Package: lacunaritycovariance (via r-universe)

August 30, 2024

Type Package

Title Gliding Box Lacunarity and Other Metrics for 2D Random Closed Sets

Version 1.1-7

Date 2023-11-02

Author Kassel Liam Hingee

Maintainer Kassel Liam Hingee <kassel.hingee@gmail.com>

Description Functions for estimating the gliding box lacunarity (GBL), covariance, and pair-correlation of a random closed set (RACS) in 2D from a binary coverage map (e.g. presence-absence land cover maps). Contains a number of newly-developed covariance-based estimators of GBL (Hingee et al., 2019) <doi:10.1007/s13253-019-00351-9> and balanced estimators, proposed by Picka (2000) <http://www.jstor.org/stable/1428408>, for covariance, centred covariance, and pair-correlation. Also contains methods for estimating contagion-like properties of RACS and simulating 2D Boolean models. Binary coverage maps are usually represented as raster images with pixel values of TRUE, FALSE or NA, with NA representing unobserved pixels. A demo for extracting such a binary map from a geospatial data format is provided. Binary maps may also be represented using polygonal sets as the foreground, however for most computations such maps are converted into raster images. The package is based on research conducted during the author's PhD studies.

License GPL (>= 2)

URL https://github.com/kasselhingee/lacunaritycovariance

Depends spatstat (>= 2.0-0)

Imports RcppRoll, spatstat.explore (>= 3.0-3), spatstat.geom, spatstat.random

Suggests cubature, sf, terra, testthat

ByteCompile true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/kasselhingee/lacunaritycovariance

RemoteRef HEAD

RemoteSha 12c59da1134bef61fc3751bcbdedba6e65cf17b1

Contents

`cencovariance` *Centred covariance estimation*

Description

This function estimates the centred covariance of a stationary RACS. Available estimators are the plug-in moment centred covariance estimator, two 'balanced' estimators suggested by Picka (2000), and a third 'balanced' estimator inspired by one of Picka's pair-correlation estimators.

Usage

```
cencovariance(
  xi,
  obswin = NULL,
  setcov_boundarythresh = NULL,
  estimators = "all",
  drop = FALSE
)

cencovariance.cvchat(
  cvchat,
  cpp1 = NULL,
  phat = NULL,
  setcov_boundarythresh = NULL,
  estimators = "all",
  drop = FALSE
)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied.
`obswin`	If `xi` is an `owin` object then `obswin` is an `owin` object that specifies the observation window.
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the covariance is given a value of NA.
`estimators`	A list of strings specifying estimators to use. See details. `estimators = "all"` will select all available estimators.
`drop`	If TRUE and one estimator selected then the returned value will be a single `im` object and not a list of `im` object.
`cvchat`	The plug-in moment estimate of covariance as an `im` object. Typically created with `plugincvc`.
`cpp1`	Picka's reduced window estimate of coverage probability as an `im` object - used in improved (balanced) covariance estimators. Can be generated using `cppicka`.
`phat`	The usual estimate of coverage probability, which is the observed foreground area in `xi` divided by the total area of the observation window. See `coverageprob` for more information.

Details

The centred covariance of a stationary RACS is

$$\kappa(v) = C(v) - p^2.$$

The estimators available are (see (Section 3.4, Hingee, 2019) for more information):

- `plugin` the plug-in moment centred covariance estimator
- `mattfeldt` an estimator inspired by an 'intrinsically' balanced pair-correlation estimator from Picka (1997) that was later studied in an isotropic situation by Mattfeldt and Stoyan (Mattfeldt and Stoyan, 2000)
- `pickaint` Picka's 'intrinsically' balanced centred covariance estimator (Picka, 2000).
- `pickaH` Picka's 'additively' balanced centred covariance estimator (Picka, 2000).

Currently computes centred covariance using `racscovariance`.

Value

If `drop = TRUE` and only one estimator is requested then a `im` object containing the centred covariance estimate is returned. Otherwise a named `imlist` of `im` objects containing the centred covariance estimates for each requested estimator.

Functions

- `cencovariance()`: Centred covariance estimates from a binary map.
- `cencovariance.cvchat()`: Generates centred covariances estimates from a plug-in moment estimate of covariance, Picka's reduced window estimate of coverage probability, and the plug-in moment estimate of coverage probability. If these estimates already exist, then `cencovariance.cvchat` saves significant computation time over `cencovariance`.

Author(s)

Kassel Liam Hingee

References

Hingee, K.L. (2019) *Spatial Statistics of Random Closed Sets for Earth Observations*. PhD: Perth, Western Australia: University of Western Australia. Submitted.

Mattfeldt, T. and Stoyan, D. (2000) Improved estimation of the pair correlation function of random sets. *Journal of Microscopy*, 200, 158-173.

Picka, J.D. (1997) *Variance-Reducing Modifications for Estimators of Dependence in Random Sets*. Ph.D.: Illinois, USA: The University of Chicago.

Picka, J.D. (2000) Variance reducing modifications for estimators of standardized moments of random sets. *Advances in Applied Probability*, 32, 682-700.

Examples

```
xi <- heather$coarse
obswin <- Frame(xi)
cencovariance(xi, obswin, estimators = "all")
```

`contagdiscstate` *Disc State Contagion*

Description

Calculates the disc-state contagion landscape metric as described in (Hingee, 2016; Hingee, 2019). The disc-state contagion landscape metric describes the entropy (mixing) between four possible states of a disc:

1. the disc is completely contained in Ξ
2. the disc does not intersect Ξ
3. the centre of the disc is in Ξ but the disc is not contained in Ξ
4. the disc intersects Ξ but the centre is outside Ξ

Disc-state contagion is a function of the disc radius.

The main difference to classical contagion (O'Neill, 1988) is that disc-state contagion is based on the spherical contact distribution instead of pixel neighbours. One impact of this design is that the distance with which to quantify the mixing between Ξ and the background may be chosen by the user by choosing the disc radius (for classical contagion this distance is fixed by the map resolution).

Note: to create `fv` objects the function copies about 20 lines of code from the **spatstat** collection of packages.

Usage

```
contagdiscstate(XiH, XicH, p, normalise = FALSE)
```

Arguments

`XiH`	Conditional spherical contact distribution function for Ξ. Typically this is an `fv` object but could also be a vector of values. In applications `XiH` would likely be estimated from a binary map using `Hest` in **spatstat.explore**.
`XicH`	Conditional spherical contact distribution for the complement of Ξ. This is called the Conditional Core Probability in Hingee 2016. Typically this is an `fv` object but could also be a vector of values. In applications `XiH` would likely be estimated from a binary map using `Hest` in **spatstat.explore**.
`p`	The coverage probability of Ξ. Can be estimated from binary maps using `coverageprob`.
`normalise`	Optional. If TRUE `contagdiscstate` normalises the results so that all RACS return a value between 0 and 1. Default is FALSE.

Details

`XiH` should be a function of radius that contains (estimates of) the probability of a disc of radius r not intersecting Ξ if the disc's centre is not in Ξ

$$\texttt{XiH}(r) = P(B_r(x) \subseteq \Xi^c | x \in \Xi^c).$$

Similarly XicH should be (an estimate of) the probability of a disc being fully contained in Ξ given its centre is in Ξ

$$\texttt{XicH}(r) \approx P(B_r(x) \subseteq \Xi | x \in \Xi).$$

These can both be obtained using Hest in **spatstat**. For XicH take care to apply Hest to the complement of Ξ with the observation window W.

If normalise is TRUE then the result is divided by $-2ln(2)$ and increased by 1 so that contagion will always be between 0 and 1.

Value

If XiH is an fv object then an fv object is returned. If XiH is a vector then the returned object is a vector the same length as XiH with each element corresponding to the contagion at each r value of XiH.

References

Hingee, K.L. (2016) Statistics for patch observations. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* pp. 235-242. Prague: ISPRS.

Hingee, K.L. (2019) *Spatial Statistics of Random Closed Sets for Earth Observations*. PhD: Perth, Western Australia: University of Western Australia. Submitted.

O'Neill, R.V., Krummel, J.R., Gardner, R.H., Sugihara, G., Jackson, B., DeAngelis, D.L., et al. (1988) Indices of landscape pattern. *Landscape Ecology*, 1, 153-162.

Examples

