Package 'bespatial'

May 7, 2026

Title Boltzmann Entropy for Spatial Data

Version 0.1.3

Description Calculates several entropy metrics for spatial data
inspired by Boltzmann's entropy formula.
It includes metrics introduced by Cushman for landscape mosaics
(Cushman (2015) <doi:10.1007/s10980-015-0305-2>),
and landscape gradients and point patterns
(Cushman (2021) <doi:10.3390/e23121616>); by Zhao and Zhang for
landscape mosaics (Zhao and Zhang (2019) <doi:10.1007/s10980-019-00876-x>);
and by Gao et al. for landscape gradients
(Gao et al. (2018) <doi:10.1111/tgis.12315>; Gao and Li (2019) <doi:10.1007/s10980-019-00854-3>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.1)

LinkingTo comat (>= 0.9.2), Rcpp, RcppArmadillo

Imports belg, comat, Rcpp, terra (>= 1.5-13), tibble, landscapemetrics

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://jakubnowosad.com/bespatial/

BugReports https://github.com/Nowosad/bespatial/issues

NeedsCompilation yes

Author Jakub Nowosad [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1057-3721>)

Maintainer Jakub Nowosad <nowosad.jakub@gmail.com>

Repository CRAN

Date/Publication 2025-04-02 14:10:06 UTC

Contents

`bes_g_cushman` *Configurational entropy for surfaces*

Description

Calculates Cushman's configurational entropy for surfaces (2021)

Usage

```
bes_g_cushman(x, nr_of_permutations = 1000, independent = FALSE)
```

Arguments

`x`	SpatRaster, stars, RasterLayer, RasterStack, RasterBrick, matrix, or array containing one or more continuous rasters
`nr_of_permutations`	Number of permutations performed on each input raster to calculate possible distribution of "slope" values
`independent`	Should an independent set of permutations be performed for each input raster? `TRUE/FALSE`. Use `FALSE` (default) when each of your input rasters has the same configuration.

Value

A tibble

References

Cushman, S. A. (2021). Generalizing Boltzmann Configurational Entropy to Surfaces, Point Patterns and Landscape Mosaics. In Entropy (Vol. 23, Issue 12, p. 1616). MDPI AG. https://doi.org/10.3390/e23121616

Examples

```
library(bespatial)
library(terra)
gradient = rast(system.file("raster/gradient.tif", package = "bespatial"),
                    lyrs = 1)
ce2 = bes_g_cushman(gradient, 100)
plot(gradient, main = round(ce2$value, 2))
bes_g_cushman(gradient, 1000, independent = TRUE)
```

---

bes_g_gao *Boltzmann entropy of a landscape gradient*

Description

Calculates the Boltzmann entropy of a landscape gradient by Gao (2017, 2019)

Usage

```
bes_g_gao(
  x,
  method = "aggregation",
  na_adjust = TRUE,
  base = "log10",
  relative = FALSE
)
```

Arguments

x	SpatRaster, stars, RasterLayer, RasterStack, RasterBrick, matrix, or array.
method	A method used. Either "hierarchy" for the hierarchy-based method (Gao et al., 2017) or "aggregation" (default) for the aggregation-based method (Gao et al., 2019).
na_adjust	Should the output value be adjusted to the proportion of not missing cells? Either TRUE (default) or FALSE
base	A logarithm base ("log", "log2" or "log10").
relative	Should a relative or absolute entropy be calculated? TRUE or FALSE (default).

Details

The method for computing the Boltzmann entropy of a landscape gradient works on integer values that are either positive or equals to zero. This function automatically rounds values to the nearest integer value (rounding halfway cases away from zero) and negative values are shifted to positive values.

Value

A tibble

References

Gao, Peichao, Hong Zhang, and Zhilin Li. "A hierarchy-based solution to calculate the configurational entropy of landscape gradients." Landscape Ecology 32.6 (2017): 1133-1146.

Gao, Peichao, Hong Zhang, and Zhilin Li. "An efficient analytical method for computing the Boltzmann entropy of a landscape gradient." Transactions in GIS (2018).

Gao, Peichao and Zhilin Li. "Aggregation-based method for computing absolute Boltzmann entropy of landscape gradient with full thermodynamic consistency" Landscape Ecology (2019)

Examples

```
library(terra)
library(bespatial)
gradient = rast(system.file("raster/gradient.tif", package = "bespatial"))
gg1 = bes_g_gao(gradient)
plot(gradient, main = round(gg1$value, 2))
```

---

bes_m_cushman *Configurational entropy for landscape mosaics*

Description

Calculates Cushman's configurational entropy for landscape mosaics (2015)

Usage

```
bes_m_cushman(x, nr_of_permutations, independent = FALSE)
```

Arguments

x	SpatRaster, stars, RasterLayer, RasterStack, RasterBrick, matrix, or array containing one or more categorical rasters
nr_of_permutations	Number of permutations performed on each input raster to calculate possible distribution of total edge values
independent	Should an independent set of permutations be performed for each input raster? TRUE/FALSE. Use FALSE (default) when each of your input rasters has the same configuration (proportion of categories).

