes a (possibly recursive) call to `LSCV.density` to set this using fixed-bandwidth LSCV. The remaining defaults are `pilot.density = pp`, `gamma.scale = "geometric"`, `trim = 5`, and `dimz = resolution`.

Details

This function implements the bivariate, edge-corrected versions of fixed-bandwidth least squares cross-validation and likelihood cross-validation as outlined in Sections 3.4.3 and 3.4.4 of Silverman (1986) in order to select an optimal fixed smoothing bandwidth. With `type = "adaptive"` it may also be used to select the global bandwidth for adaptive kernel density estimates, making use of multi-scale estimation (Davies and Baddeley, 2018) via `multiscale.density`. Note that for computational reasons, the leave-one-out procedure is not performed on the pilot density in the adaptive setting; it is only performed on the final stage estimate. Current development efforts include extending this functionality, see `SLIK.adapt`. See also 'Warning' below.

Where `LSCV.density` is based on minimisation of an unbiased estimate of the mean integrated squared error (MISE) of the density, `LIK.density` is based on maximisation of the cross-validated leave-one-out average of the log-likelihood of the density estimate with respect to h.

In both functions, the argument `zero.action` can be used to control the level of severity in response to small bandwidths that result (due to numerical error) in at least one density value being zero or less. When `zero.action = -1`, the function strictly forbids bandwidths that would result in one or more *pixel* values of a kernel estimate of the original data (i.e. anything over the whole region) being zero or less—this is the most restrictive truncation. With `zero.action = 0` (default), the function automatically forbids bandwidths that yield erroneous values at the leave-one-out data point locations only. With `zero.action = 1`, the minimum machine value (see `.Machine$double.xmin` at the prompt) is used to replace these individual leave-one-out values. When `zero.action = 2`, the minimum value of the valid (greater than zero) leave-one-out values is used to replace any erroneous leave-one-out values.

Value

A single numeric value of the estimated bandwidth (if `auto.optim = TRUE`). Otherwise, a [`seqres` x 2] matrix giving the bandwidth sequence and corresponding CV function value.

Warning

Leave-one-out CV for bandwidth selection in kernel density estimation is notoriously unstable in practice and has a tendency to produce rather small bandwidths, particularly for spatial data. Satisfactory bandwidths are not guaranteed for every application; `zero.action` can curb adverse numeric effects for very small bandwidths during the optimisation procedures. This method can also be computationally expensive for large data sets and fine evaluation grid resolutions. The user may also need to experiment with adjusting `hlim` to find a suitable minimum.

Author(s)

T. M. Davies

References

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

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

See Also

`SLIK.adapt` and functions for bandwidth selection in package `spatstat`: `bw.diggle`; `bw.ppl`; `bw.scott`; `bw.frac`.

Examples

```
data(pbc)
pbccas <- split(pbc)$case

LIK.density(pbccas)
LSCV.density(pbccas)


#* FIXED

# custom limits
LIK.density(pbccas,hlim=c(0.01,4))
LSCV.density(pbccas,hlim=c(0.01,4))
```

```
# disable edge correction
LIK.density(pbccas,hlim=c(0.01,4),edge=FALSE)
LSCV.density(pbccas,hlim=c(0.01,4),edge=FALSE)

# obtain objective function
hcv <- LIK.density(pbccas,hlim=c(0.01,4),auto.optim=FALSE)
plot(hcv);abline(v=hcv[which.max(hcv[,2]),1],lty=2,col=2)

#* ADAPTIVE
LIK.density(pbccas,type="adaptive")
LSCV.density(pbccas,type="adaptive")

# change pilot bandwidth used
LIK.density(pbccas,type="adaptive",hp=2)
LSCV.density(pbccas,type="adaptive",hp=2)
```

LIK.spattemp *Cross-validation bandwidths for spatiotemporal kernel density estimates*

Description

Bandwidth selection for standalone spatiotemporal density/intensity based on either unbiased least squares cross-validation (LSCV) or likelihood (LIK) cross-validation, providing an isotropic scalar spatial bandwidth and a scalar temporal bandwidth.

Usage

```
LIK.spattemp(pp, tt = NULL, tlim = NULL, sedge = c("uniform", "none"),
  tedge = sedge, parallelise = NA, start = NULL, verbose = TRUE)

LSCV.spattemp(pp, tt = NULL, tlim = NULL, sedge = c("uniform", "none"),
  tedge = sedge, sres = 64, tres = sres, parallelise = NA,
  start = NULL, verbose = TRUE)
```

Arguments

`pp`	An object of class `ppp` giving the spatial coordinates of the observations to be smoothed. Possibly marked with the time of each event; see argument `tt`.
`tt`	A numeric vector of equal length to the number of points in pp, giving the time corresponding to each spatial observation. If unsupplied, the function attempts to use the values in the `marks` attribute of the `ppp.object` in pp.
`tlim`	A numeric vector of length 2 giving the limits of the temporal domain over which to smooth. If supplied, all times in `tt` must fall within this interval (equality with limits allowed). If unsupplied, the function simply uses the range of the observed temporal values.

`sedge`	Character string dictating spatial edge correction. `"uniform"` (default) corrects based on evaluation grid coordinate. Setting `sedge="none"` requests no edge correction.
`tedge`	As `sedge`, for temporal edge correction.
`sres`	Numeric value > 0. Resolution of the [`sres` $\times$ `sres`] evaluation grid in the spatial margin.
`tres`	Numeric value > 0. Resolution of the evaluation points in the temporal margin as defined by the `tlim` interval. If unsupplied, the density is evaluated at integer values between `tlim[1]` and `tlim[2]`.
`parallelise`	Optional numeric argument to invoke parallel processing, by giving the number of CPU cores to use optimisation. This is only useful for larger data sets of many thousand observations. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`start`	Optional positive numeric vector of length 2 giving starting values for the internal call to `optim`, in the order of (<spatial bandwidth>, <temporal bandwidth>).
`verbose`	Logical value indicating whether to print a function progress bar to the console during evaluation.

Value

A numeric vector of length 2 giving the jointly optimised spatial and temporal bandwidths (named `h` and `lambda` respectively).

Warning

Leave-one-out CV for bandwidth selection in kernel density estimation is notoriously unstable in practice and has a tendency to produce rather small bandwidths in the fixed bandwidth case. Satisfactory bandwidths are not guaranteed for every application. This method can also be computationally expensive for large data sets and fine evaluation grid resolutions.

Author(s)

T. M. Davies

References

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

See Also

`BOOT.spattemp`, `spattemp.density`

Examples

```
data(burk) # Burkitt's Uganda lymphoma data
burkcas <- burk$cases

hlam1 <- LSCV.spattemp(burkcas) #~9 secs
hlam2 <- LSCV.spattemp(burkcas,tlim=c(400,5800))
hlam3 <- LSCV.spattemp(burkcas,start=c(7,400))
rbind(hlam1,hlam2,hlam3)

hlam1 <- LIK.spattemp(burkcas) #~3 secs
hlam2 <- LIK.spattemp(burkcas,tlim=c(400,5800))
hlam3 <- LIK.spattemp(burkcas,start=c(7,400))
rbind(hlam1,hlam2,hlam3)
```

`LSCV.risk` *Jointly optimal bandwidth selection for the spatial relative risk function*

Description

Methods to find a jointly optimal, common case-control isotropic bandwidth for use in estimation of the fixed or adaptive kernel-smoothed relative risk function.

Usage

```
LSCV.risk(f, g = NULL, hlim = NULL, hseq = NULL, type = c("fixed",
  "adaptive"), method = c("kelsall-diggle", "hazelton", "davies"),
  resolution = 64, edge = TRUE, hp = NULL,
  pilot.symmetry = c("none", "f", "g", "pooled"), auto.optim = TRUE,
  seqres = 30, parallelise = NA, verbose = TRUE, ...)
```

Arguments

`f`	Either a pre-calculated object of class `bivden` representing the 'case' (numerator) density estimate, or an object of class `ppp` giving the observed case data. Alternatively, if `f` is `ppp` object with dichotomous factor-valued `marks`, the function treats the first level as the case data, and the second as the control data, obviating the need to supply g.
`g`	As for `f`, for the 'control' (denominator) density; this object must be of the same class as `f`. Ignored if, as stated above, `f` contains both case and control observations.
`hlim`	An optional vector of length 2 giving the limits of the optimisation routine with respect to the bandwidth. If unspecified, the function attempts to choose this automatically.

`hseq`	An optional increasing sequence of bandwidth values at which to manually evaluate the optimisation criterion. Used only in the case `(!auto.optim && is.null(hlim))`.
`type`	A character string; `"fixed"` (default) performs classical leave-one-out cross-validation for a jointly optimal fixed bandwidth. Alternatively, `"adaptive"` utilises multiscale adaptive kernel estimation (Davies & Baddeley, 2018) to run the cross-validation in an effort to find a suitable jointly optimal, common global bandwidth for the adaptive relative risk function. See 'Details'.
`method`	A character string controlling the selector to use. There are three types, based on either the mean integrated squared error (MISE) (Kelsall and Diggle, 1995; default – `method = "kelsall-diggle"`); a weighted MISE (Hazelton, 2008 – `method = "hazelton"`); or an approximation to the asymptotic MISE (Davies, 2013 – `method = "davies"`). See 'Details'.
`resolution`	Spatial grid size; the optimisation will be based on a [`resolution` × `resolution`] density estimate.
`edge`	Logical value indicating whether to edge-correct the density estimates used.
`hp`	A single numeric value or a vector of length 2 giving the pilot bandwidth(s) to be used for estimation of the pilot densities for adaptive risk surfaces. Ignored if `type = "fixed"`.
`pilot.symmetry`	A character string used to control the type of symmetry, if any, to use for the bandwidth factors when computing an adaptive relative risk surface. See 'Details'. Ignored if `type = "fixed"`.
`auto.optim`	Logical value indicating whether to automate the numerical optimisation using `optimise`. If `FALSE`, the optimisation criterion is evaluated over `hseq` (if supplied), or over a seqence of values controlled by `hlim` and `seqres`.
`seqres`	Optional resolution of an increasing sequence of bandwidth values. Only used if `(!auto.optim && is.null(hseq))`.
`parallelise`	Numeric argument to invoke parallel processing, giving the number of CPU cores to use when `!auto.optim`. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`verbose`	Logical value indicating whether to provide function progress commentary.
`...`	Additional arguments such as `dimz` and `trim` to be passed to the internal calls to `multiscale.density`.