```
xi <- heather$coarse
obswindow <- Frame(heather$coarse)
p <- coverageprob(xi, Frame(xi))
xiH <- Hest(xi, W = obswindow) #Sph. Contact Distrution Estimate
xicH <- Hest(complement.owin(xi), W = obswindow) #Conditional Core Prob. Estimate

contagion <- contagdiscstate(xiH, xicH, p)
```

---

contagpixelgrid *Pixel Adjacency Contagion*

Description

Function for calculating the classic pixel-adjacency contagion landscape metric from a binary map (O'Neill, 1988).

Usage

```
contagpixelgrid(xi, obswin, normalise = FALSE)

adjacency(xi, obswin = NULL)
```

Arguments

xi	A raster binary map (as an im object) or as a foreground set (as an owin object). In the latter case the observation window, obswin, must be supplied. See lacunaritycovariance-package for details. If xi is an owin object it must be of mask type.
obswin	If xi is an owin object then obswin is an owin object that specifies the observation window.
normalise	If TRUE will divide result by $2ln(2)$ and add 1 to make contagion between 0 and 1 for all binary maps.

Details

The unnormalised contagion landscape metric of categorical map is defined as

$$\sum_i \sum_j Qijln(Qij),$$

where Qij is the probability of randomly selected adjacent pixels being in class i and class j respectively, and m is the number of classes.

Here $m = 2$ as xi is a binary map and we have defined 'adjacent' pixels using the 4-neighbourhood regime.

Contagion is calculated from an adjacency matrix created using the function adjacency. The adjacency matrix is a 2 by 2 table containing the number of pairs of neighbouring pixels (where order matters) such that:

	Second pixel in xi	Second pixel not in xi
First pixel in xi	-	-
First pixel not in xi	-	-

Value

The computed pixel-adjacency contagion value. If normalise is TRUE then the value will be between 0 and 1. Otherwise the value will be negative.

Functions

- contagpixelgrid(): Pixel-adjacency contagion landscape metric of a binary map.
- adjacency(): Calculates the adjacency matrix used in the pixel contagion

Warning

Will fail if map is either all foreground or all background.

References

O'Neill, R.V., Krummel, J.R., Gardner, R.H., Sugihara, G., Jackson, B., DeAngelis, D.L., et al. (1988) Indices of landscape pattern. *Landscape Ecology*, 1, 153-162.

Examples

```
xi <- heather$coarse
obswin <- owin(xrange = c(0,7),yrange=c(0,16))
adjmat <- adjacency(xi,obswin)
pixeladjcontagion <- contagpixelgrid(xi,obswin)
```

coverageprob *Estimate the coverage probability of a stationary RACS*

Description

Computes the proportion of the observation window that is foreground, which is the usual estimate for the coverage probability of a stationary RACS from a binary map.

Usage

```
coverageprob(xi, obswin = NULL)

coveragefrac(xi, obswin = NULL)

cp(xi, obswin = NULL)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied.
`obswin`	The window of observation (not necessarily rectangular) also as an `owin` object.

Details

The coverage probability of a stationary RACS is the probability that an arbitrary point is covered by the RACS. Given a binary map, `xi`, of a realisation of stationary RACS Ξ in a window W, this function computes the fraction of W covered by foreground, which is an estimate of the coverage probability. See (Chiu et al., 2013, Section 6.4.2) for more details.

If `xi` is an `im` object then `xi` must be an image of 1s, 0s and NAs representing inside the set, outside the set and outside the observation window respectively. `coverageprob` will not accept a `obswin` argument if `xi` is an `im` object.

Value

An estimate of the coverage probability

Author(s)

Kassel Liam Hingee

References

Chiu, S.N., Stoyan, D., Kendall, W.S. and Mecke, J. (2013) Stochastic Geometry and Its Applications, 3rd ed. Chichester, United Kingdom: John Wiley & Sons.

Examples

```
xi <- heather$coarse
obswindow <- Frame(heather$coarse)
cp <- coverageprob(xi, obswindow)
```

---

cppicka *Picka's Reduced Window Estimator of Coverage Probability*

Description

This function provides estimates of coverage probability from subsets of the observation window, which are a key component of balanced estimators of covariance, centred covariance, pair-correlation and gliding box lacunarity.

Usage

```
cppicka(xi, obswin = NULL, setcov_boundarythresh = NULL)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied.
`obswin`	If `xi` is an `owin` object then `obswin` is an `owin` object that specifies the observation window.
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the returned pixel value is NA.

Details

The plug-in moment covariance estimator (`plugincvc`) uses less of the observation window than the usual coverage probability estimators. Picka (1997, 2000) created new 'balanced' estimators of centred covariance and pair-correlation that accounted for this difference. A key component of Picka's estimators is an estimate of the coverage probability from the subregion of the binary map that is the intersection between W and W shifted by vector v, where W is the observation window (p.~687, Picka, 2000). If we treat X and W as indicator functions representing the foreground and observation window respectively, this coverage probability estimator used by Picka is

$$\frac{\int X(u)W(u)W(u-v)du}{\int W(u)W(u-v)du}.$$

`cppicka` produces these estimates for an array of vectors v using fast Fourier transforms.

Value

An `im` object. Pixel values correspond to estimates of the coverage probability from the subregion of the observation window, W, that is the intersection of W and W shifted by vector v, where v is the pixel location.

Author(s)

Kassel Liam Hingee

References

Picka, J.D. (1997) *Variance-Reducing Modifications for Estimators of Dependence in Random Sets*. Ph.D.: Illinois, USA: The University of Chicago.

Picka, J.D. (2000) Variance reducing modifications for estimators of standardized moments of random sets. *Advances in Applied Probability*, 32, 682-700.

Examples

```
xi <- heather$coarse
obswindow <- Frame(heather$coarse)
cp <- coverageprob(xi, obswindow)
cpp1 <- cppicka(xi, obswindow)
```

`gbl` *Gliding box lacunarity estimation using all estimators*

Description

Estimates gliding box lacunarity (GBL) for square boxes using all estimators described in (Hingee et al., 2017). It calls the functions `gblc`, `gblg`, `gblcc` and `gblemp`.

Usage

```
gbl(
  xi,
  boxwidths,
  estimators = c("GBLcc.pickaH"),
  obswin = NULL,
  includenormed = FALSE,
  setcov_boundarythresh = 1e-06
)

gbl.cvchat(
  boxwidths,
 estimators = c("GBLg.mattfeldt", "GBLg.pickaint", "GBLg.pickaH", "GBLcc.mattfeldt",
    "GBLcc.pickaint", "GBLc"),
  phat = NULL,
  cvchat = NULL,
  cpp1 = NULL
)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied.
`boxwidths`	A list of box widths
`estimators`	A vector of estimator names - see details for possible names. `estimators = "all"` will select all estimators.
`obswin`	If `xi` is an `owin` object then `obswin` is an `owin` object that specifies the observation window.
`includenormed`	A logical value. If TRUE then GBL estimates normalised by the GBL values at zero will be included in a returned list of `fv` objects
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the covariance is given a value of NA. If NULL is supplied (default) then `setcov_boundarythresh` is set to 1E-6.
`phat`	The fraction of foreground area in the observation window, which is the usual estimator of coverage probability given by `coverageprob`.
`cvchat`	The plug-in moment covariance estimate (often from `plugincvc`).
`cpp1`	Picka's estimate of coverage probability for subsets of the observation window. See `cppicka`.

