Package 'SpatEntropy'

February 28, 2018

Type Package

Title Spatial Entropy Measures

Version 0.1.0

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

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

Depends R (>= 3.1.0), spatstat

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. References for all measures can be found under the topic 'SpatEntropy'. The package is able to work with lattice and point data.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-28 16:51:14 UTC

R topics documented:

`adj_list` *Adjacency list for spatial entropy.*

Description

`adj_list` builds a list of adjacency matrices for the computation of Altieri's entropy, one for each possible distance range within the observation area.

Usage

```
adj_list(dist.mat, dist.breaks)
```

Arguments

`dist.mat`	An upper-triangular matrix of Euclidean distances, as returned by `euclid_dist()`.
`dist.breaks`	A numeric vector with the breaks for the partition of the maximum distance into distance classes. It must start at 0 and end at the maximum possible distance within the observation area.

Details

This function is needed for the computation of Altieri's spatial entropy. After defining the distance classes in which the observation window has to be partitioned, for each class ad adjacency matrix is constructed. Each adjacency matrix identifies what pairs of points/areas fall within the specific distance range is the basis for the computation of the local term of Altieri's spatial entropy.

Value

A list of length `length(dist.breaks)-1`; each element is an upper-triangular adjacency matrix as returned by `adj_mat()` according to the corresponding distance range.

Examples

```
dist.breaks=c(0,1,2,5,10*sqrt(2))
dist.mat=euclid_dist(coords_pix(square(10), nrow=10, ncol=10))
my.adj.list=adj_list(dist.mat, dist.breaks)
```

---

adj_mat *Adjacency matrix.*

Description

adj_mat builds an upper-triangular adjacency matrix for the set of points/areas in a chosen distance range.

Usage

```
adj_mat(dist.mat, dd0 = 0, dd1)
```

Arguments

dist.mat	An upper-triangular matrix of Euclidean distances, as returned by euclid_dist().
dd0	Numeric, minimum distance for the neighbourhood/couples/pairs, 0 by default.
dd1	Numeric, maximum distance for the neighbourhood/couples/pairs.

Details

The adjacency matrix is a square matrix, with each row corresponding to a point/area, and with 1 values along the row marking the points/areas that are considered 'neighbours' or 'pairing/coupling'. In the context of spatial entropy, an adjacency matrix may take two roles. If Karlstrom and Ceccato's entropy is computed, the adjacency matrix identifies what areas are neighbours, i.e. what areas enter the computation of the local entropy of a specific area. If a spatial entropy based on the transformed variable Z is computed (see shannonZ for details on Z), the adjacency matrix identifies what pairs/couples of points/areas must be considered for the computation, according to the chosen distance range of interest.

Value

An nxn upper-triangular adjacency matrix (where n in the data vector length) with value 1 if two units are neighbours or form a couple/pair, 0 otherwise.

Examples

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
plot(cbind(rep(1:5, each=5), rep(1:5,5)))
adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
adj_mat(dist.mat, 1, 3)
```

areapart *Area partition.*

Description

This function partitions the observation area in a number of sub-areas, and assigns the data points/pixels to the areas.

Usage

```
areapart(win, G, data.coords)
```

Arguments

win	The observation area, an object of class owin, see package spatstat.
G	An integer if sub-areas are randomly generated, determining the number G of sub-areas. Alternatively, a Gx2 matrix with the sub-areas centroids' coordinates.
data.coords	A two column matrix. If the dataset is a point pattern, the point coordinates. If the dataset is a raster/pixel matrix, the centroids' coordinates, provided by user or returned by coords_pix().

Details

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.

Value

A list with elements:

- G.coords a point pattern containing the G areas' centroids
- data.assign a three column matrix, where each pair of data coordinates is matched to one of the G areas (numbered 1 to G).

Examples

```
#LATTICE DATA
#random generation of areas
ccc=coords_pix(area=square(10), nrow=10, ncol=10)
partition=areapart(square(10), G=5, data.coords=ccc)

#providing a pre-fixed area partition
win=square(10)
G=5
GG=cbind(runif(G, win$xrange[1], win$xrange[2]),
         runif(G, win$yrange[1], win$yrange[2]))
```

