Package: spagmix (via r-universe)

August 25, 2024

Type Package

Title Artificial Spatial and Spatio-Temporal Densities on Bounded Windows

Version 0.4-2

Date 2024-06-25

Maintainer Tilman M. Davies <`tilman.davies@otago.ac.nz`>

Description Simple utilities to design and generate density functions on bounded regions in space and space-time, and simulate independent, identically distributed data therefrom. See Davies & Lawson (2019) <doi:10.1080/00949655.2019.1575066> for example.

Depends R (>= 3.5.0), spatstat (>= 3.0-0)

Imports abind, sparr, mvtnorm, spatstat.geom, spatstat.random

Suggests rgl

License GPL (>= 2)

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Anna K. Redmond [aut], Tilman M. Davies [aut, cre]

Repository CRAN

Date/Publication 2024-06-25 02:50:02 UTC

Contents

`spagmix-package` *The spagmix Package: Artificial Spatial and Spatio-Temporal Densities on Bounded Windows*

Description

Provides functions to design synthetic spatial and spatiotemporal densities and relative risk functions based mainly on Gaussian mixture distributions, and simulate independent and identically distributed data therefrom.

Details

Package:	spagmix
Version:	0.4-2
Date:	2024-06-25
License:	GPL (>= 2)

Appraisal of existing, refined, and new statistical methods for the analysis of spatial and spatiotemporal point pattern data usually involves numeric experimentation. Motivated by relevant problems in nonparametric density estimation (see e.g. Wand & Jones, 1995), `spagmix` ("spatial Gaussian mixtures") provides some simple utilities for designing heterogeneous density and density-ratio or *relative risk* (Bithell 1990, 1991; Kelsall & Diggle, 1995) functions in space and space-time (see Fernando & Hazelton, 2014 for the latter). The package is also capable of producing realisations of (possibly inhomogeneous) spatial log-Gaussian Cox process intensities (Møller et al., 1998; see also Davies & Hazelton, 2013).

Additionally, functions for simulating datasets given these scenarios are included. For examples of how these kinds of synthetic functions have been used in simulation studies in various publications, see for example Clark & Lawson, 2004; Davies & Hazelton, 2010; Davies, 2013a,b; Davies & Hazelton, 2013; Fernando et al., 2014; Davies et al., 2016; Davies et al., 2018a; and Davies & Lawson, 2019.

We have designed the objects of `spagmix` to use and be compatible with standard object classes of the `spatstat` (Baddeley & Turner, 2005; Baddeley et al., 2015) and `sparr` (Davies et al., 2018b) packages. The content of `spagmix` can be broken up as follows:

Artificial polygonal windows
Some pre-made synthetic spatial windows; these are all single closed polygons as objects of class `owin` and are lazy-loaded:
`bx`, `heart`, `shp1`, `shp2`, `star`, `toywin`

Spatial scenarios
`sgmix` is used to create spatial (2D) Gaussian mixture distributions on a bounded subset of the plane.
`rgmix` also creates 2D Gaussian mixture densities, but does so by stochastic generation of the contributing bumps.
`rrmix` creates Gaussian mixture relative risk scenarios based on a supplied control density (see e.g. Davies & Hazelton, 2010). `lgcpmix` generates a spatial log-Gaussian Cox process intensity in space, given a deterministic intensity function and residual correlation governed by a stochastic realisation of a Gaussian field with a specified covariance structure.

Spatiotemporal scenarios
`stgmix` is used to create spatiotemporal (3D) Gaussian mixture densities on a bounded subset of the plane and a single closed interval in time.
`stkey` is used to create spatiotemporal densities by pixel-wise interpolation of multiple spatial image 'keyframes'. `rrstmix` is a spatiotemporal version of `rrmix`, used to create artificial spatiotemporal relative risk functions. Note the control density may be purely spatial, representing a distribution 'at-risk' points that does not change over time (Fernando & Hazelton, 2014).

Data generation
To generate purely spatial data for a single spatial density, the user is directed to `rpoint` of the `spatstat` package or `rimpoly` of the `sparr` package.
`rpoispoly` is a wrapper of `rimpoly`, and is used to generate realisations of Poisson point processes in space, given an intensity function. `rrpoint` is a wrapper of `rimpoly`, and is used to generate iid datasets based on a synthetic spatial relative risk surface object.
`rstpoint` is a 3D rejection algorithm for sampling iid data from a supplied spatiotemporal density.
`rrstpoint` is a wrapper of `rstpoint` to generate iid datasets from a synthetic spatiotemporal relative risk surface object.

Miscellaneous
`plot.stim` is an `S3` plotting method for spatiotemporal density objects.
`stintegral` computes the 3D integral of a spatiotemporal density object.
`unify.owin` is a wrapper for `affine` that transforms any spatial `owin` to fall inside the unit square.

Dependencies/Imports

Depends on `spatstat` functionality (Baddeley & Turner, 2005; Baddeley et al., 2015) and imports from `abind` (Plate & Heiberger, 2016), `sparr` (Davies et al., 2018b), and mvtnorm (Genz et al., 2018). We also highly recommend the `rgl` package (Adler et al., 2018) which can be used to create interactive plots of spatiotemporal data.

Author(s)

A.K. Redmond and T.M. Davies
Dept. of Mathematics & Statistics, University of Otago, Dunedin, New Zealand

Maintainer: T.M.D. <`tilman.davies@otago.ac.nz`>

References

Adler, D., Murdoch, D. and others (2018), rgl: 3D Visualization Using OpenGL, R package version 0.99.16 https://CRAN.R-project.org/package=rgl

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

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

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.

Clark, A.B. and Lawson, A.B. (2004), An evaluation of non-parametric relative risk estimators for disease maps, *Computational Statistics & Data Analysis*, **47**, 63-78.

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

Davies, T.M. (2013b), Scaling oversmoothing factors for kernel estimation of spatial relative risk, *Epidemiological Methods*, **2**(1), 67-83.

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. and Hazelton, M.L. (2013), Assessing minimum contrast parameter estimation for spatial and spatiotemporal log-Gaussian Cox processes, *Statistica Neerlandica*, **67**(4), 355-389.

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.

Davies, T.M. and Lawson, A.B. (2019), An evaluation of likelihood-based bandwidth selectors for spatial and spatiotemporal kernel estimates, *Journal of Statistical Computation and Simulation*, **89** 1131-1152.