Details

As empirical GBL is one of the GBL estimators available through this function, non-square boxes are not allowed. To estimate GBL for non-square boxes use gblcc or gblg directly.

If `xi` is an `owin` object then `obswin` and `xi` are converted into an `im` object using `as.im`

The estimators available are

- `"GBLc"` The unmodified (unbalanced) covariance estimator provided by `gblc`
- `"GBLemp"` Empirical gliding box lacunarity (Allain and Cloitre, 1991). Calls `gblemp`
- `"GBLg.mattfeldt"` See help for `gblg` and `paircorr`
- `"GBLg.pickaint"` See help for `gblg` and `paircorr`
- `"GBLg.pickaH"` See help for `gblg` and `paircorr`
- `"GBLcc.mattfeldt"` See help for `gblcc`
- `"GBLcc.pickaint"` See help for `gblcc`
- `"GBLcc.pickaH"` See help for `gblcc`

The default, GBLcc.pickaH, is a method based on centred covariance. Centred covariance approaches zero for large vectors, and are thus easier to integrate with the set covariance of the boxes.

The set covariance of the boxes is computed empirically using **spatstat**'s `setcov` function, which converts B into a binary pixel mask using `as.mask` defaults. Computation speed can be increased by setting a small default number of pixels, `npixel`, in **spatstat**'s global options (accessed through `spatstat.options`), however fewer pixels also decreases the accuracy of the GBL computation.

Value

An `fv` object.

Functions

- `gbl()`: computes GBL estimates from a binary map.
- `gbl.cvchat()`: computes covariance-based estimator of GBL from the plug-in moment estimate of covariance, Picka's reduced window coverage probability estimates (see `cppicka`) and the usual coverage probability estimate, `phat`.

References

Allain, C. and Cloitre, M. (1991) Characterizing the lacunarity of random and deterministic fractal sets. *Physical Review A*, 44, 3552-3558.

Hingee K, Baddeley A, Caccetta P, Nair G (2019). Computation of lacunarity from covariance of spatial binary maps. *Journal of Agricultural, Biological and Environmental Statistics*, 24, 264-288. DOI: 10.1007/s13253-019-00351-9.

Examples

```
xi <- as.im(heather$coarse, value = TRUE,
            na.replace = FALSE)
boxwidths <- seq(1, 10, by = 0.5)

# reduce resolution in setcov() for faster (less accurate) computation
oldopt <- spatstat.options()
spatstat.options("npixel" = 2^5)

gblests <- gbl(xi, boxwidths, estimators = "GBLcc.pickaH")
spatstat.options(oldopt)
```

`gblc` *Gliding box lacunarity estimator using plug-in moment covariance estimator*

Description

Can be used to estimate the gliding box lacunarity (GBL) of a stationary RACS from a binary map using the plug-in moment covariance covariance estimator (Hingee et al., 2019). It can also calculate the GBL of a RACS from a given covariance function and coverage probability.

Usage

```
gblc(
  boxes,
  covariance = NULL,
  p = NULL,
  xiim = NULL,
  integrationMethod = "cubature"
)
```

Arguments

`boxes`	Either a list of side lengths for square boxes or a list of `owin` objects of any shape.
`covariance`	A `im` object containing the covariance function
`p`	The coverage probability. Typically estimated by the fraction of the observation window covered by the set of interest.
`xiim`	A binary coverage map as an `im` object. `xiim` must have values of either 1, 0 or NA; 1 denotes inside the RACS, 0 denotes outside, and NA denotes unobserved.
`integrationMethod`	The integration method used by `innerprod.im()`.

Details

Computes a numerical approximation of

$$\int \gamma_B(v)C(v)dv/(p^2|B|^2).$$

where B is each of the sets (often called a box) specified by `boxes`, γ_B is the set covariance of B, $|B|$ is the area of B, p is the coverage probability of a stationary RACS, and $C(v)$ is the covariance of a stationary RACS. This can be used to compute the GBL from model parameters by passing `gblc` the covariance and coverage probability of the model.

The set covariance of B is computed empirically using **spatstat**'s `setcov` function, which converts B into a binary pixel mask using `as.mask` defaults. Computation speed can be increased by setting a small default number of pixels, `npixel`, in **spatstat**'s global options (accessed through `spatstat.options`), however fewer pixels also decreases the accuracy of the GBL computation.

The default method of integration for the above integral is `cubature::cubintegrate()` from the **cubature** package. The '`harmonisesum`' method is known to produce numerical artefacts (Section 6.2 of (Hingee et al., 2019))

If a binary map is supplied then p and $C(v)$ are estimated using the usual coverage probability estimator and the plug-in moment covariance estimator, respectively (see `coverageprob` and `plugincvc`).

Value

If `boxes` is a list of numerical values then GBL is estimated for square boxes with side length given by `boxes`. The returned object is then an `fv` object containing estimates of GBL, box mass variance and box mass mean. If `boxes` is a list of `owin` objects then `gblc` returns a dataframe with columns corresponding to estimates of GBL, box mass variance and box mass mean.

Note if `NA` or `NaN` values in the `covariance` object are used then `gblc` will return `NA` or `NaN`.

References

Hingee K, Baddeley A, Caccetta P, Nair G (2019). Computation of lacunarity from covariance of spatial binary maps. *Journal of Agricultural, Biological and Environmental Statistics*, 24, 264-288. DOI: 10.1007/s13253-019-00351-9.

Examples

```
xi <- heather$coarse

# reduce resolution in setcov() for faster (less accurate) computation
oldopt <- spatstat.options()
spatstat.options("npixel" = 2^5)

covar <- plugincvc(xi, Frame(xi))
p <- area(xi) / area(Frame(xi))
sidelengths <- seq(0.3, 14, by = 1)

# compute GBL estimate for square boxes from estimated covariance
gblest <- gblc(sidelengths, covar, p)

# compute GBL estimate for boxes that are discs
discboxes <- lapply(sidelengths / 2, disc)
discgbls <- gblc(discboxes, covar, p)

# compute GBL estimates from binary map
xiim <- as.im(xi, na.replace = 0)
gblest <- gblc(sidelengths, xiim = xiim)

spatstat.options(oldopt)
```

`gblcc` *Centred covariance based estimates of gliding box lacunarity*

Description

Estimates the gliding box lacunarity (GBL) of a stationary RACS using centred covariance estimates (Hingee et al., 2017). The centred covariance and coverage probability can be provided or estimated from binary map.

Usage

```
gblcc(
  boxes,
  cencovar = NULL,
  p = NULL,
  xiim = NULL,
  estimator = "pickaH",
  integrationMethod = "harmonisesum"
)

gblcc.inputcovar(boxes, cencovar, p, integrationMethod = "harmonisesum")
```

Arguments

`boxes`	Either a list of side lengths for square boxes or a list of `owin` objects of any shape.
`cencovar`	A `im` object containing the centred covariance function
`p`	The coverage probability. Typically estimated by the fraction of the observation window covered by the set of interest.
`xiim`	An observation of a stationary RACS as an `im` object. `xiim` must have values of either 1, 0 or NA; 1 denotes inside the RACS, 0 denotes outside, and NA denotes unobserved.
`estimator`	If an observation `xiim` is passed then `estimator` will select the balancing method that `ccvc` uses to estimate the centred covariance.
`integrationMethod`	The integration method used by `innerprod.im()`. Default is 'harmonisesum' due centred covariance approaching zero for large vectors.

Details

If we denote the estimated centred covariance by $\hat{\kappa}(v)$ and coverage probability $\hat{p}$ then the estimate of GBL is

$$1 + \frac{1}{\hat{p}^2 |B|^2} \int \gamma_B(v) \hat{\kappa}(v) dv,$$

where B is each of the sets (often called a box) specified by `boxes`, γ_B is the set covariance of B, $|B|$ is the area of B, p is the coverage probability of a stationary RACS.

