Package 'motif'

August 23, 2021

Title Local Pattern Analysis

Version 0.5.0

Description Describes spatial patterns of categorical raster data for any defined regular and irregular areas.
Patterns are described quantitatively using built-in signatures based on co-occurrence matrices but also allows for any user-defined functions.
It enables spatial analysis such as search, change detection, and clustering to be performed on spatial patterns (Nowosad (2021) <doi:10.1007/s10980-020-01135-0>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Depends R (>= 3.1)

LinkingTo comat (>= 0.7.0), Rcpp, RcppArmadillo

Imports comat, philentropy, Rcpp, sf, stars, tibble

Suggests covr, dplyr, spdep, knitr, rmarkdown, testthat (>= 2.1.0), terra

URL https://nowosad.github.io/motif/

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

VignetteBuilder knitr

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2021-08-23 12:50:02 UTC

R topics documented:

`determine_classes` *Determine unique classes (internal function)*

Description

Determine unique classes (internal function)

Usage

```
determine_classes(x, window)
```

Arguments

`x`	• a `stars` or `stars_proxy` object
`window`	• a windows argument from `lsp_signature()`, `lsp_search()`, or `lsp_compare()`

Value

A list with vector of numbers (unique classes)

lsp_add_clusters *Adds clusters' ids to a lsp object*

Description

Adds clusters' ids to a lsp object. The output can be of `stars`, `sf`, or `terra` class. See examples.

Usage

```
lsp_add_clusters(x, clust, output = "sf", window = NULL)
```

Arguments

`x`	Object of class `lsp` - usually the output of the `lsp_signature()` function
`clust`	Vector containing an id value for each row in `x`
`output`	The class of the output. Either `stars`, `sf`, or `terra`
`window`	Specifies areas for analysis. It can be either: `NULL` or an `sf` object. If `window=NULL` calculations are performed based on the metadata from `x`. If an `sf` object is provided, each feature (row) defines the extent of a local pattern. The `sf` object should have one attribute (otherwise, the first attribute is used as an id).

Value

Object of class `stars`, `sf`, or `terra` (depending on the `output` argument) with an additional column `"clust"` representing clusters' id values.

Examples

```
library(stars)
library(sf)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
landform_cove = lsp_signature(landform,
                              type = "cove",
                              window = 200,
                              normalization = "pdf")

landform_dist = lsp_to_dist(landform_cove,
                            dist_fun = "jensen-shannon")

landform_hclust = hclust(landform_dist, method = "ward.D2")
plot(landform_hclust)

clusters = cutree(landform_hclust, k = 4)

landform_grid_sf = lsp_add_clusters(landform_cove, clusters)
plot(landform_grid_sf["clust"])
```

```
landform_grid_sfq = lsp_add_quality(landform_grid_sf,
                                    landform_dist)
plot(landform_grid_sfq["quality"])


# larger data example
library(stars)
library(sf)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
landform_cove = lsp_signature(landform,
                              type = "cove",
                              window = 200,
                              normalization = "pdf")

landform_dist = lsp_to_dist(landform_cove,
                            dist_fun = "jensen-shannon")

landform_hclust = hclust(landform_dist, method = "ward.D2")
plot(landform_hclust)

clusters = cutree(landform_hclust, k = 6)

landform_grid_sf = lsp_add_clusters(landform_cove, clusters)
plot(landform_grid_sf["clust"])

landform_grid_sfq = lsp_add_quality(landform_grid_sf,
                                    landform_dist)
plot(landform_grid_sfq["quality"])
```

`lsp_add_examples` *Adds spatial data of each region in an lsp or sf object*

Description

Adds spatial data of each region in an lsp or sf object. The output is an lsp or sf object with an additional column `"region"`. See examples.

Usage

```
lsp_add_examples(x, y, window = NULL)

## S3 method for class 'lsp'
lsp_add_examples(x, y, window = NULL)

## S3 method for class 'sf'
lsp_add_examples(x, y, window = NULL)
```

Arguments

x	Object of class lsp - usually a subset of the output of lsp_signature() or an object of class sf - usually a subset of the output of lsp_search()
y	Object of class stars, stars_proxy, or terra's SpatRaster.
window	Specifies areas for analysis. It can be either: NULL or an sf object. The sf object is only needed for adding examples of irregular regions.

Value

The input object with a new column "region". The "region" column is a list with a raster extracted for each row.