Davies, T.M., Flynn, C.R. and Hazelton, M.L. (2018a), On the utility of asymptotic bandwidth selectors for spatially adaptive kernel density estimation, *Statistics & Probability Letters*, **138**, 75-81.

Davies, T.M., Marshall, J.C. and Hazelton, M.L. (2018b), Tutorial on kernel estimation of continuous spatial and spatiotemporal relative risk, *Statistics in Medicine*, **37**(7), 1191-1221.

Fernando, W.T.P.S., Ganesalingam, S. and Hazelton, M.L. (2014), A comparison of estimators of the geographical relative risk function, *Journal of Statistical Computation and Simulation*, **84**(7), 1471-1485.

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

Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F. and Hothorn, T. (2018), mvtnorm: Multivariate Normal and t Distributions, R package version 1.0-8. URL http://CRAN.R-project.org/package=mvtnorm

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

Møller, J., Syversveen, A.R. and Waagepetersen, R.P. (1998), Log-Gaussian Cox processes, *Scandinavian Journal of Statistics*, **25**(3) 451–482.

Plate, T. and Heiberger, R. (2016), abind: Combine Multidimensional Arrays, R package version 1.4-5. https://CRAN.R-project.org/package=abind

`lgcpmix` *Generate a spatial log-Gaussian Cox process intensity*

Description

Generate a realisation of a (possibly inhomogeneous) log-Gaussian Cox process (LGCP) spatial intensity function with an identifiable mean structure.

Usage

```
lgcpmix(lambda, ...)
```

Arguments

`lambda`	A pixel `im`age giving the deterministic spatial intensity as the mean structure of the process. The generated Gaussian field will match the dimensions, resolution and domain of this object.
`...`	Additional arguments controlling the Gaussian random field to be passed to `rLGCP`. Minimally, the user will need to supply `param` and `model`. See ‘Details’.

Details

This function allows the user to generate a spatial intensity function Γ of the form

$\Gamma(x) = \lambda(x)\exp[Y(x)]$

for $x \in W$, where $\lambda(x)$ (passed to `lambda`) is the deterministic spatial intensity over the spatial domain W, and $Y(x)$ is a Gaussian random field on W. This Gaussian field, implemented through rLGCP, is defined with a particular spatial covariance function (specified via the model argument given to ...) with variance and scale parameters σ^2 and ϕ respectively, as well as any additionally required parameter values (all specified in the param argument, also given to ...). For example, requesting model = "exponential" with param = list(var=σ^2,scale=ϕ)) imposes an exponential covariance structure on the generated field whereby $Cov(u) = \sigma^2 \exp(-u/\phi)$ for the Euclidean distance between any two spatial locations u.

The mean parameter μ of the Gaussian field Y is internally fixed at $-\sigma^2/2$; negative half the variance. This is for identifiability of the mean structure, forcing $E[Y(x)] = 1$ for all $x \in W$ (see theoretical properties in Møller et al., 1998). In turn, this means the deterministic intensity function $\lambda(x)$ is solely responsible for describing fixed heterogeneity in spatial intensity over W (as such, the pixel image supplied to lambda as $\lambda(x)$ must be non-negatively-valued and yield a finite integral), with the randomly generated Gaussian field left to describe residual stochastic spatial correlation. This presents a highly flexible class of model, even with stationarity and isotropy of the Gaussian field itself, and is intuitively sensible in a variety of applications. See Diggle et al. (2005) and Davies & Hazelton (2013) for example. Given this, any user-supplied value of mu in ... (intended for rLGCP) is irrelevant and will be ignored/overwritten.

To generate a subsequent dataset, use e.g. rpoispp or rpoispoly.

Value

A pixel image giving the generated intensity function, comprised of the product of lambda (fixed, and unchanging in repeated calls to this function) and the exponentiated Gaussian field (with expected value 1, this is stochastic and varies in repeated calls).

Author(s)

T.M. Davies.

References

Davies, T.M. and Hazelton, M.L. (2013), Assessing minimum contrast parameter estimation for spatial and spatiotemporal log-Gaussian Cox processes, *Statistica Neerlandica*, **67**(4) 355–389.

Diggle, P.J., Rowlingson, B. and Su, T. (2005), Point process methodology for on-line spatio-temporal disease surveillance, *Environmetrics*, **16** 423–434.

Møller, J., Syversveen, A.R. and Waagepetersen, R.P. (1998), Log-Gaussian Cox processes, *Scandinavian Journal of Statistics*, **25**(3) 451–482.

See Also

rLGCP, rpoispp, rpoispoly

Examples

```
## Homogeneous example ##

# Create constant intensity image integrating to 500

homog <- as.im(as.mask(toywin))
homog <- homog/integral(homog)*500
```

```
# Corresponding LGCP realisations using exponential covariance structure
oldpar <- par(mfrow=c(2,2),mar=rep(1.5,4))
for(i in 1:4){
  temp <- lgcpmix(homog,model="exponential",param=list(var=1,scale=0.2))
  plot(temp,main=paste("Realisation",i),log=TRUE)
}
par(oldpar)


## Inhomogeneous examples ##

# Create deterministic trend

mn <- cbind(c(0.25,0.8),c(0.31,0.82),c(0.43,0.64),c(0.63,0.62),c(0.21,0.26))
v1 <- matrix(c(0.0023,-0.0009,-0.0009,0.002),2)
v2 <- matrix(c(0.0016,0.0015,0.0015,0.004),2)
v3 <- matrix(c(0.0007,0.0004,0.0004,0.0011),2)
v4 <- matrix(c(0.0023,-0.0041,-0.0041,0.0099),2)
v5 <- matrix(c(0.0013,0.0011,0.0011,0.0014),2)
vr <- array(NA,dim=c(2,2,5))
for(i in 1:5) vr[,,i] <- get(paste("v",i,sep=""))
intens <- sgmix(mean=mn,vcv=vr,window=toywin,p0=0.1,int=500)


# Two realisations (identical calls to function), exponential covariance structure

r1exp <- lgcpmix(lambda=intens,model="exponential",param=list(var=2,scale=0.05))
r2exp <- lgcpmix(lambda=intens,model="exponential",param=list(var=2,scale=0.05))


# Two more realisations, Matern covariance with smoothness 1

r1mat <- lgcpmix(lambda=intens,model="matern",param=list(var=2,scale=0.05,nu=1))
r2mat <- lgcpmix(lambda=intens,model="matern",param=list(var=2,scale=0.05,nu=1))

# Plot everything, including 'intens' alone (no correlation)

oldpar <- par(mar=rep(2,4))
layout(matrix(c(1,2,4,1,3,5),3))
plot(intens,main="intens alone",log=TRUE)
plot(r1exp,main="realisation 1\nexponential covar",log=TRUE)
plot(r2exp,main="realisation 2\nexponential covar",log=TRUE)
plot(r1mat,main="realisation 1\nMatern covar",log=TRUE)
plot(r2mat,main="realisation 2\nMatern covar",log=TRUE)
par(oldpar)

# Plot example datasets
dint <- rpoispoly(intens,w=toywin)
d1exp <- rpoispoly(r1exp,w=toywin)
d2exp <- rpoispoly(r2exp,w=toywin)
d1mat <- rpoispoly(r1mat,w=toywin)
```

```
d2mat <- rpoispoly(r2mat,w=toywin)

oldpar <- par(mar=rep(2,4))
layout(matrix(c(1,2,4,1,3,5),3))
plot(dint,main="intens alone",log=TRUE)
plot(d1exp,main="realisation 1\nexponential covar",log=TRUE)
plot(d2exp,main="realisation 2\nexponential covar",log=TRUE)
plot(d1mat,main="realisation 1\nMatern covar",log=TRUE)
plot(d2mat,main="realisation 2\nMatern covar",log=TRUE)
par(oldpar)
```

