

Package ‘SpatEntropy’

November 17, 2023

Type Package

Title Spatial Entropy Measures

Version 2.2-4

Author L. Altieri, D. Cocchi, G. Roli

Maintainer Altieri Linda <linda.altieri@unibo.it>

Depends R (>= 4.0.0), spatstat (>= 3.0-2)

Imports spatstat.geom, spatstat.random, graphics, grDevices

Description The heterogeneity of spatial data presenting a finite number of categories can be measured via computation of spatial entropy. Functions are available for the computation of the main entropy and spatial entropy measures in the literature. They include the traditional version of Shannon's entropy (Shannon, 1948 <[doi:10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x)>), Batty's spatial entropy (Batty, 1974 <[doi:10.1111/j.1538-4632.1974.tb01014.x](https://doi.org/10.1111/j.1538-4632.1974.tb01014.x)>), O'Neill's entropy (O'Neill et al., 1998 <[doi:10.1007/BF00162741](https://doi.org/10.1007/BF00162741)>), Li and Reynolds' contagion index (Li and Reynolds, 1993 <[doi:10.1007/BF00125347](https://doi.org/10.1007/BF00125347)>), Karlstrom and Ceccato's entropy (Karlstrom and Ceccato, 2002 <<https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-61351>>), Leibovici's entropy (Leibovici, 2009 <[doi:10.1007/978-3-642-03832-7_24](https://doi.org/10.1007/978-3-642-03832-7_24)>), Parresol and Edwards' entropy (Parresol and Edwards, 2014 <[doi:10.3390/e16041842](https://doi.org/10.3390/e16041842)>) and Altieri's entropy (Altieri et al., 2018, <[doi:10.1007/s10651-017-0383-1](https://doi.org/10.1007/s10651-017-0383-1)>). Full references for all measures can be found under the topic 'SpatEntropy'. The package is able to work with lattice and point data. The updated version works with the updated 'spatstat' package (>= 3.0-2).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-11-17 12:30:02 UTC

R topics documented:

altieri	2
areapart	4
batty	6
bologna	9
bolognaTess	10
bolognaW	11
contagion	12
karlstrom	13
leibovici	16
oneill	18
parredw	20
raintrees	21
raintrees2	23
raintreesCOV	25
shannon	26
shannonZ	27
SpatEntropy	29
turin	30
turinTess	31
turinW	33
varshannon	33
varshannonZ	34
Index	36

altieri	<i>Altieri's spatial entropy.</i>
---------	-----------------------------------

Description

This function computes spatial mutual information and spatial residual entropy as in Altieri et al (2017) and following works. References can be found at SpatEntropy.

Usage

```
altieri(data, cell.size = 1, distbreak = "default", verbose = F, plotout = T)
```

Arguments

data	If data are lattice, a data matrix, which can be numeric, factor, character, ... If the dataset is a point pattern, data is a ppp object.
cell.size	A single number or a vector of length two, only needed if data are lattice. It gives the length of the side of each pixel; if the pixel is rectangular, the first number gives the horizontal side and the second number gives the vertical side. Default to 1. Ignored if data are points.

<code>distbreak</code>	Numeric. The chosen distance breaks for selecting pairs of pixels/points within the observation area. The default option is <code>c(cell.size[1], 2*cell.size[1])</code> for lattice data, and <code>c(mindist, 2*mindist)</code> for point data, where <code>mindist</code> is the first decile of the nearest neighbour distance distribution. Only the internal breaks have to be specified, the first and last break are automatically added as 0 and the maximum distance within the observation area, respectively.
<code>verbose</code>	Logical. If TRUE an output is printed in order to follow the progress of the work (recommended for large dataset). Default set to FALSE.
<code>plotout</code>	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

The computation of Altieri's entropy starts from a point or areal dataset, for which Shannon's entropy of the transformed variable Z (for details see [shannonZ](#))

$$H(Z) = \sum p(z_r) \log(1/p(z_r))$$

is computed using all possible pairs within the observation area. Then, its two components spatial mutual information

$$SMI(Z, W) = \sum p(w_k) \sum p(z_r|w_k) \log(p(z_r|w_k)/p(z_r))$$

and spatial residual entropy

$$H(Z)_W = \sum p(w_k) \sum p(z_r|w_k) \log(1/p(z_r|w_k))$$

are calculated in order to account for the overall role of space in determining the data heterogeneity. Besides, starting from a partition into distance classes, a list of adjacency matrices is built, which identifies what pairs of units must be considered for each class. Spatial mutual information and spatial residual entropy are split into local terms according to the chosen distance breaks, so that the role of space can be investigated both in absolute and relative terms. In the function output, the relative partial terms are returned so that they sum to 1 for each distance class: e.g. if the relative SPI terms is 0.3 and the relative residual term is 0.7, the interpretation is that, at the specific distance class, 30% of the entropy is due to the role of space as a source of heterogeneity. The function is able to work with lattice data with missing data, as long as they are specified as NAs: missing data are ignored in the computations. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation and only couples of non-NA observations are considered.

Value

A list with elements:

- `distance.breaks` a two column matrix with the lower and upper extreme of each distance class
- `SPI.terms` the spatial partial information terms
- `rel.SPI.terms` the relative version of spatial partial information terms (see the details)
- `RES.terms` the spatial partial residual entropies

- `rel.RES` terms the relative version of spatial partial residual entropies (see the details)
- `SMI` the spatial mutual information
- `RES` the global residual entropy
- `ShannonZ` Shannon's entropy of Z in the same format as the output of `shannonZ()`
- `W.distribution` the spatial weights for each distance range
- `total.pairs` the total number of pairs over the area (realizations of Z)
- `class.pairs` the number of pairs for each distance range.
- `cond.Z.distribution` a list with the conditional absolute and relative frequencies of Z for each distance range

Examples

```
#lattice data
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
outp=altieri(data)
outp=altieri(data, cell.size=2) #same result
outp=altieri(data, cell.size=2, distbreak=c(2, 5))
#plot data
plot(as.im(data, W=square(nrow(data))),
      col=grDevices::gray(seq(1,0,l=length(unique(c(data))))),
      main="", ribbon=TRUE)

#lattice data with missing values
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
data=rbind(rep(NA, ncol(data)), data, rep(NA, ncol(data)))
outp=altieri(data)
#plot data
plot(as.im(data, W=square(nrow(data))),
      col=topo.colors(length(unique(c(data)[!is.na(c(data))])),
      main="", ribbon=TRUE)

#point data
data=ppp(x=runif(400), y=runif(400), window=square(1),
         marks=(sample(c("a","b","c"), 400, replace=TRUE)))
outp=altieri(data)
outp=altieri(data, verbose=TRUE)
#plot data
plot(data, cols=1:length(unique(marks(data))), main="", pch=16)
#check what happens for badly specified distance breaks
#outp=altieri(data, distbreak=c(1,1.4))
#outp=altieri(data, distbreak=c(1,2))
```

Description

This function partitions the observation area in a number of sub-areas, and assigns the data points/pixels to the areas. This function is useful either when a random partition wants to be created, or when the user wants to set the area's centroids and is happy with an area tessellation in Voronoi polygons according to the defined centroids.