Examples

```
library(stars)

landcover = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))

landcover_coma = lsp_signature(landcover, type = "coma", threshold = 0.9, window = 100)
selected_coma = subset(landcover_coma, id %in% c(5, 10, 15, 35))
selected_coma

selected_coma = lsp_add_examples(x = selected_coma, y = landcover)
selected_coma

plot(selected_coma$region[[1]])
plot(selected_coma$region[[4]])


# larger data example
library(stars)

landcover = read_stars(system.file("raster/landcover2015.tif", package = "motif"))

landcover_coma = lsp_signature(landcover, type = "coma", threshold = 0.9, window = 100)
selected_coma = subset(landcover_coma, id %in% c(5, 80, 1971, 2048))
selected_coma

selected_coma = lsp_add_examples(x = selected_coma, y = landcover)
selected_coma

plot(selected_coma$region[[1]])
plot(selected_coma$region[[4]])
```

`lsp_add_quality` *Calculates quality metrics of clustering or segmentation*

Description

Calculates three metrics to evaluate quality of spatial patterns' clustering or segmentation. When the type is `"cluster"`, then metrics of inhomogeneity, distinction, and quality are calculated. When the type is "segmentation, then metrics of inhomogeneity, isolation, and quality are calculated. For more information, see Details below.

Usage

```
lsp_add_quality(x, x_dist, type = "cluster", regions = FALSE)
```

Arguments

`x`	Object of class `sf` - usually the output of the `lsp_add_clusters()` function
`x_dist`	Object of class `dist` - usually the output of the `lsp_to_dist()` function
`type`	Either `"cluster"` or `"segmentation"`
`regions`	Not implemented yet

Details

For type `"cluster"`, this function calculates three quality metrics to evaluate spatial patterns' clustering: (1) inhomogeneity - it measures a degree of mutual dissimilarity between all objects in a cluster. This value is between 0 and 1, where small value indicates that all objects in the cluster represent consistent patterns so the cluster is pattern-homogeneous. (2) distinction - it is an average distance between the focus cluster and all of the other clusters. This value is between 0 and 1, where large value indicates that the cluster stands out from the other clusters. (3) quality - overall quality of a cluster. It is calculated as 1 - (inhomogeneity / distinction). This value is also between 0 and 1, where increased values indicate increased quality.

For type `"segmentation"`, this function calculates three quality metrics to evaluate spatial patterns' segmentation: (1) inhomogeneity - it measures a degree of mutual dissimilarity between all objects in a cluster. This value is between 0 and 1, where small value indicates that all objects in the cluster represent consistent patterns so the cluster is pattern-homogeneous. (2) isolation - it is an average distance between the focus cluster and all of its neighbors. This value is between 0 and 1, where large value indicates that the cluster stands out from its surroundings. (3) quality - overall quality of a cluster. It is calculated as 1 - (inhomogeneity / distinction). This value is also between 0 and 1, where increased values indicate increased quality.

Value

Object of class `sf` with three additional columns representing quality metrics.

References

Jakub Nowosad & Tomasz F. Stepinski (2021) Pattern-based identification and mapping of landscape types using multi-thematic data, International Journal of Geographical Information Science, DOI: 10.1080/13658816.2021.1893324

See Also

lsp_add_clusters

Examples

```
# see examples of `lsp_add_clusters()`
```

`lsp_add_sf` *Creates or adds a sf object*

Description

Creates or adds a sf object based on the input object or a set of parameters. It accepts either an object of class `stars` or `lsp`. In the first case, the output is created based on a set of parameters (`window_size` and `window_shift` or `window`). In the second case, the output converts the `lsp` object into a `sf` object.

Usage

```
lsp_add_sf(x = NULL, window = NULL)

## Default S3 method:
lsp_add_sf(x = NULL, window = NULL)

## S3 method for class 'lsp'
lsp_add_sf(x = NULL, window = NULL)
```

Arguments

`x`	Object of class `stars` or `lsp`. For `stars`, `window` or `window_size` can be used.
`window`	Specifies areas for analysis. It can be either: `NULL`, a numeric value, or an `sf` object. If `window=NULL` calculations are performed for a whole area. If the `window` argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an `sf` object is provided, each feature (row) defines the extent of a local pattern. The `sf` object should have one attribute (otherwise, the first attribute is used as an id).