`plot.stim` *Plotting 'stim' objects*

Description

`plot` method for class `stim`.

Usage

```
## S3 method for class 'stim'
plot(x, fix.range = FALSE, sleep = 0.2, override.par = TRUE, ...)
```

Arguments

`x`	An object of class `stim`.
`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 `...` and passed to `plot.im`.
`sleep`	Single positive numeric value giving the amount of time (in seconds) to `Sys.sleep` before drawing the next image in the animation.
`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` (see 'Details').

Details

Actual visualisation is deferred to `plot.im`, for which there are a variety of customisations available the user can access via `...`.

The `stim` 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 active graphics device.

Author(s)

T.M. Davies

Examples

```
# See help(stgmix) and help(stkey) for examples
```

---

rgmix	*Random bivariate Gaussian mixture density generation*

Description

Generates a pixel image of a bivariate normal mixture density observed on a bounded window using a specified number of contributing densities with randomly selected means and variance-covariance matrices.

Usage

```
rgmix(N, window, v = 4, S = NULL, extras = FALSE, ...)
```

Arguments

`N`	The number of Gaussian components to generate for the mixture.
`window`	An object of class `owin` giving the observational window on which the mixture density is defined.
`v`	The degrees of freedom for the inverse-Wishart distribution of the variance-covariance matrices (must be at least 4). The default value of 4 ensures the generated covariance matrices are centered on `S`.
`S`	A symmetric, positive-definite 2×2 scale matrix for the inverse-Wishart distribution of the variance-covariance matrices.
`extras`	A logical value indicating whether, in addition to returning the pixel `im`age of the final mixture density, to also return the randomly realised mean locations and corresponding variance-covariance matrices. See 'Value'.
`...`	Additional arguments to be passed to `sgmix`. See 'Details'.

Details

This function creates and returns a bivariate Gaussian mixture density on a bounded `window` based on `N` randomly generated mean locations and corresponding randomly generated variance-covariance matrices. First, the `N` mean locations are generated based on a uniform distribution over the spatial `window`. Each location is then associated with a covariance matrix generated from an inverse-Wishart distribution with `v` degrees of freedom and scale matrix `S`.

Once the above steps are completed, the function calls `sgmix` with the chosen mean and covariance matrices, thereby creating the Gaussian mixture. Resolution and other aspects of this call can be controlled by using `...`, passing the contents internally to `sgmix`. By default, all generated Gaussian components have equal weight in contributing to the final mixture density. The user can alter this by passing `p0` and `p` to the `...`, though should take care that the length of `p` is `N`, and that `p0` and `p` sum to 1, as outlined in the documentation for `sgmix`.

Value

If `extras = FALSE` (default), then a pixel `im`age of the final mixture density. If `extras = TRUE`, a list is returned with members `f` (the pixel `im`age of the final mixture density); `mn` (a $2\times$ `N` matrix with each column giving the mean location of each of the `N` Gaussian bumps); and `vcv` (a $2 \times 2\times$ `N` array with layers giving the covariance matrices associated with the means in the columns of `mn`).

Author(s)

A.K. Redmond and T.M. Davies

Examples

```
set.seed(321)
dens1 <- rgmix(7,window=toywin)
plot(dens1)

set.seed(456)
dens2 <- rgmix(7,window=toywin)
plot(dens2)

# Explicitly return details of generated means and covariances
set.seed(321)
dens1.detailed <- rgmix(7,window=toywin,extras=TRUE)
dens1.detailed$f
dens1.detailed$mn
dens1.detailed$vcv

# Set underlying uniform proportion and compare with dens2 from above
set.seed(456)
dens2.wunif <- rgmix(7,window=toywin,p0=0.3)
plot(rpoint(500,dens2))
plot(rpoint(500,dens2.wunif))

# Explicitly setting scale matrix for inverse-wishart generation of covariances
dens3 <- rgmix(3,window=toywin,S=matrix(c(0.025,-0.004,-0.004,0.02),2))
plot(dens3)
```

`rpoispoly` *Generate a Poisson point pattern in a polygonal window*

Description

Generates a single realisation of a spatial Poisson point process based on a pixel `im`age and a polygonal `owin`.

Usage

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

Arguments

`z`	A pixel image of class `im` defining the spatial intensity function of the points. The number of points generated, n, will be found as a randomly generated Poisson variate with mean parameter equal to the integral of `z`.
`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' and 'Warning'.
`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 is a wrapper function for `rimpoly` that operates much like `rpoispp`, but with artificial corrections at the edges of boundary pixels. This allows the user to generate a realisation of a 2D Poisson point process using a supplied pixel `im`age as the spatial intensity function, but return the result with a polygonal `owin` instead of a binary image mask.