The set covariance of B is computed empirically using **spatstat**'s `setcov` function, which converts B into a binary pixel mask using `as.mask` defaults. Computation speed can be increased by setting a small default number of pixels, `npixel`, in **spatstat**'s global options (accessed through `spatstat.options`), however fewer pixels also decreases the accuracy of the GBL computation.

Value

If `boxes` is a list of numerical values then GBL is estimated for square boxes with side length given by `boxes`. The returned object is then an `fv` object containing estimates of GBL, box mass variance and box mass mean.

If boxes is a list of owin objects then gblcc returns a dataframe of with columns corresponding to estimates of GBL, box mass variance and box mass mean. Note if NA or NaN values in the covariance object are used then gblc will return NA or NaN instead of an GBL value.

Functions

- gblcc.inputcovar(): GBL estimates from already estimated centred covariance.

References

Hingee K, Baddeley A, Caccetta P, Nair G (2019). Computation of lacunarity from covariance of spatial binary maps. *Journal of Agricultural, Biological and Environmental Statistics*, 24, 264-288. DOI: 10.1007/s13253-019-00351-9.

Examples

```
xi <- heather$coarse
cencovar <- cencovariance(xi, obswin = Frame(xi), estimators = c("pickaH"), drop = TRUE)
p <- area(xi) / area(Frame(xi))
sidelengths <- seq(0.3, 14, by = 1)

# reduce resolution in setcov() for faster (less accurate) computation
oldopt <- spatstat.options()
spatstat.options("npixel" = 2^5)

# compute GBL estimate for square boxes from estimated centred covariance
gblccest <- gblcc(sidelengths, cencovar, p)

# compute GBL estimate for boxes that are discs
discboxes <- lapply(sidelengths / 2, disc)
discgbls <- gblcc(discboxes, cencovar, p)

# compute GBL estimates from binary map
xiim <- as.im(xi, na.replace = 0)
gblccest <- gblcc(sidelengths, xiim = xiim, estimator = "pickaH")

spatstat.options(oldopt)
```

---

gblemp *Empirical Gliding Box Lacunarity*

Description

Calculates empirical gliding box lacunarity of a binary map, which was proposed by Allain and Cloitre (1991).

Usage

```
gblemp(boxwidths, xiim, obswin = Frame(xiim))

gbltrad(boxwidths, xiim, obswin = Frame(xiim))
```

Arguments

`boxwidths`	A list of suggested box widths in the same units as `xiim`. Note the actual box widths used by `gblemp` will be the closest multiple of an odd number of pixel widths.
`xiim`	An image of pixels valued either `0`, `1` or `NA`. `NA` valued pixels are assumed to be outside the observation window.
`obswin`	Optional observation window. The observation window used for the estimator will be the intersection of `obswin` and the pixels that are not `NA` in `xiim`.

Details

Calculates empirical gliding box lacunarity (Allain and Cloitre, 1991) for a given range of square box sizes,

$$1 + Var(area(B.xi))/E[area(B.xi)]^2,$$

where B is a box that has a random location in the observation window and $area(B.xi)$ is the (random) area of the foreground in B. This is an estimate of the gliding box lacunarity of a RACS (Hingee et al., 2017).

The algorithm uses the pixel locations in `xiim` as an array of box centre locations to compute the mean and variance of the area in a random box of a given size. Locations where the box is not completely within the observation window are ignored.

Value

An `fv` object containing empirical GBL, variance of the area in the box and mean of the area in the box. The box widths (labelled `s`) are always odd multiples of the pixel width.

Functions

- `gbltrad()`: An alias of `gblemp` used in past versions of this package. This alias may be removed in future versions.

WARNING

The box side lengths are rounded such that they are an odd number of pixels across. `gblemp` uses the `roll_sum` function in **RcppRoll** to operate, so **RcppRoll** must be installed to run `gblemp`.

References

Allain, C. and Cloitre, M. (1991) Characterizing the lacunarity of random and deterministic fractal sets. *Physical Review A*, 44, 3552-3558.

Hingee K, Baddeley A, Caccetta P, Nair G (2019). Computation of lacunarity from covariance of spatial binary maps. *Journal of Agricultural, Biological and Environmental Statistics*, 24, 264-288. DOI: 10.1007/s13253-019-00351-9.

Examples

```
xiim <- as.im(heather$coarse, na.replace = 0)
boxwidths <- seq(0.2, 14, by = 0.2) #in units of xiim
gblest <- gblemp(boxwidths, xiim)
```

gblg *Pair-correlation based estimates of gliding box lacunarity*

Description

Estimates the gliding box lacunarity (GBL) of a stationary RACS by estimating pair-correlation from a binary map (Hingee et al., 2017). It can also calculate the GBL of a RACS from a provided pair-correlation function.

Usage

```
gblg(boxes, paircorr = NULL, xiim = NULL, integrationMethod = "cubature")
```

Arguments

boxes	Either a list of side lengths for square boxes or a list of owin objects of any shape.
paircorr	A im object containing the pair-correlation function
xiim	An observation of a stationary RACS as an im object. xiim must have values of either 1, 0 or NA; 1 denotes inside the RACS, 0 denotes outside, and NA denotes unobserved.
integrationMethod	The integration method used by innerprod.im().

Details

If we denote the estimated pair-correlation by $\hat{g}(v)$ then the estimate of GBL is

$$\frac{1}{|B|^2}\int \gamma_B(v)\hat{g}(v)dv,$$

where B is each of the sets (often called a box) specified by boxes, γ_B is the set covariance of B, $|B|$ is the area of B, p is the coverage probability of a stationary RACS. This can be used to compute the GBL from model parameters by passing gblc the covariance and coverage probability of the model.

If the xiim argument to gblg is used then pair correlation is estimated from xiim using paircorr and the pickaH estimator.

The set covariance of B is computed empirically using **spatstat**'s `setcov` function, which converts B into a binary pixel mask using `as.mask` defaults. Computation speed can be increased by setting a small default number of pixels, `npixel`, in **spatstat**'s global options (accessed through `spatstat.options`), however fewer pixels also decreases the accuracy of the GBL computation.

The default integration method for this function uses `cubature::cubintegrate()` from the **cubature** package. The 'harmonisesum' integration method is known to produce numerical artefacts (Section 6.2 of (Hingee et al., 2019))

Value

If `boxes` is a list of numerical values then GBL is estimated for square boxes with side length given by `boxes`. The returned object is then an `fv` object containing estimates of GBL. If `boxes` is a list of `owin` objects then `gblg` returns a dataframe of with columns corresponding to estimates of GBL.

Note that if any values in the `paircorr` object needed for `gblg` are `NA` or `NaN` then `gblg` will return `NA` or `NaN`, respectively.

References

Hingee K, Baddeley A, Caccetta P, Nair G (2019). Computation of lacunarity from covariance of spatial binary maps. *Journal of Agricultural, Biological and Environmental Statistics*, 24, 264-288. DOI: 10.1007/s13253-019-00351-9.

Examples

```
xi <- as.im(heather$coarse, na.replace = 0, eps = 4 * heather$coarse$xstep)
sidelengths <- seq(0.3, 14, by = 3)

# reduce resolution in setcov() for faster (less accurate) computation
oldopt <- spatstat.options()
spatstat.options("npixel" = 2^4)

# compute GBL estimates from binary map
xiim <- as.im(xi, na.replace = 0)
gblgest <- gblg(sidelengths, xiim = xiim)

spatstat.options(oldopt)
```

`innerprod.im` *Inner Product of Two Functions Represented as Images*

Description

Given two functions, f and g, that map from 2D space to 1D, and values of f and g represented as `im` objects then `innerprod.im` computes the (function space) inner product

$$\int f(v)g(v)dv.$$

Usage

```
innerprod.im(
  A,
  B,
  outsideA = NA,
  outsideB = NA,
  na.replace = TRUE,
  method = "cubature"
)
```

Arguments

`A`	An `im` object containing function values representing function f.
`B`	An `im` object containing function values representing function g.
`outsideA`	The value of f outside the domain of `A`. Typically will be 0 or NA. Default is NA.
`outsideB`	The value of g outside the domain of `B`. Typically will be `0` or `NA`. Default is `NA`.
`na.replace`	Logical. If TRUE NA values in `A` and `B` are replaced by `outsideA` and `outsideB`, respectively. This allows the cubature integration to be performed (roughly) twice as quickly.
`method`	Either "cubature" or "harmonisesum". The former uses `cubature::cubintegrate()`, the latter harmonises the images using `spatstat.geom::as.mask()` and sums the product of `A` and `B`.

Details

This function uses the package **cubature** to integrate the multiple of the two images. If **cubature** is not available then it harmonises the two input images, multiplies them together and returns the integral of the resulting image. `outsideA` and `outsideB` are used to determine result if the inner product requires values outside the domain of A or B. For example if `outsideA=0` and the domain of `B` is larger than `A`'s domain then the inner product can still be computed. However if `A` is `NA` outside (e.g. not known/not provided) and the domain of `B` is larger than `A`'s domain then the inner product can not be computed and the returned value is `NA`

The `harmonisesum` method appears to perform well when used by `gblcc()`, but produces numerical artefacts when used by `gblc()` and `gblg()`. The `cubature` method takes longer to compute and is more accurate for functions (A or B) that are non-zero for large vectors. This makes it suitable for use by `gblc()` and `gblg()`.

Value

If the inner product can be computed then returns sum(`A` * `B`), otherwise returns `NA`.

Examples

```
xi <- heather$coarse
covar <- plugincvc(xi, Frame(xi))
B <- setcov(square(1))
innerprod.im(covar, B, outsideB = 0)
```

`isbinarymap` *Test if an* `im` *object is a binary map*

Description

Tests whether `xi` is a binary map. The pixel values must be of logical type (`TRUE`, `FALSE` and `NA` only), or numerical (1, 0 or `NA`).

Usage

```
isbinarymap(xi, requiretrue = FALSE)
```

Arguments

`xi`	an image object
`requiretrue`	Logical. If TRUE then `isbinarymap` will error if xi is NOT a binary map.

Value

Logical value. `TRUE` if `xi` is a binary map. Otherwise `FALSE`. If `requiretrue` = `TRUE` and `xi` is not a binary map then an error will occur.

Examples

```
# The following return TRUE
isbinarymap(as.im(heather$coarse, na.value = 0))
isbinarymap(as.im(heather$coarse, na.value = FALSE, value = TRUE))

# The following return FALSE
isbinarymap(as.im(heather$coarse, na.value = 0.2, value = 1))
isbinarymap(as.im(heather$coarse, na.value = 0, value = 1.5))
```

`paircorr` *Balanced estimation of pair-correlation.*

Description

Estimates the pair-correlation function of a stationary RACS. The plug-in moment pair-correlation estimator and three 'balanced' estimators suggested by Picka (2000) are available.

Usage

```
paircorr(
  xi,
  obswin = NULL,
  setcov_boundarythresh = NULL,
  estimators = "all",
  drop = FALSE
)

paircorr.cvchat(
  cvchat,
  cpp1 = NULL,
  phat = NULL,
  estimators = "all",
  drop = FALSE
)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied. See `lacunaritycovariance-package` for details.
`obswin`	If `xi` is an `owin` object then `obswin` is an `owin` object that specifies the observation window.
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the covariance is given a value of NA.
`estimators`	A list of strings specifying estimators to use. See details. `estimators = "all"` will select all available estimators.
`drop`	If TRUE and one estimator selected then the returned value will be a single `im` object and not a list of `im` object. `estimators = "all"` will select all inbuilt estimators. See details.
`cvchat`	The plug-in moment estimate of covariance as an `im` object. Typically created with `plugincvc`.
`cpp1`	Picka's reduced window estimate of coverage probability as an `im` object - used in improved (balanced) covariance estimators. Can be generated using `cppicka`.
`phat`	The plug-in moment estimate of coverage probability, which is the observed foreground area in `xi` divided by the total area of the observation window. See `coverageprob` for more information.

Details

The pair-correlation of a stationary RACS is

$$g(v) = C(v)/p^2.$$

The estimators available are (see (Hingee, 2019) for more information):

- `plugin` the plug-in moment pair-correlation estimator which is $Chat(v)/(phat^2)$, where $Chat$ and $phat$ are the plug-in moment estimate of covariance and the usual estimate of coverage probability, respectively.
- `mattfeldt` an 'intrinsically' balanced pair-correlation estimator suggested by Picka (1997). A similar isotropic pair-correlation estimator was later studied by Mattfeldt and Stoyan (2000).
- `pickaint` Picka's 'intrinsically' balanced pair-correlation estimator (Picka, 2000).
- `pickaH` Picka's 'additively' balanced pair-correlation estimator (Picka, 2000).

Value

If `drop = TRUE` and a single estimator is requested then an `im` object containing the pair-correlation estimate is returned. Otherwise a named `imlist` of `im` objects containing the pair-correlation estimates for each requested estimator.

Functions

- `paircorr()`: Estimates pair-correlation from a binary map.
- `paircorr.cvchat()`: Generates pair-correlation estimates from the plug-in moment estimates of covariance, Picka's reduced window estimate of coverage probability, and the coverage fraction (which is an unbiased estimate of the coverage probability). If these estimates already exist then `paircorr.cvchat` can save significant computation time.

Author(s)

Kassel Liam Hingee

References

Hingee, K.L. (2019) *Spatial Statistics of Random Closed Sets for Earth Observations*. PhD: Perth, Western Australia: University of Western Australia. Submitted.

Mattfeldt, T. and Stoyan, D. (2000) Improved estimation of the pair correlation function of random sets. *Journal of Microscopy*, 200, 158-173.

Picka, J.D. (1997) *Variance-Reducing Modifications for Estimators of Dependence in Random Sets*. Ph.D.: Illinois, USA: The University of Chicago.

Picka, J.D. (2000) Variance reducing modifications for estimators of standardized moments of random sets. *Advances in Applied Probability*, 32, 682-700.

Examples

```
xi <- as.im(heather$coarse, na.replace = 0, eps = 4 * heather$coarse$xstep)

# Estimate pair correlation from a binary map
pclns_directest <- paircorr(xi, estimators = "all")

phat <- coverageprob(xi)
cvchat <- plugincvc(xi)
cpp1 <- cppicka(xi)
```

```
# Compute pair correlation estimates from estimates covariance,
# coverage probability and Picka's reduced-window coverage probability.
pclns_fromcvc <- paircorr.cvchat(cvchat, cpp1, phat, estimators = "all")
```

placegrainsfromlib *Place grains randomly on a point pattern*

Description

Places subsets (grains) of two dimension space randomly on a given point pattern. This is useful for simulating germ-grain models such as Boolean models. Also described here are functions for computing summary properties of the a list of grains.

Usage

```
placegrainsfromlib(
  pp,
  grainlib,
  replace = TRUE,
  prob = NULL,
  w = NULL,
  xy = NULL
)

meanarea.grainlib(
  grainlib,
  weights = rep(1/length(grainlib), length(grainlib))
)

meansetcov.grainlib(
  grainlib,
  weights = rep(1/length(grainlib), length(grainlib)),
  xy
)

covar.grainlib(lambda, grainlib, weights, xy)
```

Arguments

`pp`	A point pattern (in ppp format).
`grainlib`	A list of grains as `owin` objects in a `solist`.
`replace`	passed directly to `sample`. When TRUE grains are chosen from library with replacement.
`prob`	A list of probability weights for each grain in the library. Passed directly to `sample`. If NULL the grains are selected with equal probability.

`w`	Optional desired observation window. If this is non-null then any grains with Frame outside the Frame of `w` will be ignored. This reduces polygonal intersection calculations for very large buffer distances
`xy`	An `im` or binary mask object that is used to specify the pixel array of objects.
`weights`	Probability of selecting each grain in the library
`lambda`	Intensity of germs of a Boolean model - for computing the covariance of a Boolean model that has grain distribution given by `grainlib` and `weights`.

Details

Germ-grain models have two components, a point process (called germs) and a process that creates grains that are centred on the germs. The point process of germs can be easily simulated using a number of **spatstat** functions (e.g. `rpoispp` for Boolean models). To simulate a germ-grain model in a window W the germ process must be simulated in a larger window because grains centred outside W can intersect W. The result must then be cropped to W to achieve a realisation of the germ-grain process within W.

`placegrainsfromlib` randomly samples from a library of grains (`grainlib`) and places these on the points in `pp`. Sampling of the grain is independent of the location of the point in `pp`. It can be used to simulate the grain process in some germ-grain models.

Value

Returns an `owin` object.

Functions

- `placegrainsfromlib()`: Place grains randomly from a list of grains.
- `meanarea.grainlib()`: Compute mean area of a random grain given by the library
- `meansetcov.grainlib()`: Computes the mean of the set covariance of the grains in `grainlib`. `xy` is required because the set covariance function must rasterise the `owin` objects.
- `covar.grainlib()`: Compute the covariance of a Boolean model with random grain given by the library

Author(s)

Kassel Liam Hingee

Examples

```
# Simulate a germ-grain model where germs are a Poisson point process
# and grains are randomly selected from 3 different disc sizes.
grainlib <- solist(disc(radius = 1), disc(radius = 1.9), disc(radius = 0.2))
bufferdist <- 2 #chosen to be larger than the largest radius in library
w <- owin(xrange = c(0, 10), yrange = c(0, 10))

# Simulate the germ process in the window plus a buffer region around window
pp <- rpoispp(lambda = 0.1, win = dilation(w, bufferdist), nsim = 1, drop = TRUE)
xi_withbuffer <- placegrainsfromlib(pp, grainlib)
```

```
# Simulation of germ-grain model is the part within the window
xi <- intersect.owin(xi_withbuffer, w)

# Computation of properties from parameters
lambda <- 0.1
discr <- 10
weights <- c(0.9999, 0.0001)
grainlib <- solist(disc(r = discr), disc(r = 2*discr))
meanarea.grainlib(grainlib, weights)
truecovar <- covar.grainlib(lambda, grainlib, weights, xy = as.mask(w, eps = 2))
```

plugincvc *Plug-in moment covariance estimator*

Description

This function computes the plug-in moment covariance estimate of a stationary RACS from a binary map. For a stationary RACS, Ξ, the covariance for a vector v is the probability of two points separated by a vector v are covered by Ξ

$$C(v) = P(\{x, x+v\} \subseteq \Xi).$$

Usage

```
plugincvc(xi, obswin = NULL, setcov_boundarythresh = NULL)
```

Arguments

`xi`	An observation of a RACS of interest as a full binary map (as an `im` object) or as the foreground set (as an `owin` object). In the latter case the observation window, `obswin`, must be supplied.
`obswin`	If `xi` is an `owin` object then `obswin` is an `owin` object that specifies the observation window.
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the covariance is given a value of NA.

Details

The plug-in moment covariance estimator is (Serra, 1982)

$$\hat{C}(v) = \frac{\gamma_{W\cap X}(v)}{\gamma_W(v)}$$

where $\gamma_W(v)$ is the set covariance of the observation window W and $\gamma_{W\cap X}(v)$ is the set covariance of the foreground within W. `plugincvc` uses Fourier transforms to calculate the set covariance (using the `setcov` of the foreground and observation window. Vectors with small $\gamma_W(v)$ are eliminated using `setcov_boundarythresh` as division by small values is numerically unstable.

Value

A **SpatStat** `im` object containing the estimated covariance.

Author(s)

Kassel Liam Hingee

References

Serra, J.P. (1982) *Image Analysis and Mathematical Morphology*. London; New York: Academic Press.

Examples

```
xi <- as.im(heather$coarse, na.replace = 0)
covar <- plugincvc(xi)
```

`racscovariance` *Covariance Estimation*

Description

Estimates the covariance of a stationary RACS. The plug-in moment covariance estimator and newer balanced estimators based on (Picka, 1997; Picka, 2000) are available.

Usage

```
racscovariance(
  xi,
  obswin = NULL,
  setcov_boundarythresh = NULL,
  estimators = "all",
  drop = FALSE
)

racscovariance.cvchat(
  cvchat,
  cpp1 = NULL,
  phat = NULL,
  estimators = "all",
  drop = FALSE
)
```

Arguments

`xi`	A binary map. Either an `im` object or a `owin` object. If an `im` object then pixel values of 1 or TRUE represent foreground, 0 or `FALSE` values represent background, and `NA` values represent outside the observation window. If an `owin` object then `xi` represents foreground and `obswin` is required to specify the observation window.
`obswin`	The observation window as an `owin` object if `xi` is also as an `owin` object.
`setcov_boundarythresh`	To avoid instabilities caused by dividing by very small quantities, if the set covariance of the observation window is smaller than `setcov_boundarythresh`, then the covariance is given a value of NA.
`estimators`	A list of strings specifying covariance estimators to use. See details. Passing `estimators = "all"` will select all available estimators.
`drop`	If TRUE and one estimator is selected then the returned value will be a single `im` object and not a list of `im` objects.
`cvchat`	The plug-in moment estimate of covariance as an `im` object. Typically created with `plugincvc`.
`cpp1`	Picka's reduced window estimate of coverage probability as an `im` object - used in improved (balanced) covariance estimators. Can be generated using `cppicka`.
`phat`	The classical estimate of coverage probability, which is the observed area in `xi` divided by the total area of the observation window. See `coverageprob` for more information.

Details

The covariance of a RACS is also known as the two-point coverage probability, and is closely related to the semivariogram. The covariance of a stationary RACS Ξ given a vector v is the probability that two points separated by a vector v are covered by Ξ.

Given a vector v, the plug-in moment covariance estimate from a binary map is the volume of the set of points, x, such that both x and $x + v$ are observed to be in the foreground relative to the volume of points, x, for which both x and $x + v$ are in the observation window (Hingee, 2019). Picka (1997, 2000) suggested a number of improvements to centred covariance estimation (see `cencovariance`) that 'balanced' the data used to estimate covariance with the data used to estimate coverage probability. These lead to covariance estimators that give estimates for the covariance of Ξ that are a constant offset from covariance estimates for the complement of Ξ (note the constant offset depends on the coverage probability), which appears to avoid some surprising behaviour that the plug-in moment covariance estimator suffers (Hingee, 2019). These estimators are called `pickaint` and `pickaH` in this package.

Another improved estimator, inspired by an 'intrinsic modification' briefly mentioned by Picka (1997) for pair-correlation estimators, is also available. We have called this estimator `mattfeldt` as a similar isotropic estimator for pair-correlation was studied by Mattfeldt and Stoyan (2000).

The estimators available are (see (Hingee, 2019) for more information):

- `plugin` the plug-in moment covariance estimator
- `mattfeldt` an estimator inspired by an 'intrinsically' balanced pair-correlation estimator from Picka that was later studied in an isotropic situation by Mattfeldt and Stoyan (2000)

- `pickaint` an estimator inspired by an 'intrinsically' balanced centred covariance estimator from Picka (2000).
- `pickaH` an estimator inspired by the 'additively' balanced centred covariance estimator from Picka (2000).

Value

If `drop = TRUE` and only one estimator is requested then an `im` object containing the covariance estimate. Otherwise a named `imlist` of covariance estimates corresponding to each requested estimator.

Functions

- `racscovariance()`: Estimates covariance from a binary map.
- `racscovariance.cvchat()`: Computes covariance estimates from a plug-in moment estimate of covariance, Picka's reduced window estimate of coverage probability, and the usual estimate of coverage probability. If these estimates already exist then `racscovariance.cvchat` can save significant computation time.

Author(s)

Kassel Liam Hingee

References

Hingee, K.L. (2019) *Spatial Statistics of Random Closed Sets for Earth Observations*. PhD: Perth, Western Australia: University of Western Australia. Submitted.

Mattfeldt, T. and Stoyan, D. (2000) Improved estimation of the pair correlation function of random sets. *Journal of Microscopy*, 200, 158-173.

Picka, J.D. (1997) *Variance-Reducing Modifications for Estimators of Dependence in Random Sets*. Ph.D.: Illinois, USA: The University of Chicago.

Picka, J.D. (2000) Variance reducing modifications for estimators of standardized moments of random sets. *Advances in Applied Probability*, 32, 682-700.

Examples

```
xi <- heather$coarse
obswin <- Frame(xi)

# Estimate from a binary map
balancedcvchats_direct <- racscovariance(xi, obswin = obswin, estimators = "all")

phat <- coverageprob(xi, obswin = obswin)
cvchat <- plugincvc(xi, obswin)
cpp1 <- cppicka(xi, obswin = Frame(heather$coarse))
harmonised <- harmonise.im(cvchat = cvchat, cpp1 = cpp1)
cvchat <- harmonised$cvchat
cpp1 <- harmonised$cpp1
```

```
# Compute balanced estimate of covariance from other estimates
balancedcvchats_fromplugincvc <- racscovariance.cvchat(cvchat,
                         cpp1, phat, estimators = "pickaH", drop = TRUE)
```

rbdd *Simulation of Boolean Model of Deterministic Discs*

Description

Functions for simulating a Boolean model with grains that are discs of fixed constant radius (the abbreviation 'bdd' is short for Boolean model with Deterministic Discs). A Boolean model is a two stage model, first the locations (called germs) of grains are randomly distributed according to a Poisson point process, then a random grain is placed on each germ independently. Introductions to Boolean models are available in many stochastic geometry books (Chiu et al., 2013). Also described here are functions for calculating the coverage probability, germ intensity, and covariance from model parameters for a Boolean model with deterministic discs.

Usage

```
rbdd(lambda, discr, window, seed = NULL)

bddcoverageprob(lambda, discr)

bddlambda(coverp, discr)

bdddiscr(coverp, lambda)

bddcovar.iso(r, lambda, discr)

bddcovar(xrange, yrange, eps, lambda, discr)
```

Arguments

`lambda`	Intensity of the germ process (which is a Poisson point process)
`discr`	Radius of the discs
`window`	The window to simulate in (an `owin` object)
`seed`	Optional input (default in NULL). Is an integer passed to `set.seed`. Used to reproduce patterns exactly.
`coverp`	Coverage probability of the Boolean model
`r`	is the radius to calculate covariance
`xrange`	range of x values for `bddcovar`
`yrange`	range of y values for `bddcovar`
`eps`	list of length 2 of the steps between samples points in x and y respectively for `bddcovar`. If eps is of length 1 then the steps between sample points in the x and y directions will both be equal to eps.

Value

See Functions section.

Functions

- `rbdd()`: Returns an `owin` that is a set generated by simulating a Boolean model with specified intensity and disc radius. The window information is not contained in this object. If the simulated set is empty then an empty `owin` object is returned. The point process of germs is generated using **spatstat**'s `rpoispp`.
- `bddcoverageprob()`: Returns the true coverage probability given the intensity and disc radius.
- `bddlambda()`: Returns the germ intensity given coverage probability and disc radius.
- `bdddiscr()`: Returns the disc radius given coverage probability and germ intensity.
- `bddcovar.iso()`: Returns the true covariance of points separated by a distance `r` given the intensity, `lambda` and disc radius `discr` of the model.
- `bddcovar()`: Returns an image of the covariance as calculated from disc radius and intensity.

WARNING

The returned object of `rbdd` is an `owin` specifying the realisation of the Boolean model within the simulation window. The simulation window is not included, thus the object returned by `rbdd` can have much smaller extent than the simulation window (e.g. when the simulated set is empty).

References

Chiu, S.N., Stoyan, D., Kendall, W.S. and Mecke, J. (2013) *Stochastic Geometry and Its Applications*, 3rd ed. Chichester, United Kingdom: John Wiley & Sons.

Examples

```
# Simulate Boolean model with discs of radius 10.
# The coverage probability is very close to 0.5.
discr <- 10
w <- owin(xrange = c(0, 100), c(0, 100))
lambda <- 2.2064E-3
xi <- rbdd(lambda, discr, w)

# Compute properties of Boolean model from parameters
cp <- bddcoverageprob(lambda, discr)
cvc <- bddcovar(c(-10, 10), c(-10, 10), c(0.2, 0.2), lambda, discr)
```

`rbdr` *Simulation of Boolean Model of Deterministic Rectangles*

Description

Functions for simulating a Boolean model with grains that are deterministic rectangles. A Boolean model is a two stage model, first the locations (called germs) of grains are randomly distributed according to a Poisson point process, then a random grain is placed on each germ independently. An introduction can be found in (Chiu et al., 2013). Also described in this help file are functions for calculating the coverage probability and covariance.

Usage

```
rbdr(lambda, grain, win, seed = NULL)

bdrcoverageprob(lambda, grain)

bdrcovar(lambda, grain, xy)
```

Arguments

`lambda`	Intensity of the germ process (which is a Poisson point process)
`grain`	Rectangle object specifying the grain
`win`	The window to simulate in (an `owin` object)
`seed`	Optional input (default in NULL). Is an integer passed to `set.seed`. Used to reproduce patterns exactly.
`xy`	A raster object that specifies the pixel coordinates of the desired covariance image. `xy` works in similar fashion to passing an image or pixel mask through the `xy` argument of `as.mask` in **spatstat**.

Value

Depends on the function used (see Functions section).

Functions

- `rbdr()`: Returns an `owin` that is a set generated by simulating a Boolean model with a specified intensity and fixed rectangular grain. The window information is not contained in this object. If the simulated set is empty then an empty `owin` object is returned. The point process of germs is generated using spatstat's `rpoispp`.
- `bdrcoverageprob()`: Returns the true coverage probability given the intensity and grain.
- `bdrcovar()`: Returns an image of the covariance as calculated from disc radius and intensity.

WARNING

The returned object of `rbdr` is only the foreground of a binary map and thus can have much smaller extent than the simulation window (e.g. when the simulated set is empty).

References

Chiu, S.N., Stoyan, D., Kendall, W.S. and Mecke, J. (2013) *Stochastic Geometry and Its Applications*, 3rd ed. Chichester, United Kingdom: John Wiley & Sons.

Examples

```
grain <- owin(xrange = c(-5, 5), yrange = c(-5, 5))
win <- owin(xrange = c(0, 100), c(0, 100))
lambda <- 4.2064E-3
xi <- rbdr(lambda, grain, win)

cp_theoretical <- bdrcoverageprob(lambda, grain)
xy <- as.mask(dilationAny(win, win), eps = c(1, 1))
cvc_theoretical <- bdrcovar(lambda, grain, xy)
```

`rblnd` *Simulate a Boolean model of discs with log normal disc radii*

Description

Simulates a Boolean model of discs with log normal radii by first simulating a Poisson point process and then placing discs of random radii around each point (the radii are generated using a log normal distribution).

Usage

```
rblnd(obswin, bufferdist, lambda, meanlog, sdlog, seed = NULL)
```

Arguments

`obswin`	An `owin` object specifying the desired simulation region
`bufferdist`	A distance to expand `obswin` so that discs with centres near `obswin` are also simulated.
`lambda`	Intensity of the Poisson point process, passed to rpoispp. It could be either a single positive number, or any other object that rpoispp can understand.
`meanlog`	For the distribution of radii. The logarithm of the distribution is set to have mean `meanlog`.
`sdlog`	For the distribution of radii. The logarithm of the distribution is set to have standard deviation `sdlog`
`seed`	Optional input (default is `NULL`). Is an integer passed to `set.seed`. Used to reproduce patterns exactly.

Details

The point process needs to be simulated in a larger region than the desired observation window to account for the possibility of discs that intersect the observation window, but have germs outside the observation window.

The point process of germs is generated using spatstat's `rpoispp`.

Value

Returns an `owin` object cropped to `obswin`.

Warning

A good choice of `bufferdist` is required and will be sensitive to the distribution of radii.

Examples

```
w <- owin(xrange = c(0, 10), yrange = c(0, 10))
xi <- rblnd(w, 2, 0.3, -1, 0.2)
```

`rbpto` *Simulate Boolean Model with Grains Scaled According to a Truncated Pareto Distribution*

Description

Functions for simulation and computing theoretical values of a Boolean model with identically shaped grains with size given by a truncated Pareto distribution.

Usage

```
rbpto(lambda, grain, win, xm, alpha, lengthscales, seed = NULL, xy = NULL)

bpto.coverageprob(lambda, grain, xm, alpha, lengthscales = 1:500)

bpto.germintensity(coverp, grain, xm, alpha, lengthscales = 1:500)

bpto.covar(lambda, grain, xm, alpha, lengthscales = 1:500, xy)
```

Arguments

`lambda`	Intensity of the germ process (which is a Poisson point process)
`grain`	A single `owin` object that gives the shape and size of the grain at scale 1
`win`	The window to simulate in (an `owin` object)
`xm`	A parameter governing the shape of the Pareto distribution used - see details
`alpha`	A parameter governing the shape of the Pareto distribution used • see details
`lengthscales`	A list of scales of the `grain` for which to approximate the Pareto distribution: The grain for a germ is chosen by selecting a scaled version of `grain` where `lengthscales` specifies the possible scales and the Pareto distribution is used to specify the probability of selection of each scale.
`seed`	Optional input (default in NULL). Is an integer passed to `set.seed`. Used to reproduce patterns exactly.

`xy`	A raster object that specifies pixel coordinates of the final simulated binary map. It is used the same way as `xy` is `as.mask` in **spatstat**. If non-null then the computations will be performed using rasters. Otherwise if `grain` and `win` are polygonal then computations may be all polygonal.
`coverp`	Coverage probability of the Boolean model.

Details

The parameters `xm` and `alpha` are such that the CDF of the Pareto distribution is $P(s <= x) = 1 - (xm/x)^{alpha}$. The distribution of grains scales is a step-function approximation to the CDF with steps at `lengthscales`.

Value

An `owin` object.

Functions

- `rbpto()`: Simulate Boolean model with grain size distributed according to a truncated Pareto distribution.
- `bpto.coverageprob()`: The coverage probability of the Boolean model with grain size distributed according to a truncated Pareto distribution.
- `bpto.germintensity()`: The germ intensity of the Boolean model with grain size distributed according to a truncated Pareto distribution.
- `bpto.covar()`: The covariance of the Boolean model with grain size distributed according to a truncated Pareto distribution. `xy` is required to specify resolution and offset of pixel grid.

Examples

```
lambda <- 0.2
win <- square(r = 10)
grain <- disc(r = 0.2)
xm <- 0.01
alpha <- 2
lengthscales <- seq(1, 5, by = 0.1)
xi <- rbpto(lambda, grain, win, xm, alpha, lengthscales = lengthscales)

# Compute properties of the Boolean model from parameters
bpto.coverageprob(lambda, grain, xm, alpha, lengthscales = lengthscales)
covar <- bpto.covar(lambda, grain, xm, alpha, lengthscales = lengthscales,
                    xy = as.mask(win, eps = 2))
```

`secondorderprops` *Estimate Second-Order Properties of a RACS*

Description

Estimates many second order properties of RACS, gliding box lacunarity, covariance, centred covariance, and pair-correlation. This can be faster than computing estimates of multiple second order properties separately as Fourier transforms of the binary map are not repeated.

Usage

```
secondorderprops(
  xiim,
  gblargs = NULL,
  covarargs = NULL,
  cencovarargs = NULL,
  paircorrargs = NULL,
  returnrotmean = FALSE
)
```

Arguments

`xiim`	A **spatstat** `im` object with pixel values that are either TRUE, FALSE or NA. TRUE represents foreground, FALSE represents background and NA represents unobserved locations.
`gblargs`	A list of named arguments passed to `gblemp` and `gbl.cvchat`. The estimators used can be specified by passing an argument named 'estimators' contain a list of estimator names, as given in `gbl`. If NULL then GBL will not be estimated.
`covarargs`	A list of named arguments passed to `racscovariance.cvchat`. If NULL then covariance will not be returned.
`cencovarargs`	A list of named arguments passed to `cencovariance.cvchat`. If NULL then pair correlation will not be returned.
`paircorrargs`	A list of named arguments passed to `paircorr.cvchat`. If NULL then pair correlation will not be returned.
`returnrotmean`	Logical. If FALSE the anisotropic estimates of covariance and pair-correlation will be returned as `im` objects. If TRUE then average covariance and pair-correlation over all directions will be returned as `fv` objects.

Value

A named list of estimated properties. When multiple estimators have been requested for the same property, then the entry in the list is itself a list, with each entry corresponding to a different estimator.

Warning

The user interface for this function is substantially more stretched the knowledge of the author, Kassel Hingee. Therefore, there is greater chance of encountering bugs. Kassel Hingee apologises for any bugs you encounter, and requests to be informed (thank you!).

Examples

```
xiim <- as.im(heather$coarse, value = TRUE,
              na.replace = FALSE)
gblargs = list(boxwidths = seq(1, 10, by = 1), estimators = c("GBLemp", "GBLcc.pickaH"))
covarargs = list(estimators = "all")
cencovarargs = list(estimators = "pickaH")
paircorrargs = list(estimators = "pickaH")
returnrotmean = TRUE
secondests <- secondorderprops(xiim,
   gblargs = gblargs,
   covarargs = covarargs,
   cencovarargs = cencovarargs,
   paircorrargs = paircorrargs,
   returnrotmean = FALSE)
```

`summary.imlist` *Pointwise summary of a list of* `im` *objects*

Description

This function assumes that `im` objects are each realisations of the same (stochastic) object. It returns pointwise summaries such as observed sample mean and sample variance.

Usage

```
## S3 method for class 'imlist'
summary(object, ..., harmonizeobject = TRUE)
```

Arguments

`object`	A list of `im` objects
`...`	Ignored
`harmonizeobject`	If TRUE (default) the pixel dimensions of the images will be harmonized. Otherwise the object will be tested for compatibility.

Value

A list `im` objects containing the pointwise mean, variance and maxima and minima.

Author(s)

Kassel Hingee

Examples

```
# reduce resolution in setcov() for faster (less accurate) computation
oldopt <- spatstat.options()
spatstat.options("npixel" = 2^4)

obspatterns <- replicate(3, rbdd(10, 0.05, window = square(1)), simplify = FALSE)
ims <- solapply(obspatterns,
 function(x) racscovariance(x, obswin = square(1), estimators = "pickaH", drop = TRUE))
summ <- summary.imlist(ims, harmonizeobject = FALSE)
spatstat.options(oldopt)
```

Index