Package 'sapo'

October 11, 2024

Type Package

Title Spatial Association of Different Types of Polygon

Version 0.8.0

Date 2024-10-02

Description In ecology, spatial data is often represented using polygons. These polygons can represent a variety of spatial entities, such as ecological patches, animal home ranges, or gaps in the forest canopy. Researchers often need to determine if two spatial processes, represented by these polygons, are independent of each other. For instance, they might want to test if the home range of a particular animal species is influenced by the presence of a certain type of vegetation. To address this, Godoy et al. (2022) (<doi:10.1016/j.spasta.2022.100695>) developed conditional Monte Carlo tests. These tests are designed to assess spatial independence while taking into account the shape and size of the polygons.

License GPL (>= 3)

Encoding UTF-8

SystemRequirements GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)

Imports sf, methods, stats

Depends R (>= 4.0)

URL `https://github.com/lcgodoy/sapo/`

BugReports `https://github.com/lcgodoy/sapo/issues/`

RoxygenNote 7.3.2

Language en-US

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Lucas da Cunha Godoy [aut, cre]
(<`https://orcid.org/0000-0003-4265-972X`>)

Maintainer Lucas da Cunha Godoy <`lcgodoy@duck.com`>

Repository CRAN

Date/Publication 2024-10-11 08:20:02 UTC

Contents

`calc_h` *h_12(t) from matrix*

Description

Computes the h_{12} (K or L) based on a distance matrix based on a method

Usage

```
calc_h(x, var_st = FALSE, dists = NULL)
```

Arguments

`x`	distance matrix
`var_st`	logical scalar indicating if the L function should be used instead
`dists`	vector of distances to compute $h_{12}(t)$.

Value

a numeric vector

cmc_psat *Polygons Spatial Association Test - Global Envelope*

Description

A Monte Carlo test to verify if two sets of polygons are associated based in a global envelope of the functions $K_{12}(d)$ and $L_{12}(d)$ using different test statistics.

Usage

```
cmc_psat(
  p1,
  p2,
  id_col = NULL,
  n_sim = 499L,
  alpha = 0.01,
  var_st = TRUE,
  ts = "SMAD",
  distances = NULL,
  hausdorff = TRUE,
  method = "rnd_poly"
)
```

Arguments

`p1` a `sf` object containing one column specifying the objects id.

`p2` a `sf` object containing one column specifying the objects id.

`id_col` a `character` or `integer` indicating the column of `p1` storing the unique identifier for the polygons/sample units.

`n_sim` an `integer` corresponding to the number of Monte Carlo simulations for the test

`alpha` a `numeric` indicating the confidence level.

`var_st` use the variance stabilizing funciton?

`ts` a `character` associated to a test statistic. Inputs acepted: `c('IM', 'MAD', 'SIM', 'SMAD', 'IMDQ', 'MADDQ')`.

`distances` a `numeric vector` indicating the distances to evaluate $H(d)$. If `NULL` then the range considered goes from 5% to 20% of the max distance that can be observed inside the study region.

`hausdorff` a `logical` scalar indicating whether the Hausdorff distance should be used (default is TRUE).

`method` (default = "rng_poly") a `character` indicating the method used to deal with broken polygons in the Toroidal Shift. Valid options are `c("min", "max", "mean", "rnd_poly", "rnd_dist", "min_norm", "max_norm", "hybrid", "hyb_center", "hybrid_nc", "old_min")`.

Value

a `list` with values:

- **p_value** a `numeric` scalar giving the p-value of the test
- **mc_sample** a `numeric` vector giving the test statistic for each of the Monte Carlo simulations
- **mc_funct** a `matrix` where each line correspond to the function (K or L) estimated for the Monte Carlo simulations
- **distances** `numeric vector` containing the distances where mc_func were evaluated.
- **alpha** a `numeric` scalar giving the significance level
- **rejects** a `logical` scalar, TRUE if the null hypothesis is reject

Examples

```
library(sapo)
library(sf)
set.seed(2024)

 ## loading toy data
poly1 <- system.file("extdata", "poly1.rds", package = "sapo") |>
  readRDS()
poly2 <- system.file("extdata", "poly2.rds", package = "sapo") |>
  readRDS()

my_ht <- cmc_psat(poly1, poly2, n_sim = 199)
my_ht$p_value
```

`create_jump` *Create jumps for random movements*

Description

Create jumps for random movements

Usage

```
create_jump(unique_bb)
```

Arguments

`unique_bb`	a bbox shared between both "Polygon Patterns"

Details

This is an internal function.

Value

a `sfc` object representing a random jump or shift.

Author(s)

Lucas Godoy

`fix_dist` *Fix distance matrix containing broken polygons*

Description

fix a polygons' distance matrix based on a given method. This function assumes the polygon that has been broken is represented by the rows of the distance matrix.

Usage

```
fix_dist(x, method = "rnd_poly")
```

Arguments

`x`	distance matrix
`method`	method used to fix. The options are "min", "max", "mean", "rnd_poly", "rnd_dist", "min_norm", "max_norm", "hybrid", "hyb_center", "hybrid_nc", "old_min"

Value

a distance matrix

`hfun` *$h\_12(t)$ from polygons*

Description

Computes the h_{12} (K or L) based on a distance matrix based on a method

Usage

```
h_func(
  p1,
  p2,
  hausdorff = TRUE,
  method = "rnd_poly",
  var_st = FALSE,
  dists = NULL
)

h_func.list(x, ...)
```

Arguments

`p1`	sf object
`p2`	sf object
`hausdorff`	logical parameter indicating whether the Hausdorff distance should be used
`method`	method to deal with broken polygons
`var_st`	logical scalar indicating if the L function should be used instead
`dists`	vector of distances to compute $h_{12}(t)$.
`x`	a list with two sf objects.
`...`	Parameters to be used with `h_func` when inputting a list.

Value

a numeric vector

`iadist` *ID aware distance matrix*

Description

Distance between polygons accounting for toroidal shift.

Usage

```
iadist(p1, p2, hausdorff = TRUE, method = "rnd_poly")
```

Arguments

`p1`	a `sf` object containing one column specifying the objects id.
`p2`	a `sf` object containing one column specifying the objects id.
`hausdorff`	`logical` scalar indicating whether the Hausdorff distance should be used.
`method`	method for "fixing" the distance matrix.

Value

a distance matrix.

Author(s)

Lucas Godoy

im *Integram Measure*

Description

Integram Measure

Usage

```
im(x, h = 1)
```

Arguments

x	numeric matrix
h	numeric

Value

numeric vector

im_ac *Integram Measure with Assimetry Correction*

Description

Integram Measure with Assimetry Correction

Usage

```
im_ac(x, h = 1)
```

Arguments

x	numeric matrix
h	numeric

Value

numeric vector

mad *Maximum Absolute Deviation*

Description

Maximum Absolute Deviation

Usage

```
mad(x)
```

Arguments

x	numeric matrix

Value

numeric vector

mad_ac *Maximum Absolute Deviation with Assimetry Correction*

Description

Maximum Absolute Deviation with Assimetry Correction

Usage

```
mad_ac(x)
```

Arguments

x	numeric matrix

Value

numeric vector

`mean_aux` *auxiliary mean*

Description

aux function to calculate the mean of a vector when removing each of its elements one by one.

Usage

```
mean_aux(x)
```

Arguments

`x`	a numeric vector

Value

a numeric vector

Author(s)

Lucas Godoy

`pre_ts` *Pre-TS*

Description

Create rigid copies of a polygon. This function an auxilliary function for the Toroidal Shift method

Usage

```
pre_ts(poly, bb = NULL, id_col = NULL)
```

Arguments

`poly`	an object of class `sf` or `sfc`.
`bb`	(optional) a unique bounding box.
`id_col`	a `character` indicating the id column in `poly`.

Value

an `sf` with 8 additional rigid copies of `poly`.

Author(s)

Lucas Godoy

sapo *sapo: Spatial Association of Polygon Types*

Description

sapo: Spatial Association of Polygon Types

s_im *Studentized Integram Measure*

Description

Studentized Integram Measure

Usage

```
s_im(x, h = 1)
```

Arguments

x	numeric matrix
h	numeric

Value

numeric vector

s_mad *Studentized Maximum Absolute Deviation*

Description

Studentized Maximum Absolute Deviation

Usage

```
s_mad(x)
```

Arguments

x	numeric matrix

Value

numeric vector

`toroidal_shift` *Toroidal Shift*

Description

Toroidal Shift

Usage

```
toroidal_shift(x, y, shifted = FALSE, unique_bb = NULL)
```

Arguments

`x`	a `sf` or `sfc` object. Its `geometry` may contain POLYGONS and/or POINTS.
`y`	a `sf` or `sfc` object. Its `geometry` may contain POLYGONS and/or POINTS.
`shifted`	`logical` indicating whether `x` has been "shifted". This parameter is mainly for internal use and testing.
`unique_bb`	a bbox shared between both "Polygon Patterns"

Value

a list

Author(s)

Lucas Godoy

`translate_by_pt` *Translate an `sf` object by a "point"*

Description

Translate an `sf` object by a "point"

Usage

```
translate_by_pt(pt, poly)
```

Arguments

`pt`	`sfc` representing a shift.
`poly`	`sfc` of `sf` to be shifted

Value

a `sf` or `sfc` representing `poly` shifted by `pt`

Author(s)

Lucas Godoy

Index