Let n be a randomly generated integer from a Poisson distribution with mean given by the integral of the intensity function `z`. When the user specifies their own polygonal window in `w`, 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 union of the pixels of `z` is used, obtained via `as.polygonal(Window(z))`). The loop is necessary because the standard behaviour of `rpoispp` can (and often does) yield points that sit in corners of pixels which lie outside a corresponding irregular polygon `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 Poisson-generated points, defined with the polygonal `owin`, `w`.

Warning

Note that this is an artificial correction that forces the Poisson-generated number of n points to be found inside *any* supplied polygon `w` (even if `w` only partially covers the domain of `z`). As such, this function only makes sense in terms of the theory of a Poisson point process if the polygon `w` corresponds exactly to the pixellised intensity. For practical intents and purposes, it therefore must be assumed in using this function that a supplied polygon `w` is/was the original basis for the discretisation into the pixel image for the purposes of producing the intensity `z`, and hence that any adverse effects arising from imposing `w` as the window of the final result are negligible. See 'Examples'.

Author(s)

T.M. Davies

References

Diggle, P.J. (2014) *Statistical Analysis of Spatial and Spatiotemporal Point Patterns*, 3rd Ed, Chapman & Hall, Boca Raton, USA.

See Also

`rpoint`, `rimpoly`, `rpoispp`

Examples

```
mn <- cbind(c(0.25,0.8),c(0.31,0.82),c(0.43,0.64),c(0.63,0.62),c(0.21,0.26))
v1 <- matrix(c(0.0023,-0.0009,-0.0009,0.002),2)
v2 <- matrix(c(0.0016,0.0015,0.0015,0.004),2)
v3 <- matrix(c(0.0007,0.0004,0.0004,0.0011),2)
v4 <- matrix(c(0.0023,-0.0041,-0.0041,0.0099),2)
v5 <- matrix(c(0.0013,0.0011,0.0011,0.0014),2)
vr <- array(NA,dim=c(2,2,5))
for(i in 1:5) vr[,,i] <- get(paste("v",i,sep=""))
intens <- sgmix(mean=mn,vcv=vr,window=toywin,p0=0.1,int=500)

aa <- rpoispp(intens) # Default spatstat function
bb <- rpoispoly(intens) # No polygon supplied; just uses pixel union
cc <- rpoispoly(intens,w=toywin) # Original irregular polygon

plot(intens,log=TRUE)
plot(aa,main=paste("aa\nn =",npoints(aa)))
plot(bb,main=paste("bb\nn =",npoints(bb)))
plot(cc,main=paste("cc\nn =",npoints(cc)))
```

`rrmix` *Spatial relative risk surface generation*

Description

Generates an appropriately scaled spatial (bivariate) relative risk surface using a supplied control density and N isotropic Gaussian-style hotspots.

Usage

```
rrmix(g, rhotspots, rsds, rweights, rbase = 1, log = TRUE)
```

Arguments

`g`	A pixel `im`age representing the control density; this will be internally rescaled to integrate to 1 if it does not already do so.
`rhotspots`	A $2 \times N$ matrix giving the centers of the N peaks and troughs in the relative risk density.

`rsds`	A positive numeric vector of length N giving the isotropic standard deviations for each relative Gaussian peak or trough.
`rweights`	A vector of length N giving relative weightings for each peak (positive weights) or trough (negative).
`rbase`	The base level of the relative risk surface (default is 1). The peaks and troughs will be added or subtracted from this base level prior to normalisation.
`log`	A logical value. If `TRUE` (default), the relative risk surface is returned logged.

Details

A useful tool for the comparison of two estimated density functions on the same spatial region $W \subset R^2$ is the *relative risk function*, r, (Bithell, 1990; 1991; Kelsall and Diggle, 1995), defined simply as a density-ratio:

$r(x) = f(x)/g(x); x \in W.$

Various methods have been developed to improve estimation of r, most commonly with a motivation in geographical epidemiology, where the 'numerator' density f pertains to the observed disease cases and the 'denominator' density g reflects the distribution of the at-risk controls (Kelsall and Diggle, 1995; Hazelton and Davies, 2009; Davies and Hazelton, 2010). To test newly developed methodology, simulations based on known relative risk scenarios are usually necessary. This function allows the user to design such scenarios, as used in Hazelton and Davies (2009), Davies and Hazelton (2010), and Davies (2013) for example.

This function calculates a relative risk surface based on N Gaussian-style 'bumps' added and subtracted from a base level of `rbase`, with the peaks and troughs centered at the coordinates given by `rhotspots` with relative weights of `rweights` and isotropic standard deviations of `rsds`. The risk surface r is computed as

$r(x) \propto \texttt{rbase} + \sum_{i=1}^{N} \texttt{rweights[}i\texttt{]}*exp(-0.5*\texttt{rsds[}i\texttt{]}^(-2)*\|x-\texttt{rhotspots[,}i\texttt{]}\|^2)$

where $\| . \|$ denotes Euclidean norm. Because f and g are both densities, the risk surface as defined above must then be rescaled with respect to the supplied control density g (argument g) to ensure that

$\int_W r(x)g(x)dx = 1$

This is automatically performed inside the function. The case density that gives rise to the designed r is then easily recovered because $f = r * g$. By default, the function returns the log-relative risk surface $\log r = \log f - \log g$ alongside the case and control densities.

Value

An object of class `rrim`. This is a `solist` of three pixel `im`ages: `f` as the case density, `g` the control density (a copy of the argument of the same name, integrating to 1), and `r` as the (log) relative risk surface.

Author(s)

A.K. Redmond and 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.

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

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

Examples

```
set.seed(1)
gg <- rgmix(3,window=toywin,S=matrix(c(0.08^2,0,0,0.1^2),nrow=2),p0=0.2)

rho <- rrmix(g=gg,
             rhotspots=cbind(c(0.8,0.3),c(0.4,0.4),c(0.6,0.5),c(0.3,0.5)),
             rsds=c(0.005,0.025,0.01,0.025),
             rweights=c(3,2,10,5)*10)

rho.sample <- rrpoint(c(400,800),rho,toywin)


oldpar <- par(mfrow=c(2,2))
plot(rho$g,main="control density")
plot(rho$f,main="case density")
plot(rho$r,main="log relative risk surface")
plot(rho.sample$controls,main="sample data")
points(rho.sample$cases,col=2)
legend("topright",col=2:1,legend=c("cases","controls"),pch=1)
par(oldpar)
```

`rrpoint` *Generate random case/control points in space or space-time*

Description

Generates a pair of random, independent point patterns corresponding to a case density and a control density, for relative risk analyses.