Usage

```
areapart(data, G, cell.size = 1, win = NULL, plotout = T)
```

Arguments

data	If data are lattice, a data matrix, which can be numeric, factor, character, ... If the dataset is a point pattern, data is a ppp object.
G	An integer if sub-areas are randomly generated, determining the number G of sub-areas. Alternatively, a 2-column matrix with the sub-areas centroids' coordinates.
cell.size	A single number. If data are lattice, the length of the side of each pixel. Default to 1. Ignored if data are points.
win	Optional, the observation area given as a <code>owin</code> object. If data are a point pattern ppp object, this argument is ignored and the observation area is extracted from the object. If data are given as a matrix and the area is not specified, the default is a rectangle with x range from 0 to the number of columns of the data, and y range from 0 to the number of rows of the data.
plotout	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

The function is preliminary to the computation of Batty's or Karlstrom and Ceccato's entropy. An event of interest (in the form of a point or binary areal dataset) occurs over an observation area divided into sub-areas. If the partition is random, this function generates the sub-areas by randomly drawing the areas' centroids over the observation window. Then, data points/pixels are assigned to the area with the closest centroid. When data are pixels, each pixel is assigned to an area according to the coordinates of its own centroid. The function also works for non-binary datasets and marked ppp objects.

Value

A list with elements:

- `G.pp` a point pattern containing the G areas' centroids
- `data.assign` a four column matrix, with all pairs of data coordinates and data values matched to one of the G areas (numbered 1 to G). If the dataset is an unmarked ppp object, the data category column is a vector of 1s.

Moreover, a plot is produced showing the data and the area partition.

Examples

```
#LATTICE DATA

data=matrix(sort(sample(c("a","b","c"), 100, replace=TRUE)), nrow=10)
partition=areapart(data, G=5)
partition=areapart(data, G=5, cell.size=2)

#providing a pre-fixed area partition
data=matrix(sort(sample(c("a","b","c"), 100, replace=TRUE)), nrow=10)
win=square(nrow(data))
GG=cbind(runif(5, win$xrange[1], win$xrange[2]),
         runif(5, win$yrange[1], win$yrange[2]))
partition=areapart(data, G=GG)

#POINT DATA

data=ppp(x=runif(100), y=runif(100), window=square(1))
partition=areapart(data, 10)

#with marks
data=ppp(x=runif(100), y=runif(100), window=square(1),
         marks=(sample(c("a","b","c"), 100, replace=TRUE)))
GG=cbind(runif(10, data$window$xrange[1], data$window$xrange[2]),
         runif(10, data$window$yrange[1], data$window$yrange[2]))
partition=areapart(data, G=GG)
```

batty

Batty's entropy.

Description

This function computes Batty's spatial entropy, following Batty (1976), see also Altieri et al (2017 and following) (references are under the topic [SpatEntropy](#)).

Usage

```
batty(
  data,
  category = 1,
  cell.size = 1,
  partition = 10,
  win = NULL,
  rescale = T,
  plotout = T
)
```

Arguments

<code>data</code>	If data are lattice, a data matrix, which can be numeric, factor, character, ... If the dataset is a point pattern, <code>data</code> is a <code>ppp</code> object.
<code>category</code>	A single value matching the data category of interest for computing Batty's entropy. Default to 1. If the dataset is an unmarked point pattern, this argument must not be changed from the default. In the plot, only data belonging to the selected category are displayed.
<code>cell.size</code>	A single number or a vector of length two, only needed if data are lattice. It gives the length of the side of each pixel; if the pixel is rectangular, the first number gives the horizontal side and the second number gives the vertical side. Default to 1. Ignored if data are points.
<code>partition</code>	Input defining the partition into subareas. If an integer, it defines the number of sub-areas that are randomly generated by <code>areapart</code> ; if a two column matrix with coordinates, they are the centroids of the subareas built by <code>areapart</code> . Alternatively, it can be the output of <code>areapart</code> , a <code>tess</code> object built by the user, a <code>list</code> object with arguments <code>tiles</code> , i.e. a list of <code>owin</code> objects defining the partition, and <code>n</code> , the number of subareas. Lastly, it can be an <code>im</code> object, i.e. a factor- or character-valued pixel image on the same observation window as the data, so that the partition is defined according to the values of the image. The default option is <code>partition=areapart(data, G=10)</code> , which generates 10 random sub-areas.
<code>win</code>	Optional, the observation area given as a <code>owin</code> object. If data are a point pattern <code>ppp</code> object, this argument is ignored and the observation area is extracted from the object. If data are given as a matrix, the area should be specified; the default is a rectangle with <code>x</code> range from 0 to the number of columns of the data, and <code>y</code> range from 0 to the number of rows of the data.
<code>rescale</code>	Logical. Default to TRUE, checks whether the size of the observation area or of any of the sub-areas is smaller than 1. If so, computational issues may arise due to negative logarithms, therefore the function automatically performs a rescaling of all sub-areas, computes the entropy over the rescaled area and then transforms it back to the entropy of the original dataset. In such case, a warning is produced to make the user aware of such operation.
<code>plotout</code>	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

Batty's spatial entropy measures the heterogeneity in the spatial distribution of a phenomenon of interest, with regard to an area partition. It is high when the phenomenon is equally intense over the sub-areas, and low when it concentrates in one or few sub-areas. This function allows to compute Batty's entropy as

$$H_B = \sum p_g \log(T_g/p_g)$$

where p_g is the probability of occurrence of the phenomenon over sub-area g , and T_g is the sub-area size. When data are categorical, the phenomenon of interest corresponds to one category, which must be specified. If data are an unmarked point pattern, a fake mark vector is created with the same category for all points. For comparison purposes, the relative version of Batty's entropy is

also returned, i.e. Batty's entropy divided by its maximum $\log(\sum T_g)$. Note that when the total observation area is 1, then $\log(\sum T_g) = 0$, therefore in that case during the computation all T_g s are multiplied by 100 and a warning is produced. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation.

Value

A list of five elements:

- `batty` Batty's entropy
- `range` The theoretical range of Batty's entropy
- `rel.batty` Batty's entropy divided by $\log(\sum T_g)$ for comparison across observation areas.
- `areas` a dataframe giving, for each sub-area of the partition, the absolute and relative frequency of the points/pixels of interest, the sub-area size and the intensity defined as pg/Tg
- `area.tess` a tess object with the area partition

Moreover, a plot is produced showing the data and the area partition.

Examples

```
#LATTICE DATA

data=matrix((sample(c("a","b","c"), 100, replace=TRUE)), nrow=10)
batty.entropy=batty(data, category="a")

#POINT DATA

#unmarked pp
data=ppp(x=runif(100, 0, 10), y=runif(100, 0, 10), window=square(10))
batty.entropy=batty(data)

#smaller window so that some areas' size are smaller than 1
data=ppp(x=runif(100, 0, 3), y=runif(100, 0, 3), window=square(3))
batty.entropy=batty(data)

#marked pp
data=ppp(x=runif(100, 0, 10), y=runif(100, 0, 10), window=square(10),
        marks=(sample(1:5, 100, replace=TRUE)))
plot(data) #see ?plot.ppp for options
#if you want to compute the entropy on all points
batty.entropy=batty(unmark(data))
#if you want to compute the entropy on a category, say 3
batty.entropy=batty(data, category=3)
```

bologna

Bologna urban data.

Description

A lattice dataset with Bologna's Urban Morphological Zones (UMZ, see EEA, 2011).

Usage

bologna

Format

A matrix with 135 rows and 124 columns. Values are either 0 (non-urban) or 1 (urban). Pixels outside the administrative borders are classified as NA.