Value

A tibble

References

Cushman, S. A. (2015). Calculating the configurational entropy of a landscape mosaic. In Landscape Ecology (Vol. 31, Issue 3, pp. 481–489). Springer Science and Business Media LLC. https://doi.org/10.1007/s10980-015-0305-2

Examples

```
library(terra)
library(bespatial)
mosaic = rast(system.file("raster/mosaic.tif", package = "bespatial"))
ce1 = bes_m_cushman(mosaic, 1000)
plot(mosaic, main = round(ce1$value, 2))
bes_m_cushman(mosaic, 1000, independent = TRUE)
```

---

bes_m_zhao *Zhao's entropy for landscape mosaics*

Description

Calculates Zhao's entropy for landscape mosaics based on the Wasserstein metric (2019)

Usage

```
bes_m_zhao(x, neighbourhood = 4)
```

Arguments

x	SpatRaster, stars, RasterLayer, RasterStack, RasterBrick, matrix, or array containing one or more categorical rasters
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case), 8 (queen's case)

Value

A tibble

References

Zhao, Y., & Zhang, X. (2019). Calculating spatial configurational entropy of a landscape mosaic based on the Wasserstein metric. Landscape Ecology, 34(8), 1849-1858. https://doi.org/10.1007/s10980-019-00876-x

Examples

```
library(terra)
library(bespatial)
mosaic = rast(system.file("raster/mosaic.tif", package = "bespatial"))
w_dists1 = bes_m_zhao(mosaic)
plot(mosaic, main = round(w_dists1$value, 2))
```

bes_p_cushman *Configurational entropy for point patterns*

Description

Calculates Cushman's configurational entropy for point patterns (2021)

Usage

```
bes_p_cushman(x, nr_of_permutations, independent = FALSE)
```

Arguments

x	SpatRaster, stars, RasterLayer, RasterStack, RasterBrick, matrix, or array containing one or more rasters with one value and NAs
nr_of_permutations	Number of permutations performed on each input raster to calculate possible distribution of the number of nearest neighbors
independent	Should an independent set of permutations be performed for each input raster? TRUE/FALSE. Use FALSE (default) when each of your input rasters has the same configuration.

Value

A tibble

References

Cushman, S. A. (2021). Generalizing Boltzmann Configurational Entropy to Surfaces, Point Patterns and Landscape Mosaics. In Entropy (Vol. 23, Issue 12, p. 1616). MDPI AG. https://doi.org/10.3390/e23121616

Examples

```
library(terra)
library(bespatial)
point_pattern = rast(system.file("raster/point_pattern.tif", package = "bespatial"))
ce3 = bes_p_cushman(point_pattern, 100)
plot(point_pattern, main = round(ce3$value, 2))
ce3b = bes_p_cushman(point_pattern, 100, independent = TRUE)
plot(point_pattern, main = round(ce3b$value, 2))
```

`get_distance` *Calculates an average distance between non-NA cells*

Description

Calculates an average distance between non-NA cells

Usage

```
get_distance(p, x)
```

Arguments

`p`	A matrix
`x`	A SpatRaster with proper metadata (e.g., extent and CRS)

Details

It converts permuted matrix into a vector dataset, and calculates an average distance between the points

Value

An average distance between points

`get_slope` *Calculate a slope*

Description

Calculate a slope

Usage

```
get_slope(x, neighbourhood = matrix(4))
```

Arguments

`x`	A matrix
`neighbourhood`	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case), 8 (queen's case) or a binary matrix where the ones define the neighbourhood. The default is 4.

Details

"Slope" is calculated as follows:

1. For each cell, the algorithm looks at its 4 neighbors and calculates the absolute difference between the main cell and its neighbors.
2. Next, it sums these four values.
3. After repeating this operation for every cell, it calculates an average of the sum of the absolute differences for the whole raster.

Value

A slope value

`get_total_edge` *Calculate total edge based on the input matrix*

Description

Calculate total edge based on the input matrix

Usage

```
get_total_edge(x, resolution, neighbourhood = as.matrix(4))
```

Arguments

`x`	A matrix
`resolution`	A numeric vector with two values representing the input matrix resolution on the x and y axis
`neighbourhood`	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case), 8 (queen's case) or a binary matrix where the ones define the neighbourhood. The default is 4.

Value

A total edge value

`permute_raster` *Permute values in the input raster*

Description

Permute values in the input raster

Usage

```
permute_raster(x, nr_of_permutations)
```

Arguments

`x`	SpatRaster object (`terra::rast()`) containing one or more rasters
`nr_of_permutations`	Number of permutations performed on each input raster

Value

A list of matrices

Index