Usage

```
rrpoint(n, r, W = NULL, correction = 1.1, maxpass = 50)
rrstpoint(n, r, W = NULL, correction = 1.5, maxpass = 50)
```

Arguments

`n`	The number of points to be generated. This must be a numeric vector of length 2 giving the number of points to generate for the case and control densities respectively. Alternatively a single number can be supplied; then the same number of points is generated for both densities.
`r`	The relative risk surface object containing the definitions of the case and control probability densities: an object of class `rrim` or `rrs` for `rrpoint`, or an object of class `rrstim` or `rrst` for `rrstpoint`.
`W`	The polygonal `owin` defining the spatial window on which the density is defined. If `NULL`, this will be set to the `as.polygonal` version of the pixel images stored in `r`. 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.
`maxpass`	The maximum number of passes allowed before the function exits. If this is reached before `n` points are found with respect to the spatial or spatiotemporal domains of `r`, a warning is issued.

Details

These functions randomly generate a pair of independent spatial or spatiotemporal point patterns of `n` points based on the case and control density functions stored in `r`. At any given pass for each density, `n * correction` points are generated and rejection sampling is used to accept some of the points; this is repeated until the required number of points is found.

The argument `W` is optional, but is useful when the user wants the spatial window of the resulting point pattern to be a corresponding irregular polygon, as opposed to being based on the boundary of a binary image mask (which, when the pixel `im`ages in `r` are converted to a polygon directly, gives jagged edges based on the union of the pixels).

Value

A list with two components, `cases` and `controls`, each of which is an object of class `ppp` containing the `n` generated points. for spatiotemporal densities, the `marks` of the object will contain the correspondingly generated observation times.

Author(s)

T.M. Davies

Examples

```
# Using 'rrim' object:
set.seed(1)
gg <- rgmix(3,window=toywin,S=matrix(c(0.08^2,0,0,0.1^2),nrow=2),p0=0.2)
rho <- rrmix(g=gg,
             rhotspots=cbind(c(0.8,0.3),c(0.4,0.4),c(0.6,0.5),c(0.3,0.5)),
             rsds=c(0.005,0.025,0.01,0.025),
             rweights=c(3,2,10,5)*10)

rho.sample <- rrpoint(n=c(400,800),r=rho,W=toywin)

oldpar <- par(mfrow=c(2,2))
plot(rho$g,main="control density")
plot(rho$f,main="case density")
plot(rho$r,main="log relative risk surface")
plot(rho.sample$controls,main="sample data")
points(rho.sample$cases,col=2)
legend("topright",col=2:1,legend=c("cases","controls"),pch=1)
par(oldpar)


# Using 'rrs' object:
require("sparr")
data(pbc)
pbccas <- split(pbc)$case
pbccon <- split(pbc)$control
h0 <- OS(pbc,nstar="geometric")
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)
pbcrr <- risk(f,g,tolerate=TRUE,verbose=FALSE)

pbcrr.pt <- rrpoint(n=1000,r=pbcrr)

par(mfrow=c(1,3))
plot(pbcrr)
plot(pbcrr.pt$cases)
plot(pbcrr.pt$controls)


# Using 'rrstim' object:
set.seed(321)
gg <- rgmix(7,window=shp2)
rsk <- rrstmix(g=gg,rhotspots=matrix(c(-1,-1,2,2.5,0,5),nrow=3),
               rsds=sqrt(cbind(rep(0.75,3),c(0.05,0.01,0.5))),
               rweights=c(-0.4,7),tlim=c(0,6),tres=64)
plot(rsk$r,fix.range=TRUE)

rsk.pt <- rrstpoint(1000,r=rsk,W=shp2)

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

```
plot(rsk.pt$cases)
plot(rsk.pt$controls)


# Using 'rrst' object:
require("sparr")
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)

rho.pt <- rrstpoint(1000,r=rho)

par(mfrow=c(1,2))
plot(rho.pt$cases)
plot(rho.pt$controls)
```

`rrstmix` *Spatiotemporal relative risk surface generation*

Description

Generates an appropriately scaled spatiotemporal (trivariate) relative risk surface using a supplied control density and N Gaussian-style hotspots.

Usage

```
rrstmix(g, rhotspots, rsds, rweights, rbase = 1, log = TRUE,
        tlim = NULL, tres = NULL)
```

Arguments

`g`	The control density as a `stim`, `stden`, or `im` object; this will be internally rescaled to integrate to 1 if it does not already do so. When a `stim` or `stden` object, the resolution and domain of the final result will be the same as this. When this argument is passed an object of class `im`, the function assumes a static (unchanging) control density over time (see Fernando and Hazelton, 2014), and the user must additionally specify `tlim` and `tres`.
`rhotspots`	A $3 \times N$ matrix specifying the spatiotemporal coordinates of the N peaks and troughs in the relative risk density. The three entries down each column will be respectively interpreted as x-coord., y-coord., and time-coordinate of each Gaussian bump.
`rsds`	A $3 \times N$ strictly positive numeric matrix specifying the standard deviations along each axis of each of the N bumps, the ordering of the components in each column is the same as `rhotspots`.

`rweights`	A vector of length N giving relative weightings for each peak (positive weight) or trough (negative).
`rbase`	The base level of the relative risk surface (default is 1). The peaks and troughs will be added or subtracted from this base level prior to normalisation.
`log`	A logical value. If `TRUE` (default), the relative risk surface is returned logged.
`tlim`	Only used if `g` is a pixel `im`age object. A vector of length 2 giving the boundaries of the time interval on which the relative risk surface will be defined.
`tres`	Only used if `g` is a pixel `im`age object. The resolution along the temporal axis of the final result.

Details

This function is the spatiotemporal (trivariate) equivalent of `rrmix`. See 'Details' in the documentation for that function for more information.

Value

An oject of class `rrstim`. This is a list with the following components:

`f`	An object of class `stim` giving the case density.
`g`	A copy of the object passed to the argument `g`, possibly renormalised to integrate to 1 if this was necessary. If `g` was originally an `im`, this will be converted to an object of class `stim`.
`r`	An object of class `stim` giving the (log) relative risk surface.

Author(s)

A.K. Redmond and 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.

Examples