Details

This raster/pixel/lattice dataset comes from the EU CORINE Land Cover project (EEA, 2011) and is dated 2011. It is the result of classifying the original land cover data into urbanised and non-urbanised zones, known as 'Urban Morphological Zones' (UMZ, see EEA, 2011). UMZ data are useful to identify shapes and patterns of urban areas, and thus to detect what is known as urban sprawl. Bologna's metropolitan area is extracted from the European dataset and is composed by the municipality of Bologna and the surrounding municipalities: . The dataset is made of 135x124 pixels of size 250x250 metres.

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(bologna)
#plot(as.im(bologna), main="", col=gray(c(0.8,0)), ribbon=FALSE)

#shannon's entropy
shannon(bologna)

#shannon's entropy of Z (urban/non-urban pairs)
shannonZ(bologna)

#oneill's entropy
oneill(bologna)

#leibovici's entropy on a subset of the window
bolsub=bologna[30:70,45:85]
plot(as.im(bolsub), main="", col=gray(c(0.8,0)), ribbon=FALSE)
leibovici(bolsub, cell.size=250, ccdist=400, verbose=TRUE)
```

```

#altieri's entropy
bolsub=bologna[30:70,45:85]
plot(as.im(bolsub), main="", col=gray(c(0.8,0)), ribbon=FALSE)
altieri(bolsub, cell.size=250, distbreak=c(250, 500), verbose=TRUE)

#batty's entropy
#on all points, with a random partition in 10 sub-areas
batty.ent=batty(bologna, cell.size=250, partition=10, win=bolognaW)
#plot with partition
data(bolognaW)
#plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#plot(batty.ent$area.tess, add=TRUE, border=2)

#batty's entropy with a partition based on the administrative areas
data(bolognaTess)
batty.ent=batty(bologna, cell.size=250, partition=bolognaTess, win=bolognaW)
#plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#for(i in 1:bolognaTess$n) plot(bolognaTess$tiles[[i]], add=TRUE, border=2)

#karlstrom and ceccato's entropy
data(bolognaW)
KC.ent=karlstrom(bologna, cell.size=250, partition=15, win=bolognaW, neigh=3)
#plot with partition
#plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#plot(KC.ent$area.tess, add=TRUE, border=2)

#karlstrom and ceccato's entropy with a partition based on the administrative
#areas
data(bolognaTess)
KC.ent=karlstrom(bologna, cell.size=250, partition=bolognaTess, win=bolognaW,
neigh=10000, method="distance")
#plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#for(i in 1:bolognaTess$n) plot(bolognaTess$tiles[[i]], add=TRUE, border=2)

```

bolognaTess

Municipalities' administrative borders for Bologna urban data.

Description

City borders of all municipalities included in the Bologna dataset, in the format of polygonal windows `owin` objects.

Usage

```
bolognaTess
```

Format

A list of three:

- tiles a list of 11, each element is a owin object with the administrative border of one municipality of Bologna's dataset
- n the number of municipalities
- names the names of the 11 municipalities, in the same order as the windows

Details

The object contains a list of 11 observation windows created as owin objects based on the coordinates of the border polygons, for each municipality. See ?owin for details. The object also contains the names of the municipalities, in Italian. Examples on the usefulness of the administrative borders can be found at the topic [bologna](#).

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(bologna); data(bolognaW); data(bolognaTess)
plot(bolognaW, main="")
plot(bolognaTess$tiles[[1]],border=2, add=TRUE, lwd=2)
for(11 in 2:bolognaTess$n) plot(bolognaTess$tiles[[11]],border=2, add=TRUE, lwd=2)

plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.85,0.4)), ribbon=FALSE)
plot(bolognaTess$tiles[[1]],border=1, add=TRUE, lwd=2)
for(11 in 2:bolognaTess$n) plot(bolognaTess$tiles[[11]],border=1, add=TRUE, lwd=2)

#see examples under the topic "bologna"
```

bolognaW

Observation window for Bologna urban data.

Description

An owin object with the rectangle circumscribing the city border for the Bologna dataset.

Usage

```
bolognaW
```

Format

An owin object. The basic spatial unit is a 250x250 metres pixels.

Details

This observation window is an `owin` object with margins given by the CORINE project coordinates. See `?owin` for details. Examples on the usefulness of the window can be found at the topic [bologna](#).

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(bolognaW)
plot(bolognaW, main="")
plot(as.im(bologna, W=bolognaW), main="", col=gray(c(0.8,0)), ribbon=FALSE, add=TRUE)

#see examples under the topic "bologna"
```

contagion

Li and Reynolds' relative contagion index.

Description

This function computes Li and Reynold's contagion index, following Li and Reynolds (1993), starting from a data matrix. References can be found at [SpatEntropy](#).

Usage

```
contagion(
  data,
  win = spatstat.geom::owin(xrange = c(0, ncol(data)), yrange = c(0, nrow(data))),
  plotout = T
)
```

Arguments

<code>data</code>	A data matrix or vector, can be numeric, factor, character, ...
<code>win</code>	Optional, an object of class <code>owin</code> , the observation window for data plotting
<code>plotout</code>	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

This index is based on the transformed variable Z identifying couples of realizations of the variable of interest. A distance of interest is fixed: the contagion index is originally thought for areas sharing a border, as O'Neill's entropy. Then, all contiguous couples of realizations of the variable of interest are counted and their relative frequencies are used to compute the index, which is $1 - NO$ where NO is the relative version of O'Neill's entropy, i.e. O'Neill's entropy divided by its maximum

$\log(I^2)$, I being the number of categories of the variable under study. The relative contagion index ranges from 0 (no contagion, maximum entropy) to 1 (maximum contagion). The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation and only couples of non-NA observations are considered.

Value

a list of two elements:

- contagion Li and Reynold's relative contagion index
- probabilities a table with absolute frequencies and estimated probabilities (relative frequencies) for all couple categories

Moreover, a plot of the dataset is produced.

Examples

```
#numeric data, square grid
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
contagion(data)
#plot data
plot(as.im(data, W=square(nrow(data))),
      col=grDevices::gray(seq(1,0,length.out=length(unique(c(data))))),
      main="", ribbon=TRUE)

#character data, rectangular grid
data=matrix(sample(c("a","b","c"), 300, replace=TRUE), nrow=30)
contagion(data)
#plot data
plot(as.im(data, W=owin(xrange=c(0,ncol(data)), yrange=c(0,nrow(data))),
      col=terrain.colors(length(unique(c(data))))),
      main="", ribbon=TRUE)
```

karlstrom

Karlstrom and Ceccato's entropy.

Description

This function computes Karlstrom and Ceccato's spatial entropy for a chosen neighbourhood distance, following Karlstrom and Ceccato (2002), see also Altieri et al (2017) and following works (references are under the topic [SpatEntropy](#)).

Usage