Value

An `sf` object converted from the input object or a provided set of parameters

Examples

```
library(stars)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
plot(landform)
landform_lsp = lsp_add_sf(landform, window = 100)
plot(landform_lsp)

lc_cove = lsp_signature(landform, type = "cove", window = 200, normalization = "pdf")
lc_cove_lsp = lsp_add_sf(lc_cove)
plot(lc_cove_lsp["id"])
plot(lc_cove_lsp["na_prop"])


# larger data example
library(stars)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
plot(landform)
landform_lsp = lsp_add_sf(landform, window = 100)
plot(landform_lsp)

lc_cove = lsp_signature(landform, type = "cove", window = 200, normalization = "pdf")
lc_cove_lsp = lsp_add_sf(lc_cove)
plot(lc_cove_lsp["id"])
plot(lc_cove_lsp["na_prop"])
```

`lsp_add_stars` *Creates or adds a stars object*

Description

Creates or adds a stars object based on the input object or a set of parameters. It accepts either an object of class `stars` or `lsp`. In the first case, the output is created based on the `window` parameter. In the second case, the output converts the `lsp` object into a `stars` object.

Usage

```
lsp_add_stars(x = NULL, window = NULL)

## Default S3 method:
lsp_add_stars(x = NULL, window = NULL)

## S3 method for class 'lsp'
lsp_add_stars(x = NULL, window = NULL)
```

Arguments

`x`	Object of class `stars` or `lsp`. For `stars`, `window` or `window_size` can be used.

window — Specifies areas for analysis. It can be either: NULL, a numeric value, or an sf object. If window=NULL calculations are performed for a whole area. If the window argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an sf object is provided, each feature (row) defines the extent of a local pattern. The sf object should have one attribute (otherwise, the first attribute is used as an id).

Value

A stars object converted from the input object or a provided set of parameters

Examples

```
library(stars)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
plot(landform)
landform_lsp = lsp_add_stars(landform, window = 100)
plot(landform_lsp)

lc_cove = lsp_signature(landform, type = "cove", window = 200, normalization = "pdf")
lc_cove_lsp = lsp_add_stars(lc_cove)
plot(lc_cove_lsp)
plot(lc_cove_lsp["na_prop"])



# larger data example
library(stars)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
plot(landform)
landform_lsp = lsp_add_stars(landform, window = 100)
plot(landform_lsp)

lc_cove = lsp_signature(landform, type = "cove", window = 200, normalization = "pdf")
lc_cove_lsp = lsp_add_stars(lc_cove)
plot(lc_cove_lsp)
plot(lc_cove_lsp["na_prop"])
```

---

lsp_add_terra *Creates or adds a terra object*

Description

Creates or adds a terra object based on the input object or a set of parameters. It accepts either an object of class stars or lsp. In the first case, the output is created based on the window parameter. In the second case, the output converts the lsp object into a terra object.

Usage

```
lsp_add_terra(x = NULL, window = NULL)
```

Arguments

x	Object of class stars or lsp. For stars, window or window_size can be used.
window	Specifies areas for analysis. It can be either: NULL, a numeric value, or an sf object. If window=NULL calculations are performed for a whole area. If the window argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an sf object is provided, each feature (row) defines the extent of a local pattern. The sf object should have one attribute (otherwise, the first attribute is used as an id).

Value

A terra object converted from the input object or a provided set of parameters

Examples

```
library(stars)
library(terra)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
plot(landform)
landform_lsp = lsp_add_terra(landform, window = 100)
plot(landform_lsp)

lc_cove = lsp_signature(landform, type = "cove", window = 200, normalization = "pdf")
lc_cove_lsp = lsp_add_terra(lc_cove)
plot(lc_cove_lsp)
plot(lc_cove_lsp["na_prop"])
```

---

lsp_compare *Comparison between spatial patterns*

Description

Compares two spatial datasets containing categorical raster data. It accepts a categorical raster dataset with one or more attributes, and compares it to the second dataset with the same attributes and dimensions. The both dataset are either compared to as whole areas, areas divided into regular windows, or areas divided into irregular windows. This function allows for several types of comparisons using different representations of spatial patterns, including "coma" (co-occurrence matrix), "cove" (co-occurrence vector), "cocoma" (co-located co-occurrence matrix), "cocove" (co-located co-occurrence vector), "wecoma" (weighted co-occurrence matrix), "wecove" (weighted co-occurrence vector), "incoma" (integrated co-occurrence matrix), "incove" (integrated co-occurrence vector). These representations are created for both datasets, and next a distance between them is calculated using a selected measure from the philentropy::distance function. Additional parameters, such as neighbourhood or normalization types, are also available.