```
ccc=coords_pix(area=win, pixel.xsize = 2, pixel.ysize = 2)
partition=areapart(win, G=GG, data.coords=ccc)

#POINT DATA
#random generation of areas
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
ccc=coords(data.pp)
partition=areapart(square(10), G=4, data.coords=ccc)

#providing a pre-fixed area partition
win=square(10)
G=4
GG=cbind(runif(G, win$xrange[1], win$xrange[2]),
         runif(G, win$yrange[1], win$yrange[2]))
data.pp=runifpoint(100, win=win)
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
ccc=coords(data.pp)
partition=areapart(win, G=GG, data.coords=ccc)

#for plotting the area partiton
?plot_areapart
```

`batty` *Batty's entropy.*

Description

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

Usage

```
batty(data, data.assign, is.pointdata = FALSE, category, win = NULL,
  G.coords)
```

Arguments

`data`	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, `data` is the mark vector.
`data.assign`	A three column matrix, containing the data coordinates (centroids when pixels) and the id of the corresponding sub-area. Provided by user or returned by `areapart()`.
`is.pointdata`	Logical: `T` if data are a point pattern, `F` if they are pixels.
`category`	A character string, the exact name of the category for which Batty's spatial entropy is computed, as in `data`.

win	An owin object (see package spatstat), the observation area. Only needed for lattice data.
G.coords	A point pattern (an object of class ppp see package spatstat), or a two column matrix with the area centroids' coordinates. Provided by user or returned by areapart().

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 starts from the output of areapart() and allows to compute Batty's entropy as

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

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 must be created with the same category for all points.

Value

Batty's spatial entropy value, as well as a table with information about each sub-area:

- area.id the sub-area id
- abs.freq the number of points/pixels presenting the category of interest for each sub-area
- rel.freq the proportion of points/pixels presenting the category of interest in each sub-area, with regard to the total number of points/pixels with the category of interest
- Tg the sub-area size.

Examples

```
#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
ccc=coords_pix(area=square(10), nrow=10, ncol=10)
partition=areapart(square(10), G=5, data.coords=ccc)
batty(data.lat, partition$data.assign, category="a",
win=square(10), G.coords=partition$G.coords)
plot_areapart(partition$data.assign, square(10), is.pointdata=FALSE,
add.data=TRUE, data.bin=TRUE, category="a",
data=data.lat, G.coords=partition$G.coords, main="")

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
ccc=coords(data.pp)
partition=areapart(square(10), G=4, data.coords=ccc)
batty(marks(data.pp), partition$data.assign, is.pointdata=TRUE,
category="b", G.coords=partition$G.coords)
plot_areapart(partition$data.assign, square(10), is.pointdata=TRUE,
```

```
add.data=TRUE, data.bin=TRUE, category="b",
data=data.pp, G.coords=partition$G.coords, main="")
```

`contagion` *Li and Reynolds' relative contagion index.*

Description

This function computes Li and Reynold's contagion index, following Li and Reynolds (1993), starting from data or from the output of `leibovici()`. References can be found at `SpatEntropy`.

Usage

```
contagion(oneill = NULL, n.cat = NULL, data = NULL, adj.mat = NULL,
  missing.cat = NULL, ordered = TRUE)
```

Arguments

`oneill`	O'Neill's entropy as the output of `leibovici()`. If this is provided, nothing else needs to be specified, except for `n.cat` if wished.
`n.cat`	Optional, an integer denoting the number of categories of the study variable.
`data`	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, `data` is the mark vector.
`adj.mat`	The contiguity matrix, upper- or lower-triangular. Provided by user or generated by `adj_mat()`.
`missing.cat`	Optional, a vector with the names of all categories that are absent in `data`.
`ordered`	Logical, `T` if the entropy is computed using ordered couples (see `couple_count()`), `F` if it is computed using pairs (see `pair_count()`).

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. This corresponds to creating a contiguity matrix as done by `adj_mat()`. Then, all couples of realizations of the variable of interest identified by the adjacency matrix are counted and their relative frequencies are used to compute the index, which is $1 - NO$ where NO is the normalized O'Neill's entropy, i.e. O'Neill's entropy divided by its maximum $\log(I)$, I being the number of categories of the variable under study. Couples can be ordered or not (pairs), with ordered couples as the default option since it is the authors' choice. This function also allows to compute contagion for neighbourhood structures different from contiguity, by suitably defining the adjacency matrix.

Value

Li and Reynolds' contagion index, as well as a summary table containing the couples (or pairs) along with their absolute and relative frequencies.

Examples

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
data=sample(1:3, 25, replace=TRUE)
oneill=leibovici(data, adj.mat)

contag=contagion(oneill)

contag=contagion(data=data, adj.mat=adj.mat)
```