```
karlstrom(
  data,
  category = 1,
  cell.size = 1,
```

```

    partition = 10,
    win = NULL,
    neigh = 4,
    method = "number",
    plotout = T
)

battyLISA(
  data,
  category = 1,
  cell.size = 1,
  partition = 10,
  win = NULL,
  neigh = 4,
  method = "number",
  plotout = T
)

```

Arguments

<code>data</code>	If data are lattice, a data matrix, which can be numeric, factor, character, ... If the dataset is a point pattern, <code>data</code> is a ppp object.
<code>category</code>	A single value matching the data category of interest for computing Batty's entropy. Default to 1. If the dataset is an unmarked point pattern, this argument must not be changed from the default.
<code>cell.size</code>	A single number or a vector of length two, only needed if data are lattice. It gives the length of the side of each pixel; if the pixel is rectangular, the first number gives the horizontal side and the second number gives the vertical side. Default to 1. Ignored if data are points.
<code>partition</code>	Input defining the partition into subareas. If an integer, it defines the number of sub-areas that are randomly generated by areapart ; if a two column matrix with coordinates, they are the centroids of the subareas built by areapart . Alternatively, it can be the output of areapart , a tess object built by the user, a list object with arguments <code>tiles</code> , i.e. a list of <code>owin</code> objects defining the partition, and <code>n</code> , the number of subareas. Lastly, it can be an <code>im</code> object, i.e. a factor- or character-valued pixel image on the same observation window as the data, so that the partition is defined according to the values of the image. The default option is <code>partition=areapart(data, G=10)</code> , which generates 10 random sub-areas.
<code>win</code>	Optional, the observation area given as a <code>owin</code> object. If data are a point pattern ppp object, this argument is ignored and the observation area is extracted from the object. If data are given as a matrix and the area is not specified, the default is a rectangle with x range from 0 to the number of columns of the data, and y range from 0 to the number of rows of the data.
<code>neigh</code>	A single number. It can be either the number of neighbours for each sub-area (including the area itself). or the Euclidean distance to define which sub-areas are neighbours, based on their centroids. Default to 4 neighbours.

method	Character, it guides the interpretation of neigh. Either "number" (the default) or "distance".
plotout	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

Karlstrom and Ceccato's spatial entropy measures the heterogeneity in the spatial distribution of a phenomenon of interest, with regard to an area partition and accounting for the neighbourhood. It is similar to Batty's entropy (see [batty](#)) discarding the sub-area size, with the difference that the probability of occurrence of the phenomenon over area g is actually a weighted sum of the neighbouring probabilities.

$$H_{KC} = \sum p_g \log(1/\tilde{p}_g)$$

where p_g is the probability of occurrence of the phenomenon over sub-area g , and \tilde{p}_g is the averaged probability over the neighbouring areas (including the g -th area itself). When data are categorical, the phenomenon of interest corresponds to one category, which must be specified. If data are an unmarked point pattern, a fake mark vector is created with the same category for all points. For comparison purposes, the relative version of Karlstrom and Ceccato's entropy is also returned, i.e. Karlstrom and Ceccato's entropy divided by its maximum $\log(\text{number of sub-areas})$. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation.

Value

A list of five elements:

- `karlstrom` Karlstrom and Ceccato's entropy
- `range` The theoretical range of Karlstrom and Ceccato's entropy
- `rel.karl` Karlstrom and Ceccato's entropy divided by $\log(G)$ (number of sub-areas) for comparison across observation areas.
- `areas` a dataframe giving, for each sub-area, the absolute and relative frequency of the points/pixels of interest, the weighted probabilities of the neighbours and the sub-area size
- `area.tess` a tess object with the area partition

Moreover, a plot is produced showing the data and the area partition.

Examples

```
#LATTICE DATA

data=matrix((sample(c("a","b","c"), 100, replace=TRUE)), nrow=10)
KC.entropy=karlstrom(data, category="a")
KC.entropy=karlstrom(data, category="a", neigh=3.5, method="distance")
##to plot
data.binary=matrix(as.numeric(data=="a"), nrow(data))
plot(as.im(data.binary, W=KC.entropy$area.tess$window), main="",
      col=grDevices::gray(seq(1,0,l=length(unique(c(data.binary))))), ribbon=FALSE)
plot(KC.entropy$area.tess, add=TRUE, border=2)
```

```

#POINT DATA

#unmarked ppp
data=ppp(x=runif(100, 0, 10), y=runif(100, 0, 10), window=square(10))
KC.entropy=karlstrom(data)
##to plot
plot(data)
plot(KC.entropy$area.tess, add=TRUE, border=2)

#marked ppp
data=ppp(x=runif(100, 0, 10), y=runif(100, 0, 10), window=square(10),
        marks=(sample(1:5, 100, replace=TRUE)))
#if you want to compute the entropy on all points
KC.entropy=karlstrom(unmark(data))
#if you want to compute the entropy on a category, say 3
KC.entropy=karlstrom(data, category=3)
##to plot using the selected category
ind=which(spatstat.geom::marks(data)==3)
data.binary=unmark(data[ind])
plot(data.binary)
plot(KC.entropy$area.tess, add=TRUE, border=2)

```

leibovici

Leibovici's entropy.

Description

This function computes Leibovici's entropy according to a chosen distance d (with O'Neill's entropy as a special case) following Leibovici (2009), see also Altieri et al (2017). References can be found at `SpatEntropy`.

Usage

```

leibovici(
  data,
  cell.size = 1,
  ccdist = cell.size[1],
  win = NULL,
  verbose = F,
  plotout = T
)

```

Arguments

`data` If data are lattice, a data matrix, which can be numeric, factor, character, ... If the dataset is a point pattern, `data` is a `ppp` object.

<code>cell.size</code>	A single number or a vector of length two, only needed if data are lattice. It gives the length of the side of each pixel; if the pixel is rectangular, the first number gives the horizontal side and the second number gives the vertical side. Default to 1. Ignored if data are points.
<code>ccdist</code>	A single number. The chosen distance for selecting couples of pixels/points within the observation area. Default to <code>cell.size[1]</code> .
<code>win</code>	Optional, an object of class <code>owin</code> , the observation window for data plotting
<code>verbose</code>	Logical. If TRUE an output is printed in order to follow the progress of the work (recommended for large dataset). Default set to FALSE.
<code>plotout</code>	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

This index is based on the transformed variable Z identifying couples of realizations of the variable of interest. A distance of interest is fixed, which in the case of O'Neill's entropy is the contiguity, i.e. sharing a border for lattice data. Then, all couples of realizations of the variable of interest lying at a distance smaller or equal to the distance of interest are counted, and their relative frequencies are used to compute the index with the traditional Shannon's formula. #'

$$H_L(Z) = \sum p(z_r|d) \log(1/p(z_r|d))$$

where $z_r = (x_i, x_{i'})$ is a generic couple of realizations of the study variable X . The conditioning on d means that only couples within a predefined distance are considered. The maximum value for Leibovici's entropy is $\log(I^2)$ where I is the number of categories of the study variable X . The relative version of Leibovici's entropy is obtained by dividing the entropy value by its maximum, and is useful for comparison across datasets with a different number of categories. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation and only couples of non-NA observations are considered.

Value

a list of four elements:

- `leib` Leibovici's entropy
- `range` the theoretical range of Leibovici's entropy, from 0 to $\log(I^2)$
- `rel.leib` Leibovici's relative entropy
- `probabilities` a table with absolute frequencies and estimated probabilities (relative frequencies) for all couple categories

Moreover, a plot of the dataset is produced. Over the plot, a random point is displayed as a red star, and a circle is plotted around that point. The radius of the circle is set by `ccdist`, so that a visual idea is given about the choice of the distance for building co-occurrences.

Examples

```
#random grid data - high entropy
data=matrix(sample(c("a","b","c"), 400, replace=TRUE), nrow=20)
leibovici(data, cell.size=1, ccdist=2)
#plot data
plot(as.im(data, W=square(nrow(data))),
     col=grDevices::gray(seq(1,0,length.out=length(unique(c(data))))),
     main="", ribbon=TRUE)