Usage

```
lsp_compare(
  x,
  y,
  type,
  dist_fun,
  window = NULL,
  output = "stars",
  neighbourhood = 4,
  threshold = 0.5,
  ordered = TRUE,
  repeated = TRUE,
  normalization = "pdf",
  wecoma_fun = "mean",
  wecoma_na_action = "replace",
  ...
)

## S3 method for class 'stars'
lsp_compare(
  x,
  y,
  type,
  dist_fun,
  window = NULL,
  output = "stars",
  neighbourhood = 4,
  threshold = 0.5,
  ordered = TRUE,
  repeated = TRUE,
  normalization = "pdf",
  wecoma_fun = "mean",
  wecoma_na_action = "replace",
  ...
)
```

Arguments

x	Object of class stars, stars_proxy, or terra's SpatRaster. It should have one attribute (for "coma", "cove"), two attributes ("cocoma", "cocove", "wecoma", "wecove"), two or more attributes ("incoma", "incove"), or any number of attributes suitable for user-defined functions.
y	Object of class stars, stars_proxy, or terra's SpatRaster. It should have one attribute (for "coma", "cove"), two attributes ("cocoma", "cocove", "wecoma", "wecove"), two or more attributes ("incoma", "incove"), or any number of attributes suitable for user-defined functions.
type	Type of the calculated signature. It can be "coma" (co-occurrence matrix), "cove" (co-occurrence vector), "cocoma" (co-located co-occurrence matrix),

	"cocove" (co-located co-occurrence vector), "wecoma" (weighted co-occurrence matrix), "wecove" (weighted co-occurrence vector), "incoma" (integrated co-occurrence matrix), "incove" (integrated co-occurrence vector), "composition" or any function that can summarize stars objects.
dist_fun	Distance measure used. This function uses the philentropy::distance function in the background. Run philentropy::getDistMethods() to find possible distance measures.
window	Specifies areas for analysis. It can be either: NULL, a numeric value, or an sf object. If window=NULL calculations are performed for a whole area. If the window argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an sf object is provided, each feature (row) defines the extent of a local pattern. The sf object should have one attribute (otherwise, the first attribute is used as an id).
output	The class of the output. Either "stars", "sf", or terra
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.
threshold	The share of NA cells to allow metrics calculation.
ordered	For "cove", "cocove", "wecove" and "incove" only. The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
repeated	For "incove" only. Should the repeated co-located co-occurrence matrices be used? Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
normalization	For "cove", "cocove", "wecove", "incove", "composition", or user-provided functions only. Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "pdf".
wecoma_fun	For "wecoma" and "wecove" only. Function to calculate values from adjacent cells to contribute to exposure matrix, "mean" - calculate average values of local population densities from adjacent cells, "geometric_mean" - calculate geometric mean values of local population densities from adjacent cells, or "focal" assign a value from the focal cell
wecoma_na_action	For "wecoma" and "wecove" only. Decides on how to behave in the presence of missing values in w. Possible options are "replace", "omit", "keep". The default, "replace", replaces missing values with 0, "omit" does not use cells with missing values, and "keep" keeps missing values.
...	Additional arguments for the philentropy::distance function.

Value

Object of class stars (or sf or terra's SpatRaster, depending on the output argument). It has four attributes: (1) id - an id of each window. For irregular windows, it is the values provided in the window argument, (2) na_prop_x - share (0-1) of NA cells for each window in the x object, (3) na_prop_y - share (0-1) of NA cells for each window in the y object, (4) dist- calculated distance between signatures for each window

Examples

```
library(stars)

lc15 = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))
lc01 = read_stars(system.file("raster/landcover2001s.tif", package = "motif"))
ecoregions = read_sf(system.file("vector/ecoregionss.gpkg", package = "motif"))

ecoregions = st_transform(ecoregions, st_crs(lc15))

c1 = lsp_compare(lc01, lc15, type = "cove",
    dist_fun = "jensen-shannon", window = ecoregions["id"])
plot(c1["dist"])


# larger data example
library(stars)

lc15 = read_stars(system.file("raster/landcover2015.tif", package = "motif"))
lc01 = read_stars(system.file("raster/landcover2001.tif", package = "motif"))
ecoregions = read_sf(system.file("vector/ecoregions.gpkg", package = "motif"))

ecoregions = st_transform(ecoregions, st_crs(lc15))

c1 = lsp_compare(lc01, lc15, type = "cove",
    dist_fun = "jensen-shannon", window = ecoregions["id"])
plot(c1["dist"])
```

---

lsp_extract *Extracts a local landscape*

Description

Extracts a local landscape from categorical raster data based on its id and provided `window` argument.