`coords_pix` *Pixel coordinates generation.*

Description

`coords_pix` generates the coordinates of the pixel centroids, given an obervation area and the pixel size. The resulting coordinates may be used as arguments of `euclid_dist()`.

Usage

```
coords_pix(area, pixel.xsize = diff(area$xrange)/ncol,
  pixel.ysize = diff(area$yrange)/nrow,
  nrow = diff(area$yrange)/pixel.ysize,
  ncol = diff(area$xrange)/pixel.xsize)
```

Arguments

`area`	The observation area, an `owin` object, see package `spatstat`
`pixel.xsize`	A scalar, length of the pixel side along the x axis (unnecessary if `ncol` is provided)
`pixel.ysize`	A scalar, length of the pixel side along the y axis (unnecessary if `nrow` is provided)
`nrow`	An integer, number of pixels along the y axis (unnecessary if `pixel.ysize` is provided)
`ncol`	An integer, number of pixels along the x axis (unnecessary if `pixel.xsize` is provided)

Details

When the Euclidean distance is computed between areas, the most common choice is to use the area centroid as the reference point. This function returns the centroids of a regular lattice. You can either provide the number of pixels along the two cardinal directions (using `ncol` for the x direction and `nrow` for the y direction), or alternatively provide the pixel size along the two directions (rectangular areas are allowed).

Value

A two column matrix containing the x and y coordinates of the pixel centroids.

Examples

```
ccc=coords_pix(area=square(10), nrow=10, ncol=10)
plot(square(10)); points(ccc)

ccc=coords_pix(area=square(10), pixel.xsize = 2, pixel.ysize = 2)
plot(square(10)); points(ccc, pch=16)
```

`couple_count` *Build ordered couples in a dataset.*

Description

This function builds and counts the number of all types of ordered couples in a data vector or matrix, according to a chosen adjacency matrix. 'Ordered' means that a couple (i,j) is different than a couple (j,i).

Usage

```
couple_count(data, adj.mat, missing.cat = NULL)
```

Arguments

`data`	A matrix or vector, can be numeric, factor, character... If the dataset is a point pattern, `data` is the mark vector.
`adj.mat`	An adjacency matrix, upper- or lower-triangular. Provided by user or generated by `adj_mat`.
`missing.cat`	A vector with the names of all categories that are absent in `data`.

Details

This function needs a data matrix or data vector of any type (numeric, factor, character, ...), and an adjacency matrix as generated by `adj_mat()`. It returns all the data couples identified by the adjacency matrix, i.e. occurring at the chosen neighbourhood distance. Relative and absolute frequencies for all possible couples are returned, and may be used for computation of spatial entropy at the chosen distance range. 'Ordered' means that the relative spatial location is relevant, i.e. that a couple where category i occurs at the left of category j is different from a couple where category j occurs at the left of category i.

Value

The number of couples, and a table with absolute and relative frequencies for each couple of categories.

Examples

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
couple_count(sample(1:3, 25, replace=TRUE), adj.mat)
```

`data_bologna` *Bologna data.*

Description

A lattice dataset with Bologna Urban Morphological Zones.

Usage

```
data_bologna
```

Format

A `matrix` with 120 rows and 96 columns. Values are either 0 (non-urban) or 1 (urban).

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. The Bologna 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 120x96 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

data_rainforest *Rainforest tree data.*

Description

A marked point pattern dataset about four rainforest tree species.

Usage

```
data_rainforest
```

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 (2017) (references can be found at SpatEntropy).

Source

http://www.ctfs.si.edu

euclid_dist *Euclidean distance.*

Description

euclid_dist computes the Euclidean distance between all point couples/pairs identified by two coordinate matrices. Useful alone, or together with adj_mat().

Usage

```
euclid_dist(coords1, coords2 = coords1)
```

Arguments

coords1	A two column matrix containing starting coordinates. Provided by user or as output of coords_pix().
coords2	A two column matrix with containing coordinates, same as coords1 by default.

Details

euclid_dist needs two matrices listing the coordinates of two sets of points. It computes the Euclidean distance between each point of the first matrix and all points of the second matrix. The default option provides the Euclidean distance between all points in a single set.

Value

An upper-triangular distance matrix: a nrow(coords1) x nrow(coords2) matrix with Euclidean distances between points.

Examples

```
euclid_dist(cbind(runif(10), runif(10)))
```

---

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) (references are under the topic SpatEntropy).

Usage

```
karlstrom(data, data.assign, category, G.coords, neigh.dist)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, data is the mark vector.
data.assign	A three column matrix, containing the data coordinates (centroids when pixels) and the id of the corresponding sub-area. Provided by user or returned by areapart().
category	A character string, the exact name of the category for which Karlstrom and Ceccato's spatial entropy is computed, as in data.
G.coords	A point pattern (an object of class ppp see package spatstat), or a two column matrix with the area centroids' coordinates. Provided by user or returned by areapart().
neigh.dist	A scalar, the chosen neighbourhood Euclidean distance.

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. 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 must be created with the same category for all points.

Value

Karlstrom and Ceccato's spatial entropy value as well as a table with information about each sub-area:

- `area.id` the sub-area id
- `abs.freq` the number of points/pixels presenting the category of interest for each sub-area
- `rel.freq` the proportion of points/pixels presenting the category of interest in each sub-area, with regard to the total number of points/pixels with the category of interest
- `p.tilde` the probability of occurrence over area g weighted with its neighbours.

Examples

```
#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
ccc=coords_pix(area=square(10), nrow=10, ncol=10)
partition=areapart(square(10), G=5, data.coords=ccc)
karlstrom(data.lat, partition$data.assign, category="a",
G.coords=partition$G.coords, neigh.dist=2)
plot_areapart(partition$data.assign, square(10), is.pointdata=FALSE,
add.data=TRUE, data.bin=TRUE, category="a",
data=data.lat, G.coords=partition$G.coords, main="")

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
ccc=coords(data.pp)
partition=areapart(square(10), G=4, data.coords=ccc)
karlstrom(marks(data.pp), partition$data.assign,
category="b", G.coords=partition$G.coords, neigh.dist=4)
plot_areapart(partition$data.assign, square(10), is.pointdata=TRUE,
add.data=TRUE, data.bin=TRUE, category="b",
data=data.pp, G.coords=partition$G.coords, main="")
```

leibovici *O'Neill's and 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, adj.mat, missing.cat = NULL, ordered = TRUE)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, data is the mark vector.
adj.mat	An adjacency matrix, upper- or lower-triangular. Provided by user or generated by adj_mat().
missing.cat	Optional, a vector with the names of all categories that are absent in data.
ordered	Logical, T if the entropy is computed using ordered couples (see couple_count()), F if it is computed using pairs (see pair_count()).

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; this corresponds to creating an adjacency (contiguity) matrix as done by adj_mat(). Then, all couples of realizations of the variable of interest identified by the adjacency matrix are counted and their relative frequencies are used to compute the index with the traditional Shannon's formula. Couples can be ordered or not (pairs), with ordered couples as the default option being the authors' choice.

Value

Leibovici's spatial entropy value (O'Neill's entropy when contiguity is considered) as well as a summary table containing the couples (or pairs) along with their absolute and relative frequencies.

Examples

```
##O'NEILL
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
data=sample(1:3, 25, replace=TRUE)
oneill=leibovici(data, adj.mat)

##LEIBOVICI
```

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=4) #for the contiguity matrix
data=sample(1:3, 25, replace=TRUE)
leib=leibovici(data, adj.mat)
```

`pair_count` *Build pairs (unordered couples) in a dataset.*

Description

This function builds and counts the number of all types of pairs in a data vector or matrix, according to a chosen adjacency matrix.

Usage

```
pair_count(data, adj.mat, missing.cat = NULL)
```

Arguments

`data`	A matrix or vector, can be numeric, factor, character... If the dataset is a point pattern, `data` is the mark vector.
`adj.mat`	An adjacency matrix, upper- or lower-triangular. Provided by user or generated by `adj_mat()`.
`missing.cat`	Optional, a vector with the names of all categories that are absent in `data`.

Details

This function needs a data matrix or data vector of any type (numeric, factor, character, ...), and an adjacency matrix as generated by `adj_mat()`. It returns all the data pairs identified by the adjacency matrix, i.e. occurring at the chosen neighbourhood distance. Relative and absolute frequencies for all possible pairs are returned, and may be used for computation of spatial entropy at the chosen distance range. 'Unordered couple', i.e. 'pair', 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.

Value

The number of pairs, and a table with absolute and relative frequencies for each pair of categories.

Examples

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
pair_count(sample(1:3, 25, replace=TRUE), adj.mat)
```

parresol *Parresol and Edwards' entropy.*

Description

Compute Parresol and Edwards' entropy, following Parresol and Edwards (2014), starting from data or from the output of `leibovici()`. References can be found at `SpatEntropy`.

Usage

```
parresol(oneill = NULL, n.cat = NULL, data = NULL, adj.mat = NULL,
  missing.cat = NULL, ordered = TRUE)
```

Arguments

`oneill`	O'Neill's entropy as the output of `leibovici()`. If this is provided, nothing else needs to be specified, except for `n.cat` if wished.
`n.cat`	Optional, an integer denoting the number of categories of the study variable.
`data`	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, `data` is the mark vector.
`adj.mat`	The contiguity matrix, upper- or lower-triangular. Provided by user or generated by `adj_mat()`.
`missing.cat`	Optional, a vector with the names of all categories that are absent in `data`.
`ordered`	Logical, `T` if the entropy is computed using ordered couples (see `couple_count()`), `F` if it is computed using pairs (see `pair_count()`).

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 originally thought for areas sharing a border, as O'Neill's entropy. This corresponds to creating a contiguity matrix as done by `adj_mat()`. Then, all couples of realizations of the variable of interest identified by the adjacency matrix are counted and their relative frequencies are used to compute the index, which is the opposite of O'Neill's entropy. Couples can be ordered or not (pairs), with ordered couples as the default option since it is the authors' choice. This function also allows to compute Parresol and Edwards' entropy for neighbourhood structures different from contiguity, by specifying a suitable adjacency matrix.

Value

Parresol and Edwards' spatial entropy value as well as a summary table containing the couples (or pairs) along with their absolute and relative frequencies.

Examples

```
dist.mat=euclid_dist(cbind(rep(1:5, each=5), rep(1:5,5)))
adj.mat=adj_mat(dist.mat, dd1=dist.mat[1,2]) #for the contiguity matrix
data=sample(1:3, 25, replace=TRUE)
oneill=leibovici(data, adj.mat)

parr=parresol(oneill, n.cat=3)

parr=parresol(data=data, adj.mat=adj.mat)
```

---

plot_areapart *Plot area partition.*

Description

This function plots an area partition into sub-areas, generated by `areapart()`.

Usage

```
plot_areapart(data.assign, win, is.pointdata = FALSE, add.data = FALSE,
  data.bin = FALSE, category = NULL, data, G.coords = NULL, main = "",
  ribbon = TRUE)
```

Arguments

`data.assign`	A three column matrix, containing the data coordinates (centroids, when pixels) and the id of the corresponding sub-area. Provided by user or returned by `areapart()`.
`win`	The observation area, an object of class `owin` (see package `spatstat`).
`is.pointdata`	Logical: `T` if data are a point pattern, `F` if they are pixels.
`add.data`	Logical: `F` (default) if only the area partition is plotted, `T` if data are added to the area partition plot.
`data.bin`	Logical, only used when `add.data=TRUE`: `T` (default) if the plot displays the dichotomized version of the dataset, according to the category of interest.
`category`	A character string. The exact name of the category of interest for Batty's or Karlstrom and Ceccato's spatial entropy, as in `data`. Only used when `add.data=TRUE` and `data.bin=TRUE`.
`data`	A data matrix for lattice data, or a `ppp` object for point data (see package `spatstat`).
`G.coords`	A two column matrix with the coordinates of the sub-areas centroids. Only needed if `data` is a point pattern.
`main`	Optional, a character string with the plot main title.
`ribbon`	Logical, whether to display a ribbon showing the colour map.

Details

This function allows to plot a fixed or randomly generated area partition, such as the one produced by areapart(). The plot changes according to a few options: the partition may be plotted with or without data, with or without colour filling. When data present multiple categories, one can choose to plot the category of interest together with the partition. If the data are points, the Dirichlet tessellation is plotted (see dirichlet in the package spatstat). If the data are pixels, the partition follows the pixel borders.

Value

A plot of the partition in sub-areas, according to the chosen options.

Examples

```
#LATTICE DATA
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
ccc=coords_pix(area=square(10), nrow=10, ncol=10)
partition=areapart(square(10), G=5, data.coords=ccc)
#plot without data
plot_areapart(partition$data.assign, square(10), is.pointdata=FALSE,
add.data=FALSE, data=data.lat, G.coords=partition$G.coords, main="")
#plot with data
plot_areapart(partition$data.assign, square(10), is.pointdata=FALSE,
add.data=TRUE, data=data.lat, G.coords=partition$G.coords, main="")
#plot with data - dichotomize data according to a category of interest
plot_areapart(partition$data.assign, square(10), is.pointdata=FALSE,
add.data=TRUE, data.bin=TRUE, category="a",
data=data.lat, G.coords=partition$G.coords, main="")

#POINT DATA
data.pp=runifpoint(100, win=square(10))
marks(data.pp)=sample(c("a","b","c"), 100, replace=TRUE)
ccc=coords(data.pp)
partition=areapart(square(10), G=4, data.coords=ccc)
#plot without data
plot_areapart(partition$data.assign, square(10), is.pointdata=TRUE,
add.data=FALSE, data=data.pp, G.coords=partition$G.coords, main="")
#plot with data
plot_areapart(partition$data.assign, square(10), is.pointdata=TRUE,
add.data=TRUE, data=data.pp, G.coords=partition$G.coords, main="")
#plot with data - dichotomize data according to a category of interest
plot_areapart(partition$data.assign, square(10), is.pointdata=TRUE,
add.data=TRUE, data.bin=TRUE, category="a",
data=data.pp, G.coords=partition$G.coords, main="")
```

---

plot_lattice *Plot lattice data.*

Description

plot_lattice produces a gray scale plot of a matrix of categorical data.

Usage

```
plot_lattice(data, win = spatstat::owin(xrange = c(1, ncol(data)), yrange =
  c(1, nrow(data))), gray.ext = c(1, 0), main = "", ribbon = TRUE)
```

Arguments

data	A matrix, can be numeric, factor, character.
win	An owin object, the observation area (see package spatstat). Automatically created, if not provided, by setting the pixel size as 1.
gray.ext	A vector of length two with the two extremes of the gray scale (between 0 and 1).
main	Optional, a character string with the plot main title.
ribbon	Logical, whether to display a ribbon showing the colour map.

Details

This function allows to easily produce a gray scale map given a matrix of categorical data and, optionally, the observation area. It ensures that data is displayed following the matrix order (where position (1,1) corresponds to the top-left corner of plot), avoiding risks of row inversion or transposition. A few #'options may be tuned: the extent of the gray scale, the title and the legend.

Value

A gray scale plot of the categorical lattice dataset.

Examples

```
data.lat=matrix(sample(c("a","b","c"), 100, replace=TRUE), nrow=10)
plot_lattice(data.lat)

plot_lattice(data.lat, win=square(100))

plot_lattice(data.lat, win=square(10), gray.ext=c(1,.4), ribbon=FALSE)
```

shannonX *Shannon's entropy.*

Description

This function computes Shannon's entropy of a variable X with a finite number of categories.

Usage

```
shannonX(data)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, data is the mark vector.

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 by its relative frequency. This is both the non parametric and the maximum likelihood estimator. Shannon's entropy varies between 0 and $\log(I)$, I being the number of categories of the variable under study.

Value

Estimated probabilities for all data categories, and Shannon's entropy.

Examples

```
#NON SPATIAL DATA
shannonX(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)
shannonX(marks(data.pp))

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

shannonX_sq *Shannon's entropy with a squared information function.*

Description

This function computes Shannon's entropy of X with the square of the information function.

Usage

```
shannonX_sq(data)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, data is the mark vector.

Details

This computes a version of Shannon's entropy (see shannonX()) where the information function $\log(1/p(x_i))$ is squared:

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

It is useful for estimating the variance of the maximum likelihood estimator of Shannon's entropy given by shannonX().

Value

Estimated probabilities for all data categories, and Shannon's entropy of X with a squared information function.

Examples

```
#NON SPATIAL DATA
shannonX_sq(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)
shannonX_sq(marks(data.pp))

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

Arguments

`data`	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, `data` is the mark vector.
`missing.cat`	Optional, a vector with the names of all categories that are absent in `data`.

Details

Many spatial entropy indices are based on the trasformation 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 being the number of possible pairs of categories of the variable under study.

Value

Estimated probabilities for all Z categories (data pairs), and Shannon's entropy of Z.

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)

#when categories are missing
shannonZ(data.lat, missing.cat=c("d", "e"))
```

---

shannonZ_sq *Shannon's entropy of Z with a squared information function.*

Description

This function computes Shannon's entropy of Z with the square of the information function.

Usage

```
shannonZ_sq(shannZ)
```

Arguments

shannZ	Output of shannonZ()

Details

This computes a version of Shannon's entropy of Z (see shannonZ()) where the information function $\log(1/p(z_r))$ is squared:

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

It is useful for estimating the variance of the maximum likelihood estimator of Shannon's entropy given by shannonZ().

Value

Estimated probabilities for all Z categories (data pairs), and Shannon's entropy of Z with a squared information function.

Examples

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

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

#LATTICE DATA
```

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

SpatEntropy *SpatEntropy: a package for computing spatial entropy measures.*

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.

Details

References:

Altieri, L., D. Cocchi, and G. Roli (2017). A new approach to spatial entropy measures, Environmental and Ecological Statistics published online at http://link.springer.com/article/10.1007/s10651-017-0383-1, pp 1-16.

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.

spat_entropy *Altieri's spatial entropy.*

Description

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

Usage

```
spat_entropy(data, adj.list, shannZ, missing.cat = NULL)
```

Arguments

data	A data matrix or vector, can be numeric, factor, character, ... If the dataset is a point pattern, data is the mark vector.
adj.list	A list of adjacency matrices, provided by user or returned by adj_list(). Each element of the list contains a binary adjacency matrix for building pairs at a specific distance, provided by user or returned by adj_mat().
shannZ	The output of shannonZ(): Shannon's entropy of Z as well as the pairs frequency table.
missing.cat	Optional, a vector with the names of all categories that are absent in data.

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

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

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 using adj_mat(), 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.

Value

A list with elements:

- `mut.global` the spatial mutual information
- `res.global` the global residual entropy
- `shannZ` Shannon's entropy of Z
- `mut.local` the partial information terms
- `res.local` the partial residual entropies
- `pwk` the spatial weights for each distance range
- `pzr.marg` the marginal probability distribution of Z
- `pzr.cond` a list with the conditional probability distribution of Z for each distance range
- `Q` the number of pairs (realizations of Z)
- `Qk` the number of pairs for each distance range.

Examples

```
data=matrix(sample(1:5, 25, replace=TRUE), nrow=5)
plot(as.im(data, W=square(5)))
dist.breaks=c(0,1,2,5,5*sqrt(2))
dist.mat=euclid_dist(coords_pix(square(5), nrow=5, ncol=5))
my.adj.list=adj_list(dist.mat, dist.breaks) #see adj_list
my.shZ=shannonZ(data)
spat_entropy(data=data, adj.list=my.adj.list, shannZ=my.shZ)
```

Index