#compact grid data - low entropy
data=matrix(sort(sample(c("a","b","c"), 400, replace=TRUE)), nrow=20)
#Note: with sorted data, only some couple categories will be present
leibovici(data, cell.size=1, ccdist=1.5)
#plot data
plot(as.im(data, W=square(nrow(data))),
     col=heat.colors(length(unique(c(data))))),
     main="", ribbon=TRUE)

#point data
data=ppp(x=runif(400), y=runif(400),
         window=square(1), marks=sample(1:4, 400, replace=TRUE))
leibovici(data, ccdist=0.1)
#plot data
plot(data)
```

oneill

O'Neill's entropy.

Description

This function computes O'Neill's entropy for a data matrix (see O'Neill et al, 1988).

Usage

```
oneill(
  data,
  win = spatstat.geom::owin(xrange = c(0, ncol(data)), yrange = c(0, nrow(data))),
  plotout = T
)
```

Arguments

data	A data matrix, can be numeric, factor, character, ...
win	Optional, an object of class <code>owin</code> , the observation window for data plotting
plotout	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

O'Neill's entropy index is based on the transformed variable Z , identifying couples of realizations of the variable of interest:

$$H_O(Z) = \sum p(z_r|C) \log(1/p(z_r|C))$$

where $z_r = (x_i, x_{i'})$ is a generic couple of realizations of the study variable X . The conditioning on C for grid data means that only contiguous couples are considered, i.e. couples of pixels sharing a border. All contiguous couples of realizations of the variable of interest are counted and their relative frequencies are used to compute the index. The maximum value for O'Neill's entropy is $\log(I^2)$ where I is the number of categories of X . The relative version of O'Neill's entropy is obtained by dividing the entropy value by its maximum, and is useful for comparison across datasets with a different number of categories. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation and only couples of non-NA observations are considered.

Value

a list of four elements:

- oneill O'Neill's entropy
- range the theoretical range of O'Neill's entropy, from 0 to $\log(I^2)$
- rel.oneill O'Neill's relative entropy
- probabilities a table with absolute frequencies and estimated probabilities (relative frequencies) for all couple categories

Moreover, a plot of the dataset is produced.

Examples

```
#numeric data, square grid
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
oneill(data)
#plot data
plot(as.im(data, W=square(nrow(data))),
      col=grDevices::gray(seq(1,0,length.out=length(unique(c(data))))),
      main="", ribbon=TRUE)

#character data, rectangular grid
data=matrix(sample(c("a","b","c"), 300, replace=TRUE), nrow=30)
oneill(data)
#plot data
plot(as.im(data, W=owin(xrange=c(0,ncol(data)), yrange=c(0,nrow(data))),
      col=terrain.colors(length(unique(c(data))))),
      main="", ribbon=TRUE)

#data with missing values
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
data=rbind(rep(NA, ncol(data)), data, rep(NA, ncol(data)))
oneill(data)
```

 parredw

Parresol and Edwards' entropy.

Description

Compute Parresol and Edwards' entropy, following Parresol and Edwards (2014), starting from data. References can be found at [SpatEntropy](#).

Usage

```
parredw(
  data,
  win = spatstat.geom::owin(xrange = c(0, ncol(data)), yrange = c(0, nrow(data))),
  plotout = T
)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ...
win	Optional, an object of class <code>owin</code> , the observation window for data plotting
plotout	Logical. Default to TRUE, produces an informative plot as part of the function output.

Details

This index is based on the transformed variable Z identifying couples of realizations of the variable of interest. A distance of interest is fixed: Parresol and Edwards' entropy is thought for areas sharing a border, as O'Neill's entropy. All contiguous couples of realizations of the variable of interest are counted and their relative frequencies are used to compute the index, which is the opposite of O'Neill's entropy. The function is able to work with grids containing missing data, specified as NA values. All NAs are ignored in the computation and only couples of non-NA observations are considered.

Value

a list of four elements:

- `parredw` Parresol and Edwards' entropy
- `range` the theoretical range of Parresol and Edwards' entropy, from $-\log(I^2)$ to 0
- `rel.parredw` Parresol and Edwards' relative entropy (with the same interpretation as O'Neill's relative entropy)
- `probabilities` a table with absolute frequencies and estimated probabilities (relative frequencies) for all couple categories

Moreover, a plot of the dataset is produced.

Examples

```
#numeric data, square grid
data=matrix(sample(1:5, 100, replace=TRUE), nrow=10)
parredw(data)
#plot data
plot(as.im(data, W=square(nrow(data))),
      col=grDevices::gray(seq(1,0,length.out=length(unique(c(data))))),
      main="", ribbon=TRUE)

#character data, rectangular grid
data=matrix(sample(c("a","b","c"), 300, replace=TRUE), nrow=30)
parredw(data)
#plot data
plot(as.im(data, W=owin(xrange=c(0,ncol(data)), yrange=c(0,nrow(data))),
      col=terrain.colors(length(unique(c(data))))),
      main="", ribbon=TRUE)
```

raintrees

Rainforest tree data 1.

Description

A marked point pattern dataset about four rainforest tree species: *Acalypha diversifolia*, *Chamguava schippii*, *Inga pezizifera* and *Rinorea sylvatica*

Usage

```
raintrees
```

Format

A ppp object (see package spatstat) with 7251 points, containing:

window An object of type owin (see package spatstat), the 1000x500 metres observation area

x, y Numeric vectors with points' coordinates

marks A character vector matching the tree species to the data points

Details

This dataset documents the presence of tree species over Barro Colorado Island, Panama. Barro Colorado Island has been the focus of intensive research on lowland tropical rainforest since 1923 (<http://www.ctfs.si.edu>). Research identified several tree species over a rectangular observation window of size 1000x500 metres; the tree species constitute the point data categorical mark. This dataset presents 4 species with different spatial configurations: *Acalypha diversifolia*, *Chamguava schippii*, *Inga pezizifera* and *Rinorea sylvatica*. The overall dataset has a total number of 7251 points. The dataset is analyzed with spatial entropy measures in Altieri et al (2018) (references can be found at SpatEntropy).

Source

<http://www.ctfs.si.edu>

Examples

```

data(raintrees)
#plot(raintrees, main="", pch=16, cols=1:4)

#shannon's entropy of the four trees
shannon(raintrees)

#shannon's entropy of Z (tree pairs)
shannonZ(raintrees)

#leibovici's entropy
raintrees$window #to check size and unit of measurement
#example run on a subset of the data to speed up computations
subdata=raintrees[owin(c(0,200),c(0,100))]; plot(subdata)
outp=leibovici(subdata, ccdist=10, verbose=TRUE)
#do not worry about warnings like "data contain duplicated points": since
#coordinates are rounded to the first decimal place, it looks like
#some trees are overlapping when they are just very close.
#Entropy computation works properly anyway

#altieri's entropy
#example run on a subset of the data to speed up computations
subdata=raintrees[owin(c(0,200),c(0,100))]; plot(subdata)
outp=altieri(subdata, distbreak=c(1,2,5,10), verbose=TRUE)

#batty's entropy
#on all points, with a random partition in 10 sub-areas
batty.ent=batty(unmark(raintrees), partition=10)
#plot with partition
#plot(unmark(raintrees), pch=16, cex=0.6, main="")
#plot(batty.ent$area.tess, add=TRUE, border=2, lwd=2)
#on a specific tree species, with a random partition in 6 sub-areas
unique(marks(raintrees)) #to check the species' names
#plot(split.ppp(raintrees), main="") #to plot by species
batty.ent=batty(raintrees, category="cha2sc", partition=6)
#plot with partition
#plot(split.ppp(raintrees)$cha2sc, pch=16, cex=0.6, main="")
#plot(batty.ent$area.tess, add=TRUE, border=2)

#batty's entropy with a partition based on the covariate,
#exploiting spatstat functions
data(raintreesCOV)
#plot(raintreesCOV$grad, main="", col=gray(seq(1,0,l=100)))
data=split.ppp(raintrees)$acaldi
#plot(data, add=TRUE, pch=16, cex=0.6, main="")
#discretize the covariate
slopecut=cut(raintreesCOV$grad,
             breaks = quantile(raintreesCOV$grad, probs = (0:4)/4),

```

```

        labels = 1:4)
maskv=tiles=list()
for(ii in 1:nlevels(slopecut))
{
maskv[[ii]]=as.logical(c(slopecut$v)==levels(slopecut)[ii])
tiles[[ii]]=owin(xrange=data>window$xrange,
                 yrange=data>window$yrange,
                 mask=matrix(maskv[[ii]],nrow(slopecut$v)))
}
slopetess=list(tiles=tiles, n=nlevels(slopecut))
#plot(slopecut, main = "", col=gray(seq(1,0.4,l=4)))
#plot(data, add=TRUE, pch=16, cex=0.6, main="", col=1)

batty(data, partition=slopetess)

#karlstrom and ceccato's entropy
#on a specific tree species, with a random partition in 6 sub-areas
unique(marks(raintrees)) #to check the species' names
#plot(split.ppp(raintrees), main="") #to plot by species
KC.ent=karlstrom(raintrees, category="rinosy", partition=6, neigh=3)
#plot with partition
#plot(split.ppp(raintrees)$rinosy, pch=16, cex=0.6, main="")
#plot(KC.ent$area.tess, add=TRUE, border=2)

```

raintrees2

Rainforest tree data 2.

Description

A marked point pattern dataset about four rainforest tree species: *Astronium graveolens*, *Beilschmiedia pendula*, *Heisteria concinna* and *Inga sapindoides*.

Usage

```
raintrees2
```

Format

A ppp object (see package spatstat) with 5639 points, containing:

window An object of type owin (see package spatstat), the 1000x500 metres observation area

x, y Numeric vectors with points' coordinates

marks A character vector matching the tree species to the data points

Details

This dataset documents the presence of tree species over Barro Colorado Island, Panama. Barro Colorado Island has been the focus of intensive research on lowland tropical rainforest since 1923 (<http://www.ctfs.si.edu>). Research identified several tree species over a rectangular observation window of size 1000x500 metres; the tree species constitute the point data categorical mark. This dataset presents 4 species with different spatial configurations: *Astronium graveolens*, *Beilschmiedia pendula*, *Heisteria concinna* and *Inga sapindoides*. The overall dataset has a total number of 5639 points.

Source

<http://www.ctfs.si.edu>

Examples

```
data(raintrees2)
#plot(raintrees2, main="", pch=16, cols=1:4)

#shannon's entropy of the four trees
shannon(raintrees2)

#shannon's entropy of Z (tree pairs)
shannonZ(raintrees2)

#leibovici's entropy
raintrees2$window #to check size and unit of measurement
#example run on a subset of the data to speed up computations
subdata=raintrees2[owin(c(0,200),c(0,100))]; plot(subdata)
outp=leibovici(subdata, ccdist=10, verbose=TRUE)
#do not worry about warnings like "data contain duplicated points": since
#coordinates are rounded to the first decimal place, it looks like
#some trees are overlapping when they are just very close.
#Entropy computation works properly anyway

#altieri's entropy
#example run on a subset of the data to speed up computations
subdata=raintrees2[owin(c(0,200),c(0,100))]; plot(subdata)
outp=altieri(subdata, distbreak=c(1,2,5,10), verbose=TRUE)

#batty's entropy
#on all points, with a random partition in 10 sub-areas
batty.ent=batty(unmark(raintrees2), partition=10)
#plot with partition
#plot(unmark(raintrees2), pch=16, cex=0.6, main="")
#plot(batty.ent$area.tess, add=TRUE, border=2, lwd=2)
#on a specific tree species, with a random partition in 6 sub-areas
unique(marks(raintrees2)) #to check the species' names
#plot(split.ppp(raintrees2), main="") #to plot by species
batty.ent=batty(raintrees2, category=levels(marks(raintrees2))[1], partition=6)
#plot with partition
#plot(split.ppp(raintrees2)[[1]], pch=16, cex=0.6, main="")
```



```

#plot(batty.ent$area.tess, add=TRUE, border=2)

#batty's entropy with a partition based on the covariate,
#exploiting spatstat functions
data(raintreesCOV)
#plot(raintreesCOV$grad, main="", col=gray(seq(1,0,1=100)))
data=split.ppp(raintrees2)[[1]]
#plot(data, add=TRUE, pch=16, cex=0.6, main="")
#discretize the covariate
slopecut=cut(raintreesCOV$grad,
             breaks = quantile(raintreesCOV$grad, probs = (0:4)/4),
             labels = 1:4)
maskv=tiles=list()
for(ii in 1:nlevels(slopecut))
{
maskv[[ii]]=as.logical(c(slopecut$v)==levels(slopecut)[ii])
tiles[[ii]]=owin(xrange=data$window$xrange,
                 yrange=data$window$yrange,
                 mask=matrix(maskv[[ii]],nrow(slopecut$v)))
}
slopetess=list(tiles=tiles, n=nlevels(slopecut))
#plot(slopecut, main = "", col=gray(seq(1,0.4,1=4)))
#plot(data, add=TRUE, pch=16, cex=0.6, main="", col=1)

batty(data, partition=slopetess)

#karlstrom and ceccato's entropy
#on a specific tree species, with a random partition in 6 sub-areas
unique(marks(raintrees2)) #to check the species' names
#plot(split.ppp(raintrees2), main="") #to plot by species
KC.ent=karlstrom(raintrees2, category=levels(marks(raintrees2))[2], partition=6, neigh=3)
#plot with partition
#plot(split.ppp(raintrees2)[[2]], pch=16, cex=0.6, main="")
#plot(KC.ent$area.tess, add=TRUE, border=2)

```

raintreesCOV

Covariates for the rainforest tree data.

Description

A list of two pixel images with covariates altitude and soil slope for the rainforest tree data 1 and 2, i.e. [raintrees](#) and [raintrees2](#)

Usage

```
raintreesCOV
```

Format

A list of two elements:

elev An object of type `im` (see package `spatstat`), the soil elevation

grad An object of type `im` (see package `spatstat`), the soil slope (gradient of elevation)

Details

For details of the point datasets, see [raintrees](#) and [raintrees2](#). This accompanying dataset gives information about the elevation in the study region. It is a list containing two pixel images, `elev` (elevation in metres) and `grad` (norm of elevation gradient). These pixel images are objects of class `im`. Covariate values are continuous. Once discretized as wished, they can turn into categorical datasets for the computation of all entropy measures. Moreover, they can be used to build sensible sub-areas for Batty's and Karlstrom and Ceccato's entropies (see the examples).

Source

<http://www.ctfs.si.edu>

Examples

```
data(raintreesCOV)
plot(raintreesCOV, main="")
```

shannon

Shannon's entropy.

Description

This function computes Shannon's entropy of a variable X with a finite number of categories. Shannon's entropy is a non-spatial measure.

Usage

```
shannon(data)
```

Arguments

`data` A data matrix or vector, can be numeric, factor, character, ... Alternatively, a marked `ppp` object.

Details

Shannon's entropy measures the heterogeneity of a set of categorical data. It is computed as

$$H(X) = \sum p(x_i) \log(1/p(x_i))$$

where $p(x_i)$ is the probability of occurrence of the i -th category, here estimated, as usual, by its relative frequency. This is both the non parametric and the maximum likelihood estimator for entropy. Shannon's entropy varies between 0 and $\log(I)$, I being the number of categories of the variable under study. The relative version of Shannon's entropy, i.e. the entropy divided by $\log(I)$, is also computed, under the assumption that all data categories are present in the dataset. The relative entropy is useful for comparison across datasets with different I . The function is able to work with lattice data with missing data, as long as they are specified as NAs: missing data are ignored in the computations.

Value

a list of four elements:

- shann Shannon's entropy
- range The theoretical range of Shannon's entropy, from 0 to $\log(I)$
- rel.shann Shannon's relative entropy
- probabilities a table with absolute frequencies and estimated probabilities (relative frequencies) for all data categories

Examples

```
#NON SPATIAL DATA
shannon(sample(1:5, 50, replace=TRUE))

#POINT DATA
#requires marks with a finite number of categories
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
shannon(marks(data.pp))

#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
shannon(data.lat)
```

shannonZ

Shannon's entropy of the transformed variable Z.

Description

This function computes Shannon's entropy of variable Z , where Z identifies pairs of realizations of the variable of interest.

Usage

```
shannonZ(data)
```

Arguments

`data` A data matrix or vector, can be numeric, factor, character, ... Alternatively, a marked ppp object.

Details

Many spatial entropy indices are based on the transformation Z of the study variable, i.e. on pairs (unordered couples) of realizations of the variable of interest. 'Unordered couples' means that the relative spatial location is irrelevant, i.e. that a couple where category i occurs at the left of category j is identical to a couple where category j occurs at the left of category i . When all possible pairs occurring within the observation areas are considered, Shannon's entropy of the variable Z may be computed as

$$H(Z) = \sum p(z_r) \log(1/p(z_r))$$

where $p(z_r)$ is the probability of the r -th pair of realizations, here estimated by its relative frequency. Shannon's entropy of Z varies between 0 and $\log(R)$, $R = \text{binom}(n+1, 2)$ (where n is the number of observations) being the number of possible pairs of categories of the variable under study. The function is able to work with lattice data with missing data, as long as they are specified as NAs: missing data are ignored in the computations.

Value

a list of three elements:

- `shannZ` Shannon's entropy of Z
- `range` The theoretical range of Shannon's entropy of Z , from 0 to $\log(R)$
- `rel.shannZ` Shannon's relative entropy of Z
- `probabilities` a table with absolute frequencies and estimated probabilities (relative frequencies) for all Z categories (data pairs)

Examples

```
#NON SPATIAL DATA
shannonZ(sample(1:5, 50, replace=TRUE))

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
shannonZ(marks(data.pp))

#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
shannonZ(data.lat)
```

Description

The heterogeneity of spatial data presenting a finite number of categories can be measured via computation of spatial entropy. Functions are available for the computation of the main entropy and spatial entropy measures in the literature. They include the traditional version of Shannon's entropy, Batty's spatial entropy, O'Neill's entropy, Li and Reynolds' contagion index, Karlstrom and Ceccato's entropy, Leibovici's entropy, Parresol and Edwards' entropy and Altieri's entropy. The package is able to work with lattice and point data. A step-by-step guide for new users can be found in the first referenced article.

Details

References:

- ALTIERI L., COCCHI D., ROLI G. (2021). Spatial entropy for biodiversity and environmental data: The R-package SpatEntropy. *Environmental Modelling and Software*
- ALTIERI L., COCCHI D., ROLI G. (2019). Advances in spatial entropy measures. *Stochastic Environmental Research and Risk Assessment*
- ALTIERI L., COCCHI D., ROLI G. (2019). Measuring heterogeneity in urban expansion via spatial entropy. *Environmetrics*, 30(2), e2548
- ALTIERI L., COCCHI D., ROLI G. (2018). A new approach to spatial entropy measures. *Environmental and Ecological Statistics*, 25(1), 95-110
- Altieri, L., D. Cocchi, and G. Roli (2017). The use of spatial information in entropy measures. arXiv:1703.06001
- Batty, M. (1974). Spatial entropy. *Geographical Analysis* 6, 1-31.
- Batty, M. (1976). Entropy in spatial aggregation. *Geographical Analysis* 8, 1-21.
- EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>.
- Karlstrom, A. and V. Ceccato (2002). A new information theoretical measure of global and local spatial association. *The Review of Regional Research* 22, 13-40.
- Leibovici, D. (2009). Defining spatial entropy from multivariate distributions of co-occurrences. Berlin, Springer: In K. S. Hornsby et al. (eds.): 9th International Conference on Spatial Information Theory 2009, *Lecture Notes in Computer Science* 5756, 392-404.
- Li, H. and J. Reynolds (1993). A new contagion index to quantify spatial patterns of landscapes. *Landscape Ecology* 8(3), 155-162.
- O'Neill, R., J. Krummel, R. Gardner, G. Sugihara, B. Jackson, D. DeAngelis, B. Milne, M. Turner, B. Zygmunt, S. Christensen, V. Dale, and R. Graham (1988). Indices of landscape pattern. *Landscape Ecology* 1(3), 153-162.
- Parresol, B. and L. Edwards (2014). An entropy-based contagion index and its sampling properties for landscape analysis. *Entropy* 16(4), 1842-1859.

Shannon, C. (1948). A mathematical theory of communication. Bell Dyditem Technical Journal 27, 379-423, 623-656.

turin

Turin urban data.

Description

A lattice dataset with Turin's Urban Morphological Zones (UMZ, see EEA, 2011).

Usage

turin

Format

A matrix with 111 rows and 113 columns. Values are either 0 (non-urban) or 1 (urban). Pixels outside the administrative borders are classified as NA.

Details

This raster/pixel/lattice dataset comes from the EU CORINE Land Cover project (EEA, 2011) and is dated 2011. It is the result of classifying the original land cover data into urbanised and non-urbanised zones, known as 'Urban Morphological Zones' (UMZ, see EEA, 2011). UMZ data are useful to identify shapes and patterns of urban areas, and thus to detect what is known as urban sprawl. Turin's metropolitan area is extracted from the European dataset and is composed by the municipality of Turin and the surrounding municipalities: Beinasco, Venaria Reale, San Mauro Torinese, Grugliasco, Borgaro Torinese, Collegno, Pecetto Torinese, Pino Torinese, Moncalieri, Nichelino, Settimo Torinese, Baldissero Torinese, Rivoli, Orbassano. The dataset is made of 111x113 pixels of size 250x250 metres.

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(turin)
#plot(as.im(turin), main="", col=gray(c(0.8,0)), ribbon=FALSE)

#shannon's entropy
shannon(turin)

#shannon's entropy of Z (urban/non-urban pairs)
shannonZ(turin)

#oneill's entropy
oneill(turin)
```

```

#leibovici's entropy only on Collegno's municipality
data(turinTess)
cell.size=250; ncl=ncol(turin); nrw=nrow(turin)
coords=expand.grid(rev(seq(cell.size/2, (nrw*cell.size-cell.size/2), l=nrw)),
                    seq(cell.size/2, (ncl*cell.size-cell.size/2), l=ncl))
data.pp=ppp(x=coords[which(!is.na(c(turin))),2],
            y=coords[which(!is.na(c(turin))),1],
            window=owin(xrange=c(0, ncl*cell.size), yrange=c(0,nrw*cell.size)),
            marks=c(turin)[which(!is.na(c(turin)))])
data=data.pp[turinTess$tiles[[which(turinTess$names=="Collegno")]]]
#plot(data, pch=16, cex=0.4)
outp=leibovici(data, cell.size=250, ccdist=400, verbose=TRUE)

#altieri's entropy only on Collegno's municipality
outp=altieri(data, cell.size=250, distbreak=c(cell.size, 2*cell.size), verbose=TRUE)

#batty's entropy
#on all points, with a random partition in 10 sub-areas
batty.ent=batty(turin, cell.size=250, partition=10)
#plot with partition
data(turinW)
#plot(as.im(turin, W=turinW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#plot(batty.ent$area.tess, add=TRUE, border=2)

#batty's entropy with a partition based on the administrative areas
data(turinTess)
batty.ent=batty(turin, cell.size=250, partition=turinTess)
#plot(as.im(turin, W=turinW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#for(i in 1:turinTess$n) plot(turinTess$tiles[[i]], add=TRUE, border=2)

#karlstrom and ceccato's entropy
data(turinW)
KC.ent=karlstrom(turin, cell.size=250, partition=15, neigh=3)
#plot with partition
#plot(as.im(turin, W=turinW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#plot(KC.ent$area.tess, add=TRUE, border=2)

#karlstrom and ceccato's entropy with a partition based on the administrative areas
data(turinTess)
KC.ent=karlstrom(turin, cell.size=250, partition=turinTess, neigh=5000, method="distance")
#plot(as.im(turin, W=turinW), main="", col=gray(c(0.8,0)), ribbon=FALSE)
#for(i in 1:turinTess$n) plot(turinTess$tiles[[i]], add=TRUE, border=2)

```

Description

City borders of all municipalities included in the Turin dataset, in the format of polygonal windows owin objects.

Usage

```
turinTess
```

Format

A list of three:

- tiles a list of 15, each element is a owin object with the administrative border of one municipality of Turin's dataset
- n the number of municipalities
- names the names of the 15 municipalities, in the same order as the windows

Details

The object contains a list of 15 observation windows created as owin objects based on the coordinates of the border polygons, for each municipality. See ?owin for details. The object also contains the names of the municipalities, in Italian. Examples on the usefulness of the administrative borders can be found at the topic [turin](#).

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(turin); data(turinW); data(turinTess)
plot(turinW, col=c("black", "white"), main="")
plot(turinTess$tiles[[1]],border=2, add=TRUE, lwd=2)
for(ll in 2:turinTess$n) plot(turinTess$tiles[[ll]],border=2, add=TRUE, lwd=2)

plot(as.im(turin, W=turinW), main="", col=gray(c(0.85,0.4)), ribbon=FALSE)
plot(turinTess$tiles[[1]],border=1, add=TRUE, lwd=2)
for(ll in 2:turinTess$n) plot(turinTess$tiles[[ll]],border=1, add=TRUE, lwd=2)

#see examples under the topic "turin"
```

turinW	<i>Observation window for Turin urban data.</i>
--------	---

Description

An owin object with the city border for the Turin dataset.

Usage

```
turinW
```

Format

An owin object. Units are given in metres; the basic image unit is a 250x250 metres pixels.

Details

This observation window is an owin object created as a binary mask. See ?owin for details. Examples on the usefulness of the window can be found at the topic [turin](#).

Source

EEA (2011). Corine land cover 2000 raster data. Technical Report, downloadable at <http://www.eea.europa.eu/data-and-maps/data/corine-land-cover-2000-raster-1>

Examples

```
data(turinW)
plot(turinW, col=c("red", "white"), main="")
plot(as.im(turin, W=turinW), main="", col=gray(c(0.8,0)), ribbon=FALSE, add=TRUE)

#see examples under the topic "turin"
```

varshannon	<i>Estimated variance of Shannon's entropy.</i>
------------	---

Description

This function estimates the variance of Shannon's entropy of a variable X .

Usage

```
varshannon(data)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... Alternatively, a marked ppp object.
------	---

Details

`varshannon` estimates the variance of the maximum likelihood estimator of Shannon's entropy given by `shannon`. The variance is

$$V(H(X)) = H(X)_2 - H(X)^2$$

, where $H(X)_2$ is a version of Shannon's entropy (see `shannon`) where the information function $\log(1/p(x_i))$ is squared:

$$H(X)_2 = \sum p(x_i) \log(1/p(x_i))^2$$

. The function is able to work with lattice data with missing data, as long as they are specified as NAs: missing data are ignored in the computations.

Value

the estimated variance of Shannon's entropy.

Examples

```
#NON SPATIAL DATA
varshannon(sample(1:5, 50, replace=TRUE))

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
varshannon(marks(data.pp))

#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
varshannon(data.lat)
```

varshannonZ

Estimated variance of Shannon's entropy of Z.

Description

This function estimates the variance of Shannon's entropy of Z , where Z identifies pairs of categories of the original study variable.

Usage

```
varshannonZ(data)
```

Arguments

`data` A data matrix or vector, can be numeric, factor, character, ... Alternatively, a marked ppp object.

Details

`varshannonZ` estimates the variance of the maximum likelihood estimator of Shannon's entropy of Z given by `shannonZ`. The variance is

$$V(H(Z)) = H(Z)_2 - H(Z)^2$$

, where

$$H(Z)_2 = \sum p(z_r) \log(1/p(z_r))^2$$

. The function is able to work with lattice data with missing data, as long as they are specified as NAs: missing data are ignored in the computations.

Value

the estimated variance of Shannon's entropy of Z .

Examples

```
#NON SPATIAL DATA
data=sample(1:5, 50, replace=TRUE)
varshannonZ(data)

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
varshannonZ(marks(data.pp))

#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
varshannonZ(data.lat)
```

Index

* datasets

- bologna, [9](#)
 - bolognaTess, [10](#)
 - bolognaW, [11](#)
 - raintrees, [21](#)
 - raintrees2, [23](#)
 - raintreesCOV, [25](#)
 - turin, [30](#)
 - turinTess, [31](#)
 - turinW, [33](#)
- [altieri](#), [2](#)
- [areapart](#), [4](#), [7](#), [14](#)
- [batty](#), [6](#), [15](#)
- [battyLISA \(karlstrom\)](#), [13](#)
- [bologna](#), [9](#), [11](#), [12](#)
- [bolognaTess](#), [10](#)
- [bolognaW](#), [11](#)
- [contagion](#), [12](#)
- [karlstrom](#), [13](#)
- [leibovici](#), [16](#)
- [oneill](#), [18](#)
- [parredw](#), [20](#)
- [raintrees](#), [21](#), [25](#), [26](#)
- [raintrees2](#), [23](#), [25](#), [26](#)
- [raintreesCOV](#), [25](#)
- [shannon](#), [26](#), [34](#)
- [shannonZ](#), [3](#), [27](#), [35](#)
- [shannonZ\(\)](#), [4](#)
- [SpatEntropy](#), [6](#), [13](#), [29](#)
- [SpatEntropy-package \(SpatEntropy\)](#), [29](#)
- [turin](#), [30](#), [32](#), [33](#)
- [turinTess](#), [31](#)
- [turinW](#), [33](#)
- [varshannon](#), [33](#), [34](#)
- [varshannonZ](#), [34](#), [35](#)