```
# time-varying control density
gg1 <- stgmix(mean=matrix(c(2,1,3,0,-1,5),nrow=3),
             vcv=array(c(1,0,0,0,1,0,0,0,1,2,0,0,0,1,0,0,0,2),dim=c(3,3,2)),
             window=shp2,tlim=c(0,6))
rsk1 <- rrstmix(g=gg1,rhotspots=matrix(c(-2,0,2,1,2,5.5),nrow=3),
               rsds=sqrt(cbind(rep(1.5,3),rep(0.25,3))),rweights=c(-0.5,5))
plot(rsk1$g,sleep=0.1,fix.range=TRUE)
plot(rsk1$f,sleep=0.1,fix.range=TRUE)
plot(rsk1$r,sleep=0.1,fix.range=TRUE)

# time-constant control density
set.seed(321)
gg2 <- rgmix(7,window=shp2)
```

```
rsk2 <- rrstmix(g=gg2,rhotspots=matrix(c(-1,-1,2,2.5,0,5),nrow=3),
                rsds=sqrt(cbind(rep(0.75,3),c(0.05,0.01,0.5))),
                rweights=c(-0.4,7),tlim=c(0,6),tres=64)
plot(rsk2$g,sleep=0.1,fix.range=TRUE)
plot(rsk2$f,sleep=0.1,fix.range=TRUE)
plot(rsk2$r,sleep=0.1,fix.range=TRUE)
```

---

rstpoint *Generate random points in space-time*

Description

Generates a random spatiotemporal point pattern containing n independent, identically distributed points with a specified distribution.

Usage

```
rstpoint(n, f, W = NULL, correction = 1.5, maxpass = 50)
```

Arguments

`n`	The number of points to be generated.
`f`	The probability density of the points, an object of class `stim` or `stden`.
`W`	The polygonal `owin` defining the spatial window on which the density is defined. If `NULL`, this will be set to the `as.polygonal` version of the pixel images stored in `f`. 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.
`maxpass`	The maximum number of passes allowed before the function exits. If this is reached before `n` points are found with respect to the spatiotemporal domain of `f`, a warning is issued.

Details

This function randomly generates a spatiotemporal point pattern of exactly `n` points based on the density function `f`. At any given pass, `n` * `correction` points are generated and rejection sampling is used to accept some of the points; this is repeated until the required number of points is found.

The argument `W` is optional, but is useful when the user wants the spatial window of the resulting point pattern to be a corresponding irregular polygon, as opposed to being based on the boundary of a binary image mask (which, when the pixel `im`ages in `f` are converted to a polygon directly, gives jagged edges based on the union of the pixels).

Value

An object of class `ppp` containing the `n` generated points. The `marks` of the object contain the correspondingly generated observation times.

Author(s)

A.K. Redmond and T.M. Davies

Examples

```
r1a <- sgmix(cbind(c(0.5,0.5)),vcv=0.01,window=toywin,p0=0.5,p=c(0.5),res=128)
r1b <- sgmix(cbind(c(0.5,0.5),c(0.4,0.6)),vcv=c(0.06,0.015),window=toywin,
             p0=0.1,p=c(0.5,0.4),res=128)
r1c <- sgmix(cbind(c(0.4,0.6)),vcv=c(0.1),window=toywin,p0=0.1,p=c(0.9),res=128)
sts1 <- stkey(start=r1a,
              stop=r1c,
              tlim=c(1,10),
              tres=64,
              window=toywin,
              kf=solist(r1a,r1b),
              kftimes=c(2,6),
              fscale=0.1+0.9*dnorm(seq(-3,3,length=64),mean=0,sd=1))
plot(sts1,sleep=0.1)

Y <- rstpoint(500,sts1,W=toywin,correction=10,maxpass=500)
plot(Y)

require("rgl")
plot3d(Y$x,Y$y,marks(Y))
```

`sgmix` *Bivariate Gaussian mixture density generation*

Description

Generates a pixel image of a specified bivariate normal mixture density observed on a bounded window.

Usage

```
sgmix(mean, vcv, window, p0 = 0, p = NULL, resolution = 128, int = 1)
```

Arguments

mean	A $2\times N$ matrix specifying the means of each of N contributing normal densities.
vcv	Either a $2 \times 2 \times N$ array specifying the variance-covariance matrices of each contributing density, or a numeric vector of length N giving the isotropic **standard deviations** of each contributing density. An error is thrown if the function encounters anything but a symmetric, positive-definite covariation specification for each component.
window	An object of class owin giving the observational window on which the mixture density is defined.

`p0`	The proportion of uniform density that contributes to the mixture (default is 0).
`p`	A numeric vector of the N proportions for each contributing density (default is equal proportions for each density, after subtracting `p0`). Together, `p0` and `p` must sum to exactly 1.
`resolution`	The number of pixels along each side of the grid for the pixel image (default is 128).
`int`	A positive numeric value for post-hoc rescaling of the density (useful if the user wishes to return an *intensity* function). Defaults to 1 for no change in scaling.

Details

This function generates a pixel `im`age of a 2D density function made of a mixture of N bivariate normals; each component is restricted to conserve probability mass over a bounded subset of the plane. A warning will appear if less than 1% of the integral of each Gaussian bump is inside the observational window.

Value

An object of class `im` giving the mixture density.

Author(s)

A.K. Redmond

Examples

```
# Example using isotropic standard deviations
m1 <- c(0.4,0.5)
m2 <- c(0.2,0.7)
s1 <- 0.1
s2 <- 0.025
dens1 <- sgmix(mean=cbind(m1,m2),vcv=c(s1,s2),window=toywin,p0=0.3,p=c(0.5,0.2))

plot(dens1,log=TRUE)
pts1 <- rpoint(200,dens1) # generate random points via spatstat.core::rpoint
points(pts1)


# Example using full covariance matrices
mn <- cbind(c(0.25,0.8),c(0.31,0.82),c(0.43,0.64),c(0.63,0.62),c(0.21,0.26))
v1 <- matrix(c(0.0023,-0.0009,-0.0009,0.002),2)
v2 <- matrix(c(0.0016,0.0015,0.0015,0.004),2)
v3 <- matrix(c(0.0007,0.0004,0.0004,0.0011),2)
v4 <- matrix(c(0.0023,-0.0041,-0.0041,0.0099),2)
v5 <- matrix(c(0.0013,0.0011,0.0011,0.0014),2)
vr <- array(NA,dim=c(2,2,5))
for(i in 1:5) vr[,,i] <- get(paste("v",i,sep=""))
dens2 <- sgmix(mean=mn,vcv=vr,window=toywin,p0=0.1)

plot(dens2,log=TRUE)
pts2 <- rpoint(200,dens2)
```

```
points(pts2)
```

stgmix *Trivariate Gaussian mixture density generation*

Description

Generates a pixel image array of a specified trivariate normal mixture density observed on a bounded window in space and time.

Usage