Details

Given the established preference of using a common bandwidth for both case and control density estimates when constructing a relative risk surface, This function calculates a 'jointly optimal', common isotropic LSCV bandwidth for the (Gaussian) kernel-smoothed relative risk function (case-control density-ratio). It can be shown that choosing a bandwidth that is equal for both case and control density estimates is preferable to computing 'separately optimal' bandwidths (Kelsall and Diggle, 1995). The user can choose to either calculate a common smoothing parameter for a fixed-bandwidth relative risk surface (`type = "fixed"`; default), or a common global bandwidth for an adaptive risk surface (`type = "adaptive"`). See further comments below.

- `method = "kelsall-diggle"`: the function computes the common bandwidth which minimises the approximate mean integrated squared error (MISE) of the log-transformed risk surface (Kelsall and Diggle, 1995).

- `method = "hazelton"`: the function minimises a *weighted-by-control* MISE of the (raw) relative risk function (Hazelton, 2008).
- `method = "davies"`: the optimal bandwidth is one that minimises a crude plug-in approximation to the *asymptotic* MISE (Davies, 2013). Only possible for `type = "fixed"`.

For jointly optimal, common global bandwidth selection when `type = "adaptive"`, the optimisation routine utilises `multiscale.density`. Like `LSCV.density`, the leave-one-out procedure does not affect the pilot density, for which additional control is offered via the `hp` and `pilot.symmetry` arguments. The user has the option of obtaining a so-called *symmetric* estimate (Davies et al. 2016) via `pilot.symmetry`. This amounts to choosing the same pilot density for both case and control densities. By choosing `"none"` (default), the result uses the case and control data separately for the fixed-bandwidth pilots, providing the original asymmetric density-ratio of Davies and Hazelton (2010). By selecting either of `"f"`, `"g"`, or `"pooled"`, the pilot density is calculated based on the case, control, or pooled case/control data respectively (using `hp[1]` as the fixed bandwidth). Davies et al. (2016) noted some beneficial practical behaviour of the symmetric adaptive surface over the asymmetric. (The pilot bandwidth(s), if not supplied in `hp`, are calculated internally via default use of `LSCV.density`, using the requested symmetric-based data set, or separately with respect to the case and control datasets `f` and `g` if `pilot.symmetry = "none"`.)

Value

A single numeric value of the estimated bandwidth (if `auto.optim = TRUE`). Otherwise, a list of two numeric vectors of equal length giving the bandwidth sequence (as `hs`) and corresponding CV function value (as `CV`).

Warning

The jointly optimal bandwidth selector can be computationally expensive for large data sets and fine evaluation grid resolutions. The user may need to experiment with adjusting `hlim` to find a suitable minimum.

Author(s)

T. M. Davies

References

Davies, T. M. (2013), Jointly optimal bandwidth selection for the planar kernel-smoothed density-ratio, *Spatial and Spatio-temporal Epidemiology*, **5**, 51-65.

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Jones, K. and Hazelton, M.L. (2016), Symmetric adaptive smoothing regimens for estimation of the spatial relative risk function, *Computational Statistics & Data Analysis*, **101**, 12-28.

Hazelton, M. L. (2008), Letter to the editor: Kernel estimation of risk surfaces without the need for edge correction, *Statistics in Medicine*, **27**, 2269-2272.

Kelsall, J.E. and Diggle, P.J. (1995), Kernel estimation of relative risk, *Bernoulli*, **1**, 3-16.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

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

See Also

bivariate.density

Examples

```
data(pbc)
pbccas <- split(pbc)$case
pbccon <- split(pbc)$control

# FIXED (for common h)

LSCV.risk(pbccas,pbccon)
LSCV.risk(pbccas,pbccon,method="hazelton")
hcv <- LSCV.risk(pbccas,pbccon,method="davies",auto.optim=FALSE)
plot(hcv[,1],log(hcv[,2]));abline(v=hcv[which.min(hcv[,2]),1],col=2,lty=2)


# ADAPTIVE (for common h0)

LSCV.risk(pbccas,pbccon,type="adaptive")

# change pilot bandwidths used
LSCV.risk(pbccas,pbccon,type="adaptive",hp=c(OS(pbccas)/2,OS(pbccon)/2))

# specify pooled-data symmetric relative risk estimator
LSCV.risk(pbccas,pbccon,type="adaptive",hp=OS(pbc),pilot.symmetry="pooled")

# as above, for Hazelton selector
LSCV.risk(pbccas,pbccon,type="adaptive",method="hazelton")
LSCV.risk(pbccas,pbccon,type="adaptive",method="hazelton",hp=c(OS(pbccas)/2,OS(pbccon)/2))
LSCV.risk(pbccas,pbccon,type="adaptive",method="hazelton",hp=OS(pbc),pilot.symmetry="pooled")
```

`multiscale.density` *Multi-scale adaptive kernel density/intensity estimation*

Description

Computes adaptive kernel estimates of spatial density/intensity using a 3D FFT for multiple global bandwidth scales.

Usage

```
multiscale.density(pp, h0, hp = NULL, h0fac = c(0.25, 1.5),
  edge = c("uniform", "none"), resolution = 128, dimz = 64,
  gamma.scale = "geometric", trim = 5, intensity = FALSE,
  pilot.density = NULL, xy = NULL, taper = TRUE, verbose = TRUE)
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data set to be smoothed.
`h0`	Reference global bandwidth for adaptive smoothing; numeric value > 0. Multiscale estimates will be computed by rescaling this value as per `h0fac`.
`hp`	Pilot bandwidth (scalar, numeric > 0) to be used for fixed bandwidth estimation of the pilot density. If `NULL` (default), it will take on the value of `h0`. Ignored when `pilot.density` is supplied as a pre-defined pixel image.
`h0fac`	A numeric vector of length 2 stipulating the span of the global bandwidths in the multiscale estimates. Interpreted as a multiplicative factor on `h0`. See 'Details'.
`edge`	Character string dictating edge correction. `"uniform"` (default) corrects based on evaluation grid coordinate. Setting `edge="none"` requests no edge correction.
`resolution`	Numeric value > 0. Resolution of evaluation grid in the spatial domain; the densities/intensities will be returned on a [`resolution` $\times$ `resolution`] grid.
`dimz`	Resolution of z- (rescaled bandwidth)-axis in the trivariate convolution. Higher values increase precision of the multiscale estimates at a computational cost. See 'Details'.
`gamma.scale`	Scalar, numeric value > 0; controls rescaling of the variable bandwidths. Defaults to the geometric mean of the bandwidth factors given the pilot density (as per Silverman, 1986). See the documentation for `bivariate.density`.
`trim`	Numeric value > 0; controls bandwidth truncation for adaptive estimation. See the documentation for `bivariate.density`.
`intensity`	Logical value indicating whether to return an intensity estimate (integrates to the sample size over the study region), or a density estimate (default, integrates to 1).
`pilot.density`	An optional pixel image (class `im`) giving the pilot density to be used for calculation of the variable bandwidths in adaptive estimation, **or** a `ppp.object` giving the data upon which to base a fixed-bandwidth pilot estimate using `hp`. See the documentation for `bivariate.density`.
`xy`	Optional alternative specification of the spatial evaluation grid; matches the argument of the same tag in `as.mask`. If supplied, `resolution` is ignored.
`taper`	Logical value indicating whether to taper off the trivariate kernel outside the range of `h0*h0fac` in the scale space; see Davies & Baddeley (2018). Keep at the default `TRUE` if you don't know what this means.
`verbose`	Logical value indicating whether to print function progress.

Details

Davies & Baddeley (2018) investigated computational aspects of Abramson's (1982) adaptive kernel smoother for spatial (2D) data. This function is the implementation of the 3D convolution via a fast-Fourier transform (FFT) which allows simultaneous calculation of an adaptive kernel estimate at multiple global bandwidth scales.

These 'multiple global bandwidth scales' are computed with respect to rescaling a reference value of the global bandwidth passed to the `h0` argument. This rescaling is defined by the range provided to the argument `h0fac`. For example, by default, the function will compute the adaptive kernel estimate for a range of global bandwidths between 0.25*`h0` and 1.5*`h0`. The exact numeric limits are subject to discretisation, and so the returned valid range of global bandwidths will differ slightly. The exact resulting range following function execution is returned as the `h0range` element of the result, see 'Value' below.

The distinct values of global bandwidth used (which define the aforementioned `h0range`) and hence the total number of pixel `images` returned depend on both the width of the span `h0fac` and the discretisation applied to the bandwidth axis through `dimz`. Increasing this z-resolution will provide more pixel images and hence greater numeric precision, but increases computational cost. The returned pixel `images` that represent the multiscale estimates are stored in a named list (see 'Value'), whose names reflect the corresponding distinct global bandwidth. See 'Examples' for the easy way to extract these distinct global bandwidths.

The user can request an interpolated density/intensity estimate for any global bandwidth value within `h0range` by using the `multiscale.slice` function, which returns an object of class `bivden`.

Value

An object of class `"msden"`. This is very similar to a `bivden` object, with lists of pixel `im`ages in the `z`, `him`, and `q` components (instead of standalone images).

`z`	A list of the resulting density/intensity estimates; each member being a pixel image object of class `im`. They are placed in increasing order of the discretised values of `h0`.
`h0`	A copy of the reference value of `h0` used.
`h0range`	A vector of length 2 giving the actual range of global bandwidth values available (inclusive).
`hp`	A copy of the value of `hp` used.
`h`	A numeric vector of length equal to the number of data points, giving the bandwidth used for the corresponding observation in `pp` with respect to the reference global bandwidth `h0`.
`him`	A list of pixel images (class `im`), corresponding to `z`, giving the 'hypothetical' Abramson bandwidth at each pixel coordinate conditional upon the observed data and the global bandwidth used.
`q`	Edge-correction weights; list of pixel `im`ages corresponding to `z` if `edge = "uniform"`, and `NULL` if `edge = "none"`.
`gamma`	The numeric value of `gamma.scale` used in scaling the bandwidths.
`geometric`	The geometric mean of the untrimmed variable bandwidth factors. This will be identical to `gamma` if `gamma.scale = "geometric"` as per default.

pp	A copy of the ppp.object initially passed to the pp argument, containing the data that were smoothed.

Author(s)

T.M. Davies and A. Baddeley

References

Abramson, I. (1982). On bandwidth variation in kernel estimates — a square root law, *Annals of Statistics*, **10**(4), 1217-1223.

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

See Also

bivariate.density, multiscale.slice

Examples

```
data(chorley) # Chorley-Ribble data (package 'spatstat')
ch.multi <- multiscale.density(chorley,h0=1)
plot(ch.multi)

ch.pilot <- bivariate.density(chorley,h0=0.75) # with pre-defined pilot density
ch.multi2 <- multiscale.density(chorley,h0=1,pilot.density=ch.pilot$z)
plot(ch.multi2)

data(pbc)
# widen h0 scale, increase z-axis resolution
pbc.multi <- multiscale.density(pbc,h0=2,hp=1,h0fac=c(0.25,2.5),dimz=128)
plot(pbc.multi)
```

multiscale.slice *Slicing a multi-scale density/intensity object*

Description

Takes slices of a multi-scale density/intensity estimate at desired global bandwidths

Usage

```
multiscale.slice(msob, h0, checkargs = TRUE)
```

Arguments

msob	An object of class msden giving the multi-scale estimate from which to take slices.
h0	Desired global bandwidth(s); the density/intensity estimate corresponding to which will be returned. A numeric vector. All values **must** be in the available range provided by msob$h0range; see 'Details'.
checkargs	Logical value indicating whether to check validity of msob and h0. Disable only if you know this check will be unnecessary.

Details

Davies & Baddeley (2018) demonstrate that once a multi-scale density/intensity estimate has been computed, we may take slices parallel to the spatial domain of the trivariate convolution to return the estimate at any desired global bandwidth. This function is the implementation thereof based on a multi-scale estimate resulting from a call to multiscale.density.

The function returns an error if the requested slices at h0 are not all within the available range of pre-computed global bandwidth scalings as defined by the h0range component of msob.

Because the contents of the msob argument, an object of class msden, are returned based on a discretised set of global bandwidth scalings, the function internally computes the desired surface as a pixel-by-pixel linear interpolation using the two discretised global bandwidth rescalings that bound each requested h0. (Thus, numeric accuracy of the slices is improved with an increase to the dimz argument of the preceding call to multiscale.density at the cost of additional computing time.)

Value

If h0 is scalar, an object of class bivden with components corresponding to the requested slice at h0. If h0 is a vector, a list of objects of class bivden.

Author(s)

T.M. Davies

References

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

See Also

multiscale.density, bivariate.density

Examples

```
data(chorley) # Chorley-Ribble data (package 'spatstat')
ch.multi <- multiscale.density(chorley,h0=1,h0fac=c(0.5,2))
```

```
available.h0(ch.multi)
ch.slices <- multiscale.slice(ch.multi,h0=c(0.7,1.1,1.6))

par(mfcol=c(2,3)) # plot each density and edge-correction surface
for(i in 1:3) { plot(ch.slices[[i]]$z); plot(ch.slices[[i]]$q) }
```

NS *Normal scale (NS) bandwidth selector*

Description

Provides the asymptotically optimal fixed bandwidths for spatial or spatiotemporal normal densities based on a simple expression.

Usage

```
NS(pp, nstar = c("npoints", "geometric"), scaler = c("silverman",
  "IQR", "sd", "var"))

NS.spattemp(pp, tt = NULL, nstar = "npoints", scaler = c("silverman",
  "IQR", "sd", "var"))
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data to be smoothed.
`nstar`	Optional. Controls the value to use in place of the number of observations *n* in the normal scale formula. Either a character string, `"npoints"` (default) or `"geometric"` (only possible for `NS`), or a positive numeric value. See 'Details'.
`scaler`	Optional. Controls the value for a scalar representation of the spatial (and temporal for `NS.spattemp`) scale of the data. Either a character string, `"silverman"` (default), `"IQR"`, `"sd"`, or `"var"`; or a positive numeric value. See 'Details'.
`tt`	A numeric vector of equal length to the number of points in `pp`, giving the time corresponding to each spatial observation. If unsupplied, the function attempts to use the values in the `marks` attribute of the `ppp.object` in `pp`.

Details

These functions calculate scalar smoothing bandwidths for kernel density estimates of spatial or spatiotemporal data: the optimal values would minimise the asymptotic mean integrated squared error assuming normally distributed data; see pp. 46-48 of Silverman (1986). The `NS` function returns a single bandwidth for isotropic smoothing of spatial (2D) data. The `NS.spattemp` function returns two values – one for the spatial margin and another for the temporal margin, based on independently applying the normal scale rule (in 2D and 1D) to the spatial and temporal margins of the supplied data.

Effective sample size The formula requires a sample size, and this can be minimally tailored via `nstar`. By default, the function simply uses the number of observations in `pp`: `nstar = "npoints"`. Alternatively, the user can specify their own value by simply supplying a single positive numeric value to `nstar`. For `NS` (not applicable to `NS.spattemp`), if `pp` is a `ppp.object` with factor-valued `marks`, then the user has the option of using `nstar = "geometric"`, which sets the sample size used in the formula to the geometric mean of the counts of observations of each mark. This can be useful for e.g. relative risk calculations, see Davies and Hazelton (2010).

Spatial (and temporal) scale The `scaler` argument is used to specify spatial (as well as temporal, in use of `NS.spattemp`) scale. For isotropic smoothing in the spatial margin, one may use the 'robust' estimate of standard deviation found by a weighted mean of the interquartile ranges of the x- and y-coordinates of the data respectively (`scaler = "IQR"`). Two other options are the raw mean of the coordinate-wise standard deviations (`scaler = "sd"`), or the square root of the mean of the two variances (`scaler = "var"`). A fourth option, `scaler = "silverman"` (default), sets the scaling constant to be the minimum of the `"IQR"` and `"sd"` options; see Silverman (1986), p. 47. In use of `NS.spattemp` the univariate version of the elected scale statistic is applied to the recorded times of the data for the temporal bandwidth. Alternatively, like `nstar`, the user can specify their own value by simply supplying a single positive numeric value to `scaler` for `NS`, or a numeric vector of length 2 (in the order of *[<spatial scale>, <temporal scale>]*) for `NS.spattemp`.

Value

A single numeric value of the estimated spatial bandwidth for `NS`, or a named numeric vector of length 2 giving the spatial bandwidth (as `h`) and the temporal bandwidth (as `lambda`) for `NS.spattemp`.

Warning

The NS bandwidth is an approximation, and assumes *that the target density is normal*. This is considered rare in most real-world applications. Nevertheless, it remains a quick and easy 'rule-of-thumb' method with which one may obtain a smoothing parameter. Note that a similar expression for the adaptive kernel estimator is not possible (Davies et al., 2018).

Author(s)

T.M. Davies

References

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Flynn, C.R. and Hazelton, M.L. (2018), On the utility of asymptotic bandwidth selectors for spatially adaptive kernel density estimation, *Statistics & Probability Letters* [in press].

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

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

Examples

```
data(pbc)

NS(pbc)
NS(pbc,nstar="geometric") # uses case-control marks to replace sample size
NS(pbc,scaler="var") # set different scalar measure of spread

data(burk)
NS.spattemp(burk$cases)
NS.spattemp(burk$cases,scaler="sd")
```

`OS` *Oversmoothing (OS) bandwidth selector*

Description

Provides fixed bandwidths for spatial or spatiotemporal data based on the maximal smoothing (oversmoothing) principle of Terrell (1990).

Usage

```
OS(pp, nstar = c("npoints", "geometric"), scaler = c("silverman",
  "IQR", "sd", "var"))

OS.spattemp(pp, tt = NULL, nstar = "npoints", scaler = c("silverman",
  "IQR", "sd", "var"))
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data to be smoothed.
`nstar`	Optional. Controls the value to use in place of the number of observations n in the oversmoothing formula. Either a character string, `"npoints"` (default) or `"geometric"` (only possible for `OS`), or a positive numeric value. See 'Details'.
`scaler`	Optional. Controls the value for a scalar representation of the spatial (and temporal for `OS.spattemp`) scale of the data. Either a character string, `"silverman"` (default), `"IQR"`, `"sd"`, or `"var"`; or positive numeric value(s). See 'Details'.
`tt`	A numeric vector of equal length to the number of points in `pp`, giving the time corresponding to each spatial observation. If unsupplied, the function attempts to use the values in the `marks` attribute of the `ppp.object` in `pp`.

Details

These functions calculate scalar smoothing bandwidths for kernel density estimates of spatial or spatiotemporal data: the "maximal amount of smoothing compatible with the estimated scale of the observed data". See Terrell (1990). The `OS` function returns a single bandwidth for isotropic smoothing of spatial (2D) data. The `OS.spattemp` function returns two values – one for the spatial margin and another for the temporal margin, based on independently applying Terrell's (1990) rule (in 2D and 1D) to the spatial and temporal margins of the supplied data.

Effective sample size The formula requires a sample size, and this can be minimally tailored via `nstar`. By default, the function simply uses the number of observations in `pp`: `nstar = "npoints"`. Alternatively, the user can specify their own value by simply supplying a single positive numeric value to `nstar`. For `OS` (not applicable to `OS.spattemp`), if `pp` is a `ppp.object` with factor-valued `marks`, then the user has the option of using `nstar = "geometric"`, which sets the sample size used in the formula to the geometric mean of the counts of observations of each mark. This can be useful for e.g. relative risk calculations, see Davies and Hazelton (2010).

Spatial (and temporal) scale The `scaler` argument is used to specify spatial (as well as temporal, in use of `OS.spattemp`) scale. For isotropic smoothing in the spatial margin, one may use the 'robust' estimate of standard deviation found by a weighted mean of the interquartile ranges of the x- and y-coordinates of the data respectively (`scaler = "IQR"`). Two other options are the raw mean of the coordinate-wise standard deviations (`scaler = "sd"`), or the square root of the mean of the two variances (`scaler = "var"`). A fourth option, `scaler = "silverman"` (default), sets the scaling constant to be the minimum of the `"IQR"` and `"sd"` options; see Silverman (1986), p. 47. In use of `OS.spattemp` the univariate version of the elected scale statistic is applied to the recorded times of the data for the temporal bandwidth. Alternatively, like `nstar`, the user can specify their own value by simply supplying a single positive numeric value to `scaler` for `OS`, or a numeric vector of length 2 (in the order of *[<spatial scale>, <temporal scale>]*) for `OS.spattemp`.

Value

A single numeric value of the estimated spatial bandwidth for `OS`, or a named numeric vector of length 2 giving the spatial bandwidth (as `h`) and the temporal bandwidth (as `lambda`) for `OS.spattemp`.

Author(s)

T.M. Davies

References

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Terrell, G.R. (1990), The maximal smoothing principle in density estimation, *Journal of the American Statistical Association*, **85**, 470-477.

Examples

```
data(pbc)
```

```
OS(pbc)
OS(pbc,nstar="geometric") # uses case-control marks to replace sample size
OS(pbc,scaler="var") # set different scalar measure of spread

data(burk)
OS.spattemp(burk$cases)
OS.spattemp(burk$cases,scaler="sd")
```

---

pbc *Primary biliary cirrhosis data*

Description

Data of the locations of 761 cases of primary biliary cirrhosis in several adjacent health regions of north-eastern England, along with 3020 controls representing the at-risk population, collected between 1987 and 1994. These data were first presented and analysed by Prince et al. (2001); subsequent analysis of these data in the spirit of `sparr` was performed in Davies and Hazelton (2010). Also included is the polygonal study region.

Format

`pbc` is a dichotomously marked `ppp.object`, with locations expressed in UK Ordnance Survey Coordinates (km).

Acknowledgements

The authors thank Prof. Peter Diggle at Lancaster University (`http://www.lancs.ac.uk/staff/diggle/`) for providing access to these data.

Source

Prince et al. (2001), The geographical distribution of primary biliary cirrhosis in a well-defined cohort, *Hepatology*, **34**, 1083-1088.

References

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Examples

```
data(pbc)
summary(pbc)
plot(pbc)
```

`plot.bivden` *Plotting sparr objects*

Description

`plot` methods for classes `bivden`, `stden`, `rrs`, `rrst` and `msden`.

Usage

```
## S3 method for class 'bivden'
plot(x, what = c("z", "edge", "bw"), add.pts = FALSE,
  auto.axes = TRUE, override.par = TRUE, ...)

## S3 method for class 'msden'
plot(x, what = c("z", "edge", "bw"), sleep = 0.2,
  override.par = TRUE, ...)

## S3 method for class 'rrs'
plot(x, auto.axes = TRUE, tol.show = TRUE,
  tol.type = c("upper", "lower", "two.sided"), tol.args = list(levels =
  0.05, lty = 1, drawlabels = TRUE), ...)

## S3 method for class 'rrst'
plot(x, tselect = NULL, type = c("joint",
  "conditional"), fix.range = FALSE, tol.show = TRUE,
  tol.type = c("upper", "lower", "two.sided"), tol.args = list(levels =
  0.05, lty = 1, drawlabels = TRUE), sleep = 0.2, override.par = TRUE,
  expscale = FALSE, ...)

## S3 method for class 'stden'
plot(x, tselect = NULL, type = c("joint",
  "conditional"), fix.range = FALSE, sleep = 0.2,
  override.par = TRUE, ...)
```

Arguments

`x`	An object of class `bivden`, `stden`, `rrs`, `rrst`, or `msden`.
`what`	A character string to select plotting of result (`"z"`; default); edge-correction surface (`"edge"`); or variable bandwidth surface (`"bw"`).
`add.pts`	Logical value indicating whether to add the observations to the image plot using default `points`.
`auto.axes`	Logical value indicating whether to display the plot with automatically added x-y axes and an ‘L’ box in default styles.
`override.par`	Logical value indicating whether to override the existing graphics device parameters prior to plotting, resetting `mfrow` and `mar`. See ‘Details’ for when you might want to disable this.

`...`	Additional graphical parameters to be passed to `plot.im`, or in one instance, to `plot.ppp` (see 'Details').
`sleep`	Single positive numeric value giving the amount of time (in seconds) to `Sys.sleep` before drawing the next image in the animation.
`tol.show`	Logical value indicating whether to show pre-computed tolerance contours on the plot(s). The object `x` must already have the relevant p-value surface(s) stored in order for this argument to have any effect.
`tol.type`	A character string used to control the type of tolerance contour displayed; a test for elevated risk (`"upper"`), decreased risk (`"lower"`), or a two-tailed test (`two.sided`).
`tol.args`	A named list of valid arguments to be passed directly to `contour` to control the appearance of plotted contours. Commonly used items are `levels`, `lty`, `lwd` and `drawlabels`.
`tselect`	Either a single numeric value giving the time at which to return the plot, or a vector of length 2 giving an interval of times over which to plot. This argument must respect the stored temporal bound in `x$tlim`, else an error will be thrown. By default, the full set of images (i.e. over the entire available time span) is plotted.
`type`	A character string to select plotting of joint/unconditional spatiotemporal estimate (default) or conditional spatial density given time.
`fix.range`	Logical value indicating whether use the same color scale limits for each plot in the sequence. Ignored if the user supplies a pre-defined `colourmap` to the `col` argument, which is matched to `...` above and passed to `plot.im`. See 'Examples'.
`expscale`	Logical value indicating whether to force a raw-risk scale. Useful for users wishing to plot a log-relative risk surface, but to have the raw-risk displayed on the colour ribbon.

Details

In all instances, visualisation is deferred to `plot.im`, for which there are a variety of customisations available the user can access via `...`. The one exception is when plotting observation-specific `"diggle"` edge correction factors—in this instance, a plot of the spatial observations is returned with size proportional to the influence of each correction weight.

When plotting a `rrs` object, a pre-computed p-value surface (see argument `tolerate` in `risk`) will automatically be superimposed at a significance level of 0.05. Greater flexibility in visualisation is gained by using `tolerance` in conjunction with `contour`.

An `msden`, `stden`, or `rrst` object is plotted as an animation, one pixel image after another, separated by `sleep` seconds. If instead you intend the individual images to be plotted in an array of images, you should first set up your plot device layout, and ensure `override.par = FALSE` so that the function does not reset these device parameters itself. In such an instance, one might also want to set `sleep = 0`.

Value

Plots to the relevant graphics device.

Author(s)

T.M. Davies

Examples

```
data(pbc)
data(fmd)
data(burk)

# 'bivden' object
pbcden <- bivariate.density(split(pbc)$case,h0=3,hp=2,adapt=TRUE,davies.baddeley=0.05,verbose=FALSE)
plot(pbcden)
plot(pbcden,what="bw",main="PBC cases\n variable bandwidth surface",xlab="Easting",ylab="Northing")

# 'stden' object
burkden <- spattemp.density(burk$cases,tres=128) # observation times are stored in marks(burk$cases)
plot(burkden,fix.range=TRUE,sleep=0.1) # animation
plot(burkden,tselect=c(1000,3000),type="conditional") # spatial densities conditional on each time

# 'rrs' object
pbcrr <- risk(pbc,h0=4,hp=3,adapt=TRUE,tolerate=TRUE,davies.baddeley=0.025,edge="diggle")
plot(pbcrr) # default
plot(pbcrr,tol.args=list(levels=c(0.05,0.01),lty=2:1,col="seagreen4"),auto.axes=FALSE)

# 'rrst' object
f <- spattemp.density(fmd$cases,h=6,lambda=8)
g <- bivariate.density(fmd$controls,h0=6)
fmdrr <- spattemp.risk(f,g,tolerate=TRUE)
plot(fmdrr,sleep=0.1,fix.range=TRUE)
plot(fmdrr,type="conditional",sleep=0.1,tol.type="two.sided",
     tol.args=list(levels=0.05,drawlabels=FALSE))

# 'msden' object
pbcmult <- multiscale.density(split(pbc)$case,h0=4,h0fac=c(0.25,2.5))
plot(pbcmult) # densities
plot(pbcmult,what="edge") # edge correction surfaces
plot(pbcmult,what="bw") # bandwidth surfaces
```

`print.bivden` *Printing sparr objects*

Description

`print` methods for classes `bivden`, `stden`, `rrs`, `rrst` and `msden`.

Usage

```
## S3 method for class 'bivden'
print(x, ...)

## S3 method for class 'msden'
print(x, ...)

## S3 method for class 'rrs'
print(x, ...)

## S3 method for class 'rrst'
print(x, ...)

## S3 method for class 'stden'
print(x, ...)
```

Arguments

`x`	An object of class `bivden`, `stden`, `rrs`, `rrst`, or `msden`.
`...`	Ignored.

Author(s)

T.M. Davies

`rimpoly` *Random point generation inside polygon*

Description

Generates a random point pattern of n iid points with any specified distribution based on a pixel image and a corresponding polygonal window.

Usage

```
rimpoly(n, z, w = NULL, correction = 1.1, maxpass = 50)
```

Arguments

`n`	Number of points to generate.
`z`	A pixel image of class `im` defining the probability density of the points, possibly unnormalised.
`w`	A polygonal window of class `owin`. See ‘Details’.
`correction`	An adjustment to the number of points generated at the initial pass of the internal loop in an effort to minimise the total number of passes required to reach n points. See ‘Details’.

`maxpass`	The maximum number of passes allowed before the function exits. If this is reached before n points are found that fall within `w`, a warning is issued.

Details

This function is a deliberate variant of `rpoint` (Baddeley et. al, 2015), to be accessed when the user desires a randomly generated point pattern based on a pixel image, but wants the window of the point pattern to be a corresponding irregular polygon, as opposed to a binary image mask (which, when converted to a polygon directly, gives jagged edges based on the union of the pixels). When the user specifies their own polygonal window, a `while` loop is called and repeated as many times as necessary (up to `maxpass` times) to find `n` points inside `w` (when `w = NULL`, then the aforementioned union of the pixels of `z` is used, obtained via `as.polygonal(Window(z))`). The loop is necessary because the standard behaviour of `rpoint` can (and often does) yield points that sit in corners of pixels which lie outside the corresponding `w`.

The `correction` argument is used to determine how many points are generated initially, which will be `ceiling(correction*n)`; to minimise the number of required passes over the loop this is by default set to give a number slightly higher than the requested `n`.

An error is thrown if `Window(z)` and `w` do not overlap.

Value

An object of class `ppp` containing the `n` generated points, defined with the polygonal `owin`, `w`.

Author(s)

T.M. Davies

References

Baddeley, A., Rubak, E. and Turner, R. (2015) *Spatial Point Patterns: Methodology and Applications with R*, Chapman and Hall/CRC Press, UK.

Examples

```
data(pbc)
Y <- bivariate.density(pbc,h0=2.5,res=25)

# Direct use of 'rpoint':
A <- rpoint(500,Y$z)
npoints(A)

# Using 'rimpoly' without supplying polygon:
B <- rimpoly(500,Y$z)
npoints(B)

# Using 'rimpoly' with the original pbc polygonal window:
C <- rimpoly(500,Y$z,Window(Y$pp))
npoints(C)
```

```
par(mfrow=c(1,3))
plot(A,main="rpoint")
plot(B,main="rimpoly (no polygon supplied)")
plot(C,main="rimpoly (original polygon supplied)")
```

`risk` *Spatial relative risk/density ratio*

Description

Estimates a *relative risk* function based on the ratio of two 2D kernel density estimates.

Usage

```
risk(f, g = NULL, log = TRUE, h0 = NULL, hp = h0, adapt = FALSE,
  tolerate = FALSE, doplot = FALSE, pilot.symmetry = c("none", "f",
  "g", "pooled"), epsilon = 0, verbose = TRUE, ...)
```

Arguments

`f`	Either a pre-calculated object of class `bivden` representing the 'case' (numerator) density estimate, or an object of class `ppp` giving the observed case data. Alternatively, if `f` is `ppp` object with dichotomous factor-valued `marks`, the function treats the first level as the case data, and the second as the control data, obviating the need to supply `g`.
`g`	As for `f`, for the 'control' (denominator) density; this object must be of the same class as `f`. Ignored if, as stated above, `f` contains both case and control observations.
`log`	Logical value indicating whether to return the (natural) log-transformed relative risk function as recommended by Kelsall and Diggle (1995a). Defaults to `TRUE`, with the alternative being the raw density ratio.
`h0`	A single positive numeric value or a vector of length 2 giving the global bandwidth(s) to be used for case/control density estimates; defaulting to a common oversmoothing bandwidth computed via `OS` on the pooled data using `nstar = "geometric"` if unsupplied. Ignored if `f` and `g` are already `bivden` objects.
`hp`	A single numeric value or a vector of length 2 giving the pilot bandwidth(s) to be used for fixed-bandwidth estimation of the pilot densities for adaptive risk surfaces. Ignored if `adapt = FALSE` or if `f` and `g` are already `bivden` objects.
`adapt`	A logical value indicating whether to employ adaptive smoothing for internally estimating the densities. Ignored if `f` and `g` are already `bivden` objects.
`tolerate`	A logical value indicating whether to internally calculate a corresponding asymptotic p-value surface (for tolerance contours) for the estimated relative risk function. See 'Details'.

`doplot`	Logical. If `TRUE`, an image plot of the estimated relative risk function is produced using various visual presets. If additionally `tolerate` was `TRUE`, asymptotic tolerance contours are automatically added to the plot at a significance level of 0.05 for elevated risk (for more flexible options for calculating and plotting tolerance contours, see `tolerance` and `tol.contour`).
`pilot.symmetry`	A character string used to control the type of symmetry, if any, to use for the bandwidth factors when computing an adaptive relative risk surface. See 'Details'. Ignored if `adapt = FALSE`.
`epsilon`	A single non-negative numeric value used for optional scaling to produce additive constant to each density in the raw ratio (see 'Details'). A zero value requests no additive constant (default).
`verbose`	Logical value indicating whether to print function progress during execution.
`...`	Additional arguments passed to any internal calls of `bivariate.density` for estimation of the requisite densities. Ignored if `f` and `g` are already `bivden` objects.

Details

The relative risk function is defined here as the ratio of the 'case' density to the 'control' (Bithell, 1990; 1991). Using kernel density estimation to model these densities (Diggle, 1985), we obtain a workable estimate thereof. This function defines the risk function *r* in the following fashion:

r `= (fd + epsilon*max(gd))/(gd + epsilon*max(gd)),`

where `fd` and `gd` denote the case and control density estimates respectively. Note the (optional) additive constants defined by `epsilon` times the maximum of each of the densities in the numerator and denominator respectively (see Bowman and Azzalini, 1997).

The log-risk function *rho*, given by *rho* = log[*r*], is argued to be preferable in practice as it imparts a sense of symmetry in the way the case and control densities are treated (Kelsall and Diggle, 1995a;b). The option of log-transforming the returned risk function is therefore selected by default.

When computing adaptive relative risk functions, the user has the option of obtaining a so-called *symmetric* estimate (Davies et al. 2016) via `pilot.symmetry`. This amounts to choosing the same pilot density for both case and control densities. By choosing `"none"` (default), the result uses the case and control data separately for the fixed-bandwidth pilots, providing the original asymmetric density-ratio of Davies and Hazelton (2010). By selecting either of `"f"`, `"g"`, or `"pooled"`, the pilot density is calculated based on the case, control, or pooled case/control data respectively (using `hp[1]` as the fixed bandwidth). Davies et al. (2016) noted some beneficial practical behaviour of the symmetric adaptive surface over the asymmetric.

If the user selects `tolerate = TRUE`, the function internally computes asymptotic tolerance contours as per Hazelton and Davies (2009) and Davies and Hazelton (2010). When `adapt = FALSE`, the reference density estimate (argument `ref.density` in `tolerance`) is taken to be the estimated control density. The returned pixel `im`age of *p*-values (see 'Value') is interpreted as an upper-tailed test i.e. smaller *p*-values represent greater evidence in favour of significantly increased risk. For greater control over calculation of tolerance contours, use `tolerance`.

Value

An object of class "rrs". This is a named list with the following components:

rr	A pixel image of the estimated risk surface.
f	An object of class bivden used as the numerator or 'case' density estimate.
g	An object of class bivden used as the denominator or 'control' density estimate.
P	Only included if `tolerate = TRUE`. A pixel image of the *p*-value surface for tolerance contours; `NULL` otherwise.

Author(s)

T.M. Davies

References

Bithell, J.F. (1990), An application of density estimation to geographical epidemiology, *Statistics in Medicine*, **9**, 691-701.

Bithell, J.F. (1991), Estimation of relative risk functions, *Statistics in Medicine*, **10**, 1745-1751.

Bowman, A.W. and Azzalini A. (1997), *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations*, Oxford University Press Inc., New York.

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Jones, K. and Hazelton, M.L. (2016), Symmetric adaptive smoothing regimens for estimation of the spatial relative risk function, *Computational Statistics & Data Analysis*, **101**, 12-28.

Diggle, P.J. (1985), A kernel method for smoothing point process data, *Journal of the Royal Statistical Society Series C*, **34**(2), 138-147.

Hazelton, M.L. and Davies, T.M. (2009), Inference based on kernel estimates of the relative risk function in geographical epidemiology, *Biometrical Journal*, **51**(1), 98-109.

Kelsall, J.E. and Diggle, P.J. (1995a), Kernel estimation of relative risk, *Bernoulli*, **1**, 3-16.

Kelsall, J.E. and Diggle, P.J. (1995b), Non-parametric estimation of spatial variation in relative risk, *Statistics in Medicine*, **14**, 2335-2342.

Examples

```
data(pbc)
pbccas <- split(pbc)$case
pbccon <- split(pbc)$control
h0 <- OS(pbc,nstar="geometric")

# Fixed
pbcrr1 <- risk(pbccas,pbccon,h0=h0,tolerate=TRUE)

# Asymmetric adaptive
pbcrr2 <- risk(pbccas,pbccon,h0=h0,adapt=TRUE,hp=c(OS(pbccas)/2,OS(pbccon)/2),
               tolerate=TRUE,davies.baddeley=0.05)

# Symmetric (pooled) adaptive
pbcrr3 <- risk(pbccas,pbccon,h0=h0,adapt=TRUE,tolerate=TRUE,hp=OS(pbc)/2,
               pilot.symmetry="pooled",davies.baddeley=0.05)

# Symmetric (case) adaptive; from two existing 'bivden' objects
f <- bivariate.density(pbccas,h0=h0,hp=2,adapt=TRUE,pilot.density=pbccas,
                       edge="diggle",davies.baddeley=0.05,verbose=FALSE)
g <- bivariate.density(pbccon,h0=h0,hp=2,adapt=TRUE,pilot.density=pbccas,
                       edge="diggle",davies.baddeley=0.05,verbose=FALSE)
pbcrr4 <- risk(f,g,tolerate=TRUE,verbose=FALSE)

par(mfrow=c(2,2))
plot(pbcrr1,override.par=FALSE,main="Fixed")
plot(pbcrr2,override.par=FALSE,main="Asymmetric adaptive")
plot(pbcrr3,override.par=FALSE,main="Symmetric (pooled) adaptive")
plot(pbcrr4,override.par=FALSE,main="Symmetric (case) adaptive")
```

SLIK.adapt *Simultaneous global/pilot likelihood bandwidth selection*

Description

Isotropic global and pilot bandwidth selection for adaptive density/intensity based on likelihood cross-validation.

Usage

```
SLIK.adapt(pp, hold = TRUE, start = rep(OS(pp), 2), hlim = NULL,
  edge = TRUE, zero.action = c(-1, 0), optim.control = list(),
  parallelise = NULL, verbose = TRUE, ...)
```

Arguments

`pp`	An object of class `ppp` giving the observed 2D data to be smoothed.
`hold`	Logical value indicating whether to hold the global and pilot bandwidths equal throughout the optimisation; defaults to `TRUE`. See 'Details'.
`start`	A positively-valued numeric vector of length 2 giving the starting values to be used for the global/pilot optimisation routine when `hold = FALSE`. Defaults to the oversmoothing bandwidth (`OS`) for both values; ignored when `hold = TRUE`.
`hlim`	An optional vector of length 2 giving the limits of the optimisation routine with respect to the bandwidth when `hold = TRUE`. If unspecified, the function attempts to choose this automatically. Ignored when `hold = FALSE`.
`edge`	Logical value indicating whether to edge-correct the density estimates used.

`zero.action`	A numeric vector of length 2, each value being either `-1`, `0` (default), `1` or `2` controlling how the function should behave in response to numerical errors at very small bandwidths, when such a bandwidth results in one or more zero or negative density values during the leave-one-out computations. See 'Details'.
`optim.control`	An optional list to be passed to the `control` argument of `optim` for further control over the numeric optimisation when `hold = FALSE`. See the documentation for `optim` for further details.
`parallelise`	Numeric argument to invoke parallel processing in the brute force leave-one-out calculations, giving the number of CPU cores to use. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you. If `NA` (default), no parallelisation performed and a single loop is used.
`verbose`	Logical value indicating whether to provide function progress commentary.
`...`	Additional arguments controlling density estimation for the internal calculations. Relevant arguments are `resolution`, `gamma.scale`, and `trim`. If unsupplied these default to `64`, `"geometric"`, and `5` respectively; see `bivariate.density` for a further explanation of these arguments.

Details

This function is a generalisation of `LIK.density`, and is used in attempts to simultaneously choose an optimal global and pilot bandwidth for adaptive kernel density estimates. Where `LIK.density` for adaptive estimates assumes the pilot density is held constant (and is not subject to the leave-one-out operations), this function allows the pilot bandwidth to vary alongside the global.

Thus, in contrast to `LIK.density` the internal leave-one-out operations now also affect the pilot estimation stage. Hence, the set of variable bandwidths changes as each point is left out. In turn, this means the leave-one-out operations must be computed by brute force, and this is computationally expensive.

Identifiability problems can sometimes arise when the global and pilot bandwidths are allowed to 'float freely' in the bivariate optimisation routine, which is the default behaviour of the function (with `hold = FALSE`). This can be curbed by setting `hold = TRUE`, which forces both the global and pilot to be held at the same value during optimisation. Doing this also has the beneficial side effect of turning the problem into one of univariate optimisation, thereby reducing total computational cost. Current work (Davies & Lawson, 2018) provides some empirical evidence that this strategy performs quite well in practice.

Like `LSCV.density` and `LIK.density`, the argument `zero.action` can be used to control the level of severity in response to small bandwidths that result (due to numerical error) in at least one density value being zero or less. When this argument is passed a vector of length 2, the first entry corresponds to the global bandwidth (and hence refers to checks of the final adaptive density estimate and its leave-one-out values) and the second to the pilot bandwidth (and hence checks the fixed-bandwidth pilot density and its leave-one-out values). Alternatively a single value may be supplied, which will be taken to be the same for both global and pilot. See the help page for `LIK.density` for an explanation of the four allowable values (`-1`, `0`, `1`, `2`) for each component of this argument.

Value

A numeric vector of length 2 giving the likelihood-maximised global and pilot bandwidths.

Note

While theoretically valid, this is a largely experimental function. There is presently little in the literature to suggest how well this type of simultaneous global/pilot bandwidth selection might perform in practice. Current research efforts (Davies & Lawson, 2018) seek in part to address these questions.

Author(s)

T. M. Davies

References

Davies, T.M. and Lawson, A.B. (2018), An evaluation of likelihood-based bandwidth selectors for spatial and spatiotemporal kernel estimates, *Submitted for publication*.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

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

See Also

Functions for bandwidth selection in package `spatstat`: `bw.diggle`; `bw.ppl`; `bw.scott`; `bw.frac`.

Examples

```
data(pbc)
pbccas <- split(pbc)$case

SLIK.adapt(pbccas)
SLIK.adapt(pbccas,hold=TRUE)
```

`spattemp.density` *Spatiotemporal kernel density estimation*

Description

Provides a fixed-bandwidth kernel estimate of continuous spatiotemporal data.

Usage

```
spattemp.density(pp, h = NULL, tt = NULL, lambda = NULL,
  tlim = NULL, sedge = c("uniform", "none"), tedge = sedge,
  sres = 128, tres = NULL, verbose = TRUE)
```

Arguments

pp	An object of class ppp giving the spatial coordinates of the observations to be smoothed. Possibly marked with the time of each event; see argument tt.
h	Fixed bandwidth to smooth the spatial margin. A numeric value > 0. If unsupplied, the oversmoothing bandwidth is used as per OS.
tt	A numeric vector of equal length to the number of points in pp, giving the time corresponding to each spatial observation. If unsupplied, the function attempts to use the values in the marks attribute of the ppp.object in pp.
lambda	Fixed bandwidth to smooth the temporal margin; a numeric value > 0. If unsupplied, the function internally computes the Sheather-Jones bandwith using bw.SJ (Sheather & Jones, 1991).
tlim	A numeric vector of length 2 giving the limits of the temporal domain over which to smooth. If supplied, all times in tt must fall within this interval (equality with limits allowed). If unsupplied, the function simply uses the range of the observed temporal values.
sedge	Character string dictating spatial edge correction. "uniform" (default) corrects based on evaluation grid coordinate. Setting sedge="none" requests no edge correction.
tedge	As sedge, for temporal edge correction.
sres	Numeric value > 0. Resolution of the [sres $\times$ sres] evaluation grid in the spatial margin.
tres	Numeric value > 0. Resolution of the evaluation points in the temporal margin as defined by the tlim interval. If unsupplied, the density is evaluated at integer values between tlim[1] and tlim[2].
verbose	Logical value indicating whether to print a function progress bar to the console during evaluation.

Details

This function produces a fixed-bandwidth kernel estimate of a single spatiotemporal density, with isotropic smoothing in the spatial margin, as per Fernando & Hazelton (2014). Estimates may be edge-corrected for an irregular spatial study window *and* for the bounds on the temporal margin as per tlim; this edge-correction is performed in precisely the same way as the "uniform" option in bivariate.density.

Specifically, for n trivariate points in space-time (pp, tt, tlim), we have

$$\hat{f}(x,t) = n^{-1}\sum_{i=1}^{n} h^{-2}\lambda^{-1}K((x-x_i)/h)L((t-t_i)/\lambda)/(q(x)q(t)),$$

where $x \in W \subset R^2$ and $t \in T \subset R$; K and L are the 2D and 1D Gaussian kernels controlled by fixed bandwidths h (h) and λ (lambda) respectively; and $q(x) = \int_W h^{-2}K((u-x)/h)du$ and $q(t) = \int_T \lambda^{-1}L((w-t)/\lambda)dw$ are optional edge-correction factors (sedge and tedge).

The above equation provides the *joint* or *unconditional* density at a given space-time location (x,t). In addition to this, the function also yields the *conditional* density at each grid time, defined as

$$\hat{f}(x|t) = \hat{f}(x,t)/\hat{f}(t),$$

where $\hat{f}(t) = n^{-1}\sum_{i=1}^{n}\lambda^{-1}L((t-t_i)/\lambda)/q(t)$ is the univariate kernel estimate of the temporal margin. Normalisation of the two versions $\hat{f}(x,t)$ and $\hat{f}(x|t)$ is the only way they differ. Where in the unconditional setting we have $\int_W\int_T \hat{f}(x,t)dtdx = 1$, in the conditional setting we have $\int_W \hat{f}(x|t)dx = 1$ for all t. See Fernando & Hazelton (2014) for further details and practical reasons as to why we might prefer one over the other in certain situations.

The objects returned by this function (see 'Value' below) are necessary for kernel estimation of spatiotemporal relative risk surfaces, which is performed by `spattemp.risk`.

Value

An object of class `"stden"`. This is effectively a list with the following components:

`z`	A named (by time-point) list of pixel `im`ages corresponding to the joint spatiotemporal density over space at each discretised time.
`z.cond`	A named (by time-point) list of pixel `im`ages corresponding to the conditional spatial density given each discretised time.
`h`	The scalar bandwidth used for spatial smoothing.
`lambda`	The scalar bandwidth used for temporal smoothing.
`tlim`	A numeric vector of length two giving the temporal bound of the density estimate.
`spatial.z`	A pixel `im`age giving the overall spatial margin as a single 2D density estimate (i.e. ignoring time).
`temporal.z`	An object of class `density` giving the overall temporal margin as a single 1D density estimate (i.e. ignoring space).
`qs`	A pixel `im`age giving the edge-correction surface for the spatial margin. `NULL` if `sedge = "none"`.
`qt`	A numeric vector giving the edge-correction weights for the temporal margin. `NULL` if `tedge = "none"`.
`pp`	A `ppp.object` of the spatial data passed to the argument of the same name in the initial function call, with `marks` of the observation times.
`tgrid`	A numeric vector giving the discretised time grid at which the spatiotemporal density was evaluated (matches the names of `z` and `z.cond`).

Author(s)

T.M. Davies

References

Duong, T. (2007), ks: Kernel Density Estimation and Kernel Discriminant Analysis for Multivariate Data in R, *Journal of Statistical Software*, **21**(7), 1-16.

Fernando, W.T.P.S. and Hazelton, M.L. (2014), Generalizing the spatial relative risk function, *Spatial and Spatio-temporal Epidemiology*, **8**, 1-10.

Kelsall, J.E. and Diggle, P.J. (1995), Kernel estimation of relative risk, *Bernoulli*, **1**, 3-16.

Sheather, S. J. and 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.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, New York.

See Also

bivariate.density, spattemp.risk, spattemp.slice

Examples

```
data(burk)
burkcas <- burk$cases

burkden1 <- spattemp.density(burkcas,tres=128)
summary(burkden1)


hlam <- LIK.spattemp(burkcas,tlim=c(400,5900),verbose=FALSE)
burkden2 <- spattemp.density(burkcas,h=hlam[1],lambda=hlam[2],tlim=c(400,5900),tres=256)
tims <- c(1000,2000,3500)
par(mfcol=c(2,3))
for(i in tims){
  plot(burkden2,i,override.par=FALSE,fix.range=TRUE,main=paste("joint",i))
  plot(burkden2,i,"conditional",override.par=FALSE,main=paste("cond.",i))
}
```

`spattemp.risk` *Spatiotemporal relative risk/density ratio*

Description

Produces a spatiotemporal relative risk surface based on the ratio of two kernel estimates of spatiotemporal densities.

Usage

```
spattemp.risk(f, g, log = TRUE, tolerate = FALSE, finiteness = TRUE, verbose = TRUE)
```

Arguments

f	An object of class stden representing the 'case' (numerator) density estimate.
g	Either an object of class stden, or an object of class bivden for the 'control' (denominator) density estimate. This object **must** match the spatial (and temporal, if stden) domain of f completely; see 'Details'.

`log`	Logical value indicating whether to return the log relative risk (default) or the raw ratio.
`tolerate`	Logical value indicating whether to compute and return asymptotic p-value surfaces for elevated risk; see 'Details'.
`finiteness`	Logical value indicating whether to internally correct infinite risk (on the log-scale) to the nearest finite value to avoid numerical problems. A small extra computational cost is required.
`verbose`	Logical value indicating whether to print function progress during execution.

Details

Fernando & Hazelton (2014) generalise the spatial relative risk function (e.g. Kelsall & Diggle, 1995) to the spatiotemporal domain. This is the implementation of their work, yielding the generalised log-relative risk function for $x \in W \subset R^2$ and $t \in T \subset R$. It produces

$$\hat{\rho}(x,t) = \log(\hat{f}(x,t)) - \log(\hat{g}(x,t)),$$

where $\hat{f}(x,t)$ is a fixed-bandwidth kernel estimate of the spatiotemporal density of the cases (argument `f`) and $\hat{g}(x,t)$ is the same for the controls (argument `g`).

- When argument `g` is an object of class `stden` arising from a call to `spattemp.density`, the resolution, spatial domain, and temporal domain of this spatiotemporal estimate must match that of `f` exactly, else an error will be thrown.
- When argument `g` is an object of class `bivden` arising from a call to `bivariate.density`, it is assumed the 'at-risk' control density is static over time. In this instance, the above equation for the relative risk becomes $\hat{\rho} = \log(\hat{f}(x,t)) + \log|T| - \log(g(x))$. The spatial density estimate in `g` must match the spatial domain of `f` exactly, else an error will be thrown.
- The estimate $\hat{\rho}(x,t)$ represents the joint or unconditional spatiotemporal relative risk over $W \times T$. This means that the raw relative risk $\hat{r}(x,t) = \exp \hat{\rho}(x,t)$ integrates to 1 with respect to the control density over space and time: $\int_W \int_T r(x,t)g(x,t)dtdx = 1$. This function also computes the **conditional** spatiotemporal relative risk at each time point, namely

  $$\hat{\rho}(x|t) = \log \hat{f}(x|t) - \log \hat{g}(x|t),$$

  where $\hat{f}(x|t)$ and $\hat{g}(x|t)$ are the conditional densities over space of the cases and controls given a specific time point t (see the documentation for `spattemp.density`). In terms of normalisation, we therefore have $\int_W r(x|t)g(x|t)dx = 1$. In the case where $\hat{g}$ is static over time, one may simply replace $\hat{g}(x|t)$ with $\hat{g}(x)$ in the above.
- Based on the asymptotic properties of the estimator, Fernando & Hazelton (2014) also define the calculation of tolerance contours for detecting statistically significant fluctuations in such spatiotemporal log-relative risk surfaces. This function can produce the required p-value surfaces by setting `tolerate = TRUE`; and if so, results are returned for both the unconditional (x,t) and conditional (x|t) surfaces. See the examples in the documentation for `plot.rrst` for details on how one may superimpose contours at specific p-values for given evaluation times t on a plot of relative risk on the spatial margin.

Value

An object of class "`rrst`". This is effectively a list with the following members:

`rr`	A named (by time-point) list of pixel `im`ages corresponding to the joint spatiotemporal relative risk over space at each discretised time.
`rr.cond`	A named list of pixel `im`ages corresponding to the conditional spatial relative risk given each discretised time.
`P`	A named list of pixel `im`ages of the p-value surfaces testing for elevated risk for the joint estimate. If `tolerate = FALSE`, this will be `NULL`.
`P.cond`	As above, for the conditional relative risk surfaces.
`f`	A copy of the object `f` used in the initial call.
`g`	As above, for `g`.
`tlim`	A numeric vector of length two giving the temporal bound of the density estimate.

Author(s)

T.M. Davies

References

Fernando, W.T.P.S. and Hazelton, M.L. (2014), Generalizing the spatial relative risk function, *Spatial and Spatio-temporal Epidemiology*, **8**, 1-10.

See Also

`spattemp.density`, `spattemp.slice`, `bivariate.density`

Examples

```
data(fmd)
fmdcas <- fmd$cases
fmdcon <- fmd$controls

f <- spattemp.density(fmdcas,h=6,lambda=8) # stden object as time-varying case density
g <- bivariate.density(fmdcon,h0=6) # bivden object as time-static control density
rho <- spattemp.risk(f,g,tolerate=TRUE)
print(rho)

par(mfrow=c(2,3))
plot(rho$f$spatial.z,main="Spatial margin (cases)") # spatial margin of cases
plot(rho$f$temporal.z,main="Temporal margin (cases)") # temporal margin of cases
plot(rho$g$z,main="Spatial margin (controls)") # spatial margin of controls
plot(rho,tselect=50,type="conditional",tol.args=list(levels=c(0.05,0.0001),
     lty=2:1,lwd=1:2),override.par=FALSE)
plot(rho,tselect=100,type="conditional",tol.args=list(levels=c(0.05,0.0001),
     lty=2:1,lwd=1:2),override.par=FALSE)
plot(rho,tselect=200,type="conditional",tol.args=list(levels=c(0.05,0.0001),
     lty=2:1,lwd=1:2),override.par=FALSE)
```

`spattemp.slice` *Slicing a spatiotemporal object*

Description

Takes slices of the spatiotemporal kernel density or relative risk function estimate at desired times

Usage

```
spattemp.slice(stob, tt, checkargs = TRUE)
```

Arguments

`stob`	An object of class `stden` or `rrst` giving the spatiotemporal estimate from which to take slices.
`tt`	Desired time(s); the density/risk surface estimate corresponding to which will be returned. This value **must** be in the available range provided by `stob$tlim`; see 'Details'.
`checkargs`	Logical value indicating whether to check validity of `stob` and `tt`. Disable only if you know this check will be unnecessary.

Details

Contents of the `stob` argument are returned based on a discretised set of times. This function internally computes the desired surfaces as pixel-by-pixel linear interpolations using the two discretised times that bound each requested `tt`.

The function returns an error if any of the requested slices at `tt` are not within the available range of times as given by the `tlim` component of `stob`.

Value

A list of lists of pixel `im`ages, each of which corresponds to the requested times in `tt`, and are named as such.
If `stob` is an object of class `stden`:

`z`	Pixel images of the joint spatiotemporal density corresponding to `tt`.
`z.cond`	Pixel images of the conditional spatiotemporal density given each time in `tt`.

If `stob` is an object of class `rrst`:

`rr`	Pixel images of the joint spatiotemporal relative risk corresponding to `tt`.

`rr.cond`	Pixel images of the conditional spatiotemporal relative risk given each time in `tt`.
`P`	Only present if `tolerate = TRUE` in the preceding call to `spattemp.risk`. Pixel images of the p-value surfaces for the joint spatiotemporal relative risk.
`P.cond`	Only present if `tolerate = TRUE` in the preceding call to `spattemp.risk`. Pixel images of the p-value surfaces for the conditional spatiotemporal relative risk.

Author(s)

T.M. Davies

References

Fernando, W.T.P.S. and Hazelton, M.L. (2014), Generalizing the spatial relative risk function, *Spatial and Spatio-temporal Epidemiology*, **8**, 1-10.

See Also

`spattemp.density`, `spattemp.risk`, `bivariate.density`

Examples