Usage

```
lsp_extract(x, window, id, output = "stars")
```

Arguments

`x`	Object of class `stars`, `stars_proxy`, or terra's `SpatRaster`.
`window`	Specifies areas for analysis. It can be either: `NULL`, a numeric value, or an `sf` object.
`id`	Id of the local landscape - it is possible to find in the output of `lsp_signature()`, `lsp_search()`, `lsp_compare()`, or `lsp_add_clusters()`.
`output`	The class of the output. Either `"stars"` or `terra`

Value

A starsor terra object cropped to the extent of a selected local landscape

Examples

```
library(stars)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
ecoregions = read_sf(system.file("vector/ecoregionss.gpkg", package = "motif"))

extract1 = lsp_extract(x = landform, window = 100, id = 25)
plot(extract1)

ecoregions = st_transform(ecoregions, st_crs(landform))
extract2 = lsp_extract(x = landform, window = ecoregions["id"], id = 11)
plot(extract2)


# larger data example
library(stars)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
ecoregions = read_sf(system.file("vector/ecoregions.gpkg", package = "motif"))

extract1 = lsp_extract(x = landform, window = 100, id = 1895)
plot(extract1)

ecoregions = st_transform(ecoregions, st_crs(landform))
extract2 = lsp_extract(x = landform, window = ecoregions["id"], id = 7)
plot(extract2)
```

---

lsp_mosaic *Creates a raster mosaic*

Description

Creates a raster mosaic by rearranging spatial data for example regions. See examples.

Usage

```
lsp_mosaic(x, output = "stars")
```

Arguments

x	Usually the output of the lsp_add_examples() function
output	The class of the output. Either "stars" or terra

Value

A stars or terra object

Examples

```
# larger data example
library(stars)
library(sf)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
landform_cove = lsp_signature(landform,
                              type = "cove",
                              window = 200,
                              normalization = "pdf")

landform_dist = lsp_to_dist(landform_cove,
                            dist_fun = "jensen-shannon")

landform_hclust = hclust(landform_dist, method = "ward.D2")
plot(landform_hclust)

clusters = cutree(landform_hclust, k = 6)

landform_grid_sf = lsp_add_clusters(landform_cove, clusters)
plot(landform_grid_sf["clust"])

landform_grid_sf_sel = landform_grid_sf %>%
    dplyr::filter(na_prop == 0) %>%
    dplyr::group_by(clust) %>%
    dplyr::slice_sample(n = 16, replace = TRUE)

landform_grid_sf_sel = lsp_add_examples(x = landform_grid_sf_sel, y = landform)
landform_grid_sf_sel

landform_clust_m = lsp_mosaic(landform_grid_sf_sel)

plot(landform_clust_m)
```

`lsp_restructure` *Changes structure of the lsp object*

Description

Converts a list-column with signatures into many numeric columns

Usage

```
lsp_restructure(x)
```

Arguments

x	• an lsp object

Value

Object of class `lsp`. It has several columns: (1) `id` - an id of each window. For irregular windows, it is the values provided in the `window` argument, (2) `na_prop` - share (0-1) of `NA` cells for each window, (3) one or more columns representing values of the signature

Examples

```
library(stars)

landcover = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))

landcover_cove = lsp_signature(landcover, type = "cove", threshold = 0.9, window = 100)
landcover_cover = lsp_restructure(landcover_cove)
landcover_cover

lsp_add_sf(landcover_cover)
```

---

lsp_search *Search for similar spatial pattern*

Description

Searches for areas with similar spatial patterns in categorical data. It accepts a categorical raster dataset with one or more attributes, and compares it to the second (usually larger) dataset with the same attributes. The first dataset is either compared to a whole area, areas divided into regular windows, or areas divided into irregular windows from the second dataset. This function allows for several types of comparisons using different representations of spatial patterns, including "coma" (co-occurrence matrix), "cove" (co-occurrence vector), "cocoma" (co-located co-occurrence matrix), "cocove" (co-located co-occurrence vector), "wecoma" (weighted co-occurrence matrix), "wecove" (weighted co-occurrence vector), "incoma" (integrated co-occurrence matrix), "incove" (integrated co-occurrence vector). These representations are created for both datasets, and next a distance between them is calculated using a selected measure from the `philentropy::distance` function. Additional parameters, such as neighbourhood or normalization types, are also available.