```
stgmix(mean, vcv, window, tlim, p0 = 0, p = NULL, sres = 128, tres = sres, int = 1)
```

Arguments

`mean`	A $3xN$ matrix specifying the means of each of N contributing normal densities; each component in the order of (x-coord, y-coord, time-coord).
`vcv`	A $3x3xN$ array specifying the variance-covariance matrices of each contributing density.
`window`	An object of class `owin` giving the spatial observational window on which the mixture density is defined.
`tlim`	A vector of length 2 giving the boundaries of the time interval on which the mixture density is defined.
`p0`	The proportion of uniform density that contributes to the final mixture (default is 0).
`p`	A numeric vector of the N proportions for each contributing density (default is equal proportions for each density, after subtracting `p0`). Together, `p0` and `p` must sum to exactly 1.
`sres`	The spatial resolution (number of pixels) along each side of the spatial grid (default is 128).
`tres`	The temporal resolution (default is to equate with `sres`).
`int`	A positive numeric value for post-hoc rescaling of the density (useful if the user wishes to return a spatiotemporal *intensity* function). Defaults to 1 for no change in scaling.

Details

This function creates a 3D array of a density function made up of a mixture of N trivariate normals with the interpretation of a continuous probability density function in space-time. As such, each component is restricted to conserve mass over a 3D region specified by a fixed polygonal `window` in space, stretched over defined temporal limits (`tlim`). A warning will appear if less than 1% of the integral of each Gaussian bump is inside this observational spatiotemporal polyhedron.

Value

An object of class `stim` giving the trivariate density. This is a list with six components:

`a`	The `sres` x `sres` x `tres` array of the specified density.
`v`	A pixel `im`age version of `a`, provided as a `solist` of length `tres`, with each member being the spatial `im`age slice of the 3D density at each of the time-coordinate values.
`xcol`	Grid coordinates in the spatial x-axis (corresponds to each spatial `im`age in `v`).
`yrow`	Grid coordinates in the spatial y-axis (corresponds to each spatial `im`age in `v`).
`tlay`	Grid coordinates in the temporal axis (corresponds to the order of the spatial `im`ages in `v`).
`W`	A copy of `window`, the spatial `owin` upon which the density is defined.

Author(s)

A.K. Redmond and T.M. Davies

Examples

```
require("abind")
m1 <- c(0.3,0.3,2)
m2 <- c(0.5,0.8,8)
m3 <- c(0.7,0.6,7)
v1 <- diag(c(0.01^2,0.01^2,1))
v2 <- diag(c(0.005,0.005,0.5))
v3 <- diag(c(0.005,0.005,0.5))
stg1 <- stgmix(mean=cbind(m1,m2,m3),
               vcv=abind(v1,v2,v3,along=3),
               window=toywin,tlim=c(1,10),
               p0=0.1,tres=64)
plot(stg1,log=TRUE)


mn <- matrix(c(0,0,0,-2,1,4,1,-2,8),nrow=3)
vr <- array(c(1,0,0,0,1,0,0,0,1,1,0,0.5,0,1,0,0.5,0,3,1,0,0,0,2,0,0,0,1),
            dim=c(3,3,3))
stg2 <- stgmix(mean=mn,vcv=vr,window=shp1,
               tlim=c(0,10),tres=50)
plot(stg2,fix.range=TRUE,sleep=0.1)
```

`stintegral` *Evaluate integral of a spatiotemporal object*

Description

Integrates an object of class `stim` or `stden`.

Usage

```
stintegral(x)
```

Arguments

x	The object of class stim or stden to be integrated.

Details

The integral is evaluated arithmetically as the sum of the product of the value of each voxel and the voxel area, for those voxels inside the relevant space-time window (i.e. ignoring NAs).

Value

A single numeric value giving the integral sought.

Author(s)

T.M. Davies

Examples

```

# 'stim' objects
require("abind")
m1 <- c(0.3,0.3,2)
m2 <- c(0.5,0.8,8)
m3 <- c(0.7,0.6,7)
v1 <- diag(c(0.01^2,0.01^2,1))
v2 <- diag(c(0.005,0.005,0.5))
v3 <- diag(c(0.005,0.005,0.5))
stg1 <- stgmix(mean=cbind(m1,m2,m3),vcv=abind(v1,v2,v3,along=3),
               window=toywin,tlim=c(0,10),p0=0.1,tres=64)
stg2 <- stgmix(mean=cbind(m1,m2,m3),vcv=abind(v1,v2,v3,along=3),
               window=toywin,tlim=c(0,10),p0=0.1,tres=64,int=200)
stintegral(stg1)
stintegral(stg2)

# 'sten' objects
require("sparr")
data(burk)
hlam <- OS.spattemp(burk$cases)
bden <- spattemp.density(burk$cases,h=hlam[1],lambda=hlam[2],sres=64,verbose=FALSE)
stintegral(bden)
```

stkey — *Spatiotemporal density generation via keyframe interpolation*

Description

Uses the supplied spatial pixel images and scalings to linearly interpolate the behaviour of the function over time, creating a trivariate density function in space-time.

Usage

```
stkey(start, stop, tlim, kf = NULL, tres = 64,
      kftimes = NULL, fscales = NULL, window = NULL)
```

Arguments

`start`	The spatial pixel `im`age corresponding to the spatial density at start of the time interval. May be unnormalised, the function internally rescales all supplied spatial images to integrate to 1.
`stop`	The pixel `im`age for the end of the time interval. Must be compatible with `start`, in that it is defined over the same spatial domain and is of identical resolution.
`tlim`	A numeric vector of length 2 representing the temporal window i.e. the time interval over which the interpolation takes place.
`kf`	A `solist` of the pixel `im`ages of the keyframes between `start` and `stop`. If supplied, each image must be compatible with `start` and `stop`. If unsupplied, the resulting interpolation is performed only on `start` and `stop`
`tres`	The resolution of the resulting array in the time dimension (default is 64).
`kftimes`	A vector of times that position the interim keyframes in `kf` between `tlim[1]` and `tlim[2]`. Ignored if `kf` = `NULL`. If unsupplied (`NULL`), but `kf` is present, the function simply positions the images of `kf` at evenly spaced time points in the `tlim` interval.
`fscales`	A numeric vector of unnormalised, relative point-intensity scales. This may be provided either as of `length(kf) + 2`, so the point intensities assinged to each frame in the order `c(start,<entries of kf>,stop)`, or of length `tres`. If unsupplied, each spatial frame is simply given equal weight. See 'Details'.
`window`	An object of class `owin` giving the polygonal spatial observational window on which the density is defined. If `NULL`, the polygon is simply obtained from the union of pixel values in the supplied images.

Details

This function interpolates in a pixel-wise fashion between the `im` objects supplied as `start` and `stop` (and `kf` if supplied), placing them as keyframes at the times in `tlim` (for `start` and `stop`) and `kftimes` (for the members of `kf`). The final result is rescaled such that its total integrated volume over the defined spatiotemporal domain is 1, yeilding a trivariate density function.

If `fscale` is a vector of length `tres`, each element will correspond to the **relative** overall scaling of one of the resulting interpolated pixel images. If it is of length `length(kf) + 2`, the scales will correspond to `start`, each keyframe in `kf` and `stop` in that order. The values in this argument are only interpretable in a relative sense: for example, with a single keyframe suppled to `kf` (in addition to the required `start` and `stop`), then `fscales = c(0.5,1,0.5)` has exactly the same effect on the final result as `fscales = c(1,2,1)`, and is interpreted as yielding a point density that reaches twice the concentration at the time of the supplied keyframe relative to the `start` and `stop` margins. Supplying `fscale` as a vector of length `tres` thus allows finer control over the relative point density over time, such as for the incorporation of harmonic seasonal variation.

Value

Like `stgmix`, an object of class `stim` giving the trivariate density. This is a list with six components:

`a`	The `xr` x `yr` x `tres` array of the specified density (where `xr` and `yr` are used here to denote the spatial resolution along the x- and y-axes; this is governed in `stkey` by the images initally supplied to `start` and `stop`).
`v`	A pixel `im`age version of `a`, provided as a `solist` of length `tres`, with each member being the spatial `im`age slice of the 3D density at each of the time-coordinate values.
`xcol`	Grid coordinates in the spatial x-axis (corresponds to each spatial `im`age in `v`).
`yrow`	Grid coordinates in the spatial y-axis (corresponds to each spatial `im`age in `v`).
`tlay`	Grid coordinates in the temporal axis (corresponds to the order of the spatial `im`ages in `v`).
`W`	A copy of `window`, the spatial `owin` upon which the density is defined.

Author(s)

A.K. Redmond and T.M. Davies

Examples

```
mn <- matrix(c(0,0,1,2,0.5,-1),nrow=2)
vr <- array(c(0.2,0,0,2,1,0,0,1,1,0.3,0.3,0.5),dim=c(2,2,3))
im1 <- sgmix(mn,c(1,2,1),shp1,p=c(0.4,0.3,0.3))
im2 <- sgmix(matrix(c(-3,0,0,-2,-1,2),nrow=2),c(3,1,1),shp1,p=c(0.4,0.3,0.3))
im3 <- sgmix(mn,vr,shp1,p0=0.1)

kf1 <- stkey(start=im1,stop=im2,tlim=c(5,20),window=shp1)
plot(kf1)

kf2 <- stkey(start=im1,stop=im1,tlim=c(0,15),kf=solist(im1,im1),kftimes=c(2,8),
             fscale=c(1,2,1.5,1),window=shp1)
plot(kf2,fix.range=TRUE)

kf3 <- stkey(start=im1,stop=im2,tlim=c(0,20),kf=solist(im1,im2),kftimes=c(8,12),
             fscale=c(1,2,2,1),window=shp1)
plot(kf3,fix.range=TRUE)
```

```
ff <- c(sin((1:64)/3)+1.5)
plot(ff,type="l")
kf4 <- stkey(start=im1,stop=im2,kf=solist(im3),kftimes=25,tlim=c(0,50),fscale=ff,window=shp1)
plot(kf4,fix.range=TRUE)
```

---

toywin *Toy Windows*

Description

Synthetic spatial windows for use in testing, simulations and demonstrations.

Usage

```
data(bx)
data(heart)
data(shp1)
data(shp2)
data(star)
data(toywin)
```

Format

Each of these is a single closed polygon of class `owin`.

- `bx` is a box on [-5,5]^2.
- `heart` is a heart, professing love for all things `spatstat`.
- `shp1` is shape of mystery.
- `shp2` is a slightly more symmetric shape of mystery.
- `star` is a star that shines brightly in even non-spatial contexts.
- `toywin` is the eponymous toy window used in publications e.g. Davies & Lawson (2019).

Details

These are lazy-loaded so may be called directly by name upon loading of `spagmix`.

Author(s)

A.K. Redmond and T.M. Davies

References

Davies, T.M. and Lawson, A.B. (2019), An evaluation of likelihood-based bandwidth selectors for spatial and spatiotemporal kernel estimates, *Journal of Statistical Computation and Simulation*, **89** 1131-1152.

Examples

```
oldpar <- par(mfrow=c(2,3))
plot(bx);axis(1);axis(2)
plot(heart);axis(1);axis(2)
plot(shp1);axis(1);axis(2)
plot(shp2);axis(1);axis(2)
plot(star);axis(1);axis(2)
plot(toywin);axis(1);axis(2)
par(oldpar)
```

unify.owin *Spatial window unit rescaler*

Description

Rescales any `owin` to fall inside the unit square.

Usage

```
unify.owin(W)
```

Arguments

`W`	An object of class `owin` giving the spatial window to be transformed.

Details

This function is a simple wrapper for `affine` deployed to rescale a supplied `owin` to fall inside the unit square.

Value

The rescaled `owin`.

Examples

```
W <- Window(chorley)
U <- unify.owin(W)

oldpar <- par(mfrow=c(1,2))
plot(W,axes=TRUE)
plot(U,axes=TRUE)
par(oldpar)
```

Index