```
data(fmd)
fmdcas <- fmd$cases
fmdcon <- fmd$controls

f <- spattemp.density(fmdcas,h=6,lambda=8)
g <- bivariate.density(fmdcon,h0=6)
rho <- spattemp.risk(f,g,tolerate=TRUE)

f$tlim # requested slices must be in this range

# slicing 'stden' object
f.slice1 <- spattemp.slice(f,tt=50) # evaluation timestamp
f.slice2 <- spattemp.slice(f,tt=150.5) # interpolated timestamp
par(mfrow=c(2,2))
plot(f.slice1$z$'50')
plot(f.slice1$z.cond$'50')
plot(f.slice2$z$'150.5')
plot(f.slice2$z.cond$'150.5')

# slicing 'rrst' object
rho.slices <- spattemp.slice(rho,tt=c(50,150.5))
par(mfrow=c(2,2))
plot(rho.slices$rr$'50');tol.contour(rho.slices$P$'50',levels=0.05,add=TRUE)
plot(rho.slices$rr$'150.5');tol.contour(rho.slices$P$'150.5',levels=0.05,add=TRUE)
plot(rho.slices$rr.cond$'50');tol.contour(rho.slices$P.cond$'50',levels=0.05,add=TRUE)
plot(rho.slices$rr.cond$'150.5');tol.contour(rho.slices$P.cond$'150.5',levels=0.05,add=TRUE)
```

`summary.bivden` *Summarising sparr objects*

Description

`summary` methods for classes `bivden`, `stden`, `rrs`, `rrst` and `msden`.

Usage

```
## S3 method for class 'bivden'
summary(object, ...)

## S3 method for class 'msden'
summary(object, ...)

## S3 method for class 'rrs'
summary(object, ...)

## S3 method for class 'rrst'
summary(object, ...)

## S3 method for class 'stden'
summary(object, ...)
```

Arguments

`object`	An object of class `bivden`, `stden`, `rrs`, `rrst`, or `msden`.
`...`	Ignored.

Author(s)

T.M. Davies

`tol.contour` *Plot tolerance contours*

Description

Draw contours based on a *p*-value matrix.

Usage

```
tol.contour(pim, test = c("upper", "lower", "two-sided"), ...)
```

Arguments

`pim`	A pixel `image` of *p*-values, typically obtained from a call to `tolerance`, computed with respect to a test for elevated risk.
`test`	An optional character string giving the type of manipulation to be applied to the *p*-values, corresponding to a test for significantly elevated risk (`"upper"`; default); for reduced risk (`"lower"`); or for both (`"two-sided"`).
`...`	Additional arguments to be passed to `contour`. Commonly used options include `add` (to superimpose the contours upon an existing plot); `levels` (to control the specific significance levels at which to delineate the *p*-values); and `lty` or `lwd` for aesthetics.

Details

Note that no checks on the numeric content of `pim` are made. The function assumes the pixel `image` of *p*-values in `pim` is supplied with respect to an upper-tailed test for elevated risk (this is exactly the way the *p*-value surface is returned when `tolerance` is used). This is important if one makes subsequent use of `test`, which manipulates the *p*-values to draw at desired significance `levels`.

Value

Opens a new graphics device and displays a `contour` plot if `add = FALSE`, otherwise adds the contours to the plot in the existing active graphics device.

Author(s)

T. M. Davies

Examples

```
# See ?tolerance
```

`tolerance` *Tolerance by* p*-value surfaces*

Description

Calculates a *p*-value surface based on asymptotic theory or Monte-Carlo (MC) permutations describing the extremity of risk given a fixed or adaptive kernel-smoothed density-ratio, allowing the drawing of *tolerance contours*.

Usage

```
tolerance(rs, method = c("ASY", "MC"), ref.density = NULL,
  beta = 0.025, ITER = 100, parallelise = NULL, verbose = TRUE,
  ...)
```

Arguments

`rs`	An object of class `rrs` giving the estimated relative risk function for which to calculate the *p*-value surface.
`method`	A character string specifying the method of calculation. `"ASY"` (default) instructs the function to compute the *p*-values using asymptotic theory. `"MC"` computes the values by random permutations of the data. See 'Details'.
`ref.density`	Required if `rs` is based on fixed-bandwidth estimates of the case and control densities and `method = "ASY"`. Either a pixel `im`age or an object of class `bivden` giving the reference density to use in asymptotic formulae. May be unnormalised. Ignored if `rs` is based on adaptive kernel estimates or if `method = "MC"`.
`beta`	A numeric value $0 <$ `beta` < 1 giving the fineness of the adaptive bandwidth partitioning to use for calculation of the required quantities for asymptotic adaptive *p*-value surfaces. Smaller values provide more accurate bandwidth bins at the cost of additional computing time, see Davies and Baddeley (2018); the default is sensible in most cases. Ignored if `rs` is based on fixed-bandwidth kernel estimates.
`ITER`	Number of iterations for the Monte-Carlo permutations. Ignored if `method = "ASY"`.
`parallelise`	Numeric argument to invoke parallel processing, giving the number of CPU cores to use when `method = "MC"`. Experimental. Test your system first using `parallel::detectCores()` to identify the number of cores available to you.
`verbose`	Logical value indicating whether to print function progress during execution.
`...`	Additional arguments to be passed to `risk` when `method = "MC"`. While most information needed for the MC repetitions is implicitly gleaned from the object passed to `rs`, this ellipsis is typically used to set the appropriate `epsilon` and `pilot.symmetry` values for the internal calls to `risk`.

Details

This function implements developments in Hazelton and Davies (2009) (fixed) and Davies and Hazelton (2010) (adaptive) to compute pointwise *p*-value surfaces based on asymptotic theory of kernel-smoothed relative risk surfaces. Alternatively, the user may elect to calculate the *p*-value surfaces using Monte-Carlo methods (see Kelsall and Diggle, 1995). Superimposition upon a plot of the risk surface contours of these *p*-values at given significance levels (i.e. "tolerance contours") can be an informative way of exploring the statistical significance of the extremity of risk across the defined study region.

Implementation of the Monte-Carlo method simply involves random allocation of case/control marks and re-estimation of the risk surface `ITER` times, against which the original estimate is compared. While not dependent on asymptotic theory, it is computationally expensive, and it has been suggested that it might have some undesirable practical consequences in certain settings (Hazelton and Davies, 2009). When performing the MC simulations, the same global (and pilot, if necessary) bandwidths and edge-correction regimens are employed as were used in the initial density estimates of the observed data. With regard to arguments to be passed to internal calls of `risk`, the user should take care to use `...` to set the `epsilon` value to match that which was used in creation of the object passed to `rs` (if this was set to a non-default value). Furthermore, if performing MC simulations for the adaptive relative risk function, the function borrows the value of the `beta` argument to speed things up via partitioning, and the user should additionally access `...` to set the same pilot.symmetry value as was used for creation of the object passed to rs, in the same way as for any non-default use of epsilon. This will ensure the simulations are all performed under the same conditions as were used to estimate the original risk function.

Value

A pixel image of the estimated *p*-value surface.

Note

The returned *p*-values are geared so that "smallness" corresponds to statistical significance of elevated risk, that is, an upper-tailed test. The complement of the *p*-values will yeild significance of reduced risk; a lower-tailed test. When using tol.contour, the user can control what type of contours to display.

Author(s)

T. M. Davies

References

Davies, T.M. and Baddeley A. (2018), Fast computation of spatially adaptive kernel estimates, *Statistics and Computing*, **28**(4), 937-956.

Davies, T.M. and Hazelton, M.L. (2010), Adaptive kernel estimation of spatial relative risk, *Statistics in Medicine*, **29**(23) 2423-2437.

Davies, T.M., Jones, K. and Hazelton, M.L. (2016), Symmetric adaptive smoothing regimens for estimation of the spatial relative risk function, *Computational Statistics & Data Analysis*, **101**, 12-28.

Hazelton, M.L. and Davies, T.M. (2009), Inference based on kernel estimates of the relative risk function in geographical epidemiology, *Biometrical Journal*, **51**(1), 98-109.

Kelsall, J.E. and Diggle, P.J. (1995), Kernel estimation of relative risk, *Bernoulli*, **1**, 3-16.

Examples

```
data(pbc)
h0 <- LSCV.risk(pbc,method="hazelton");h0
pbccas <- split(pbc)[[1]]
pbccon <- split(pbc)[[2]]

# ASY
riskfix <- risk(pbc,h0=h0)
fixtol1 <- tolerance(riskfix,ref.density=density(pbc,OS(pbc)))

riskada <- risk(pbc,h0=h0,adapt=TRUE,hp=NS(pbc),pilot.symmetry="pooled",davies.baddeley=0.025)
adatol1 <- tolerance(riskada)

par(mfrow=c(1,2))
```

```
plot(riskfix)
tol.contour(fixtol1,levels=c(0.1,0.05,0.01),lty=3:1,add=TRUE)
plot(riskada)
tol.contour(adatol1,levels=c(0.1,0.05,0.01),lty=3:1,add=TRUE)


# MC
fixtol2 <- tolerance(riskfix,method="MC",ITER=200)
adatol2 <- tolerance(riskada,method="MC",ITER=200,parallelise=4) # ~1 minute with parallelisation
par(mfrow=c(1,2))
plot(riskfix)
tol.contour(fixtol2,levels=c(0.1,0.05,0.01),lty=3:1,add=TRUE)
plot(riskada)
tol.contour(adatol2,levels=c(0.1,0.05,0.01),lty=3:1,add=TRUE)
```

Index