Usage

```
lsp_search(
  x,
  y,
  type,
  dist_fun,
  window = NULL,
  output = "stars",
  neighbourhood = 4,
  threshold = 0.5,
  ordered = TRUE,
```

```
  repeated = TRUE,
  normalization = "pdf",
  wecoma_fun = "mean",
  wecoma_na_action = "replace",
  classes = NULL,
  ...
)

## S3 method for class 'stars'
lsp_search(
  x,
  y,
  type,
  dist_fun,
  window = NULL,
  output = "stars",
  neighbourhood = 4,
  threshold = 0.5,
  ordered = TRUE,
  repeated = TRUE,
  normalization = "pdf",
  wecoma_fun = "mean",
  wecoma_na_action = "replace",
  classes = NULL,
  ...
)
```

Arguments

x	Object of class `stars`, `stars_proxy`, or terra's `SpatRaster`. It should have one attribute (for `"coma"`, `"cove"`), two attributes (`"cocoma"`, `"cocove"`, `"wecoma"`, `"wecove"`), two or more attributes (`"incoma"`, `"incove"`), or any number of attributes suitable for user-defined functions.
y	Object of class `stars`, `stars_proxy`, or terra's `SpatRaster`. It should have one attribute (for `"coma"`, `"cove"`), two attributes (`"cocoma"`, `"cocove"`, `"wecoma"`, `"wecove"`), two or more attributes (`"incoma"`, `"incove"`), or any number of attributes suitable for user-defined functions.
type	Type of the calculated signature. It can be `"coma"` (co-occurrence matrix), `"cove"` (co-occurrence vector), `"cocoma"` (co-located co-occurrence matrix), `"cocove"` (co-located co-occurrence vector), `"wecoma"` (weighted co-occurrence matrix), `"wecove"` (weighted co-occurrence vector), `"incoma"` (integrated co-occurrence matrix), `"incove"` (integrated co-occurrence vector), `"composition"` or any function that can summarize `stars` objects.
dist_fun	Distance measure used. This function uses the `philentropy::distance` function in the background. Run `philentropy::getDistMethods()` to find possible distance measures.
window	Specifies areas for analysis. It can be either: `NULL`, a numeric value, or an `sf` object. If `window=NULL` calculations are performed for a whole area. If the

	window argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an sf object is provided, each feature (row) defines the extent of a local pattern. The sf object should have one attribute (otherwise, the first attribute is used as an id).
output	The class of the output. Either "stars", "sf", or terra
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.
threshold	The share of NA cells to allow metrics calculation.
ordered	For "cove", "cocove", "wecove" and "incove" only. The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
repeated	For "incove" only. Should the repeated co-located co-occurrence matrices be used? Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
normalization	For "cove", "cocove", "wecove", "incove", "composition", or user-provided functions only. Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "pdf".
wecoma_fun	For "wecoma" and "wecove" only. Function to calculate values from adjacent cells to contribute to exposure matrix, "mean" - calculate average values of local population densities from adjacent cells, "geometric_mean" - calculate geometric mean values of local population densities from adjacent cells, or "focal" assign a value from the focal cell
wecoma_na_action	For "wecoma" and "wecove" only. Decides on how to behave in the presence of missing values in w. Possible options are "replace", "omit", "keep". The default, "replace", replaces missing values with 0, "omit" does not use cells with missing values, and "keep" keeps missing values.
classes	Which classes (categories) should be analyzed? This parameter expects a list of the same length as the number of attributes in x, where each element of the list contains integer vector. The default is NULL, which means that the classes are calculated directly from the input data and all of them are used in the calculations.
...	Additional arguments for the philentropy::distance function.

Value

Object of class stars (or sf or terra's SpatRaster, depending on the output argument). It has three attributes: (1) id - an id of each window. For irregular windows, it is the values provided in the window argument, (2) na_prop - share (0-1) of NA cells for each window in the y object, (3) dist- calculated distance between the x object and each window in the y object

Examples

```
library(stars)

landcover = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))
plot(landcover)
```

```
ext = st_bbox(c(xmin = -249797.344531127, xmax = -211162.693944285,
                ymin = -597280.143035389, ymax = -558645.492448547),
                crs = st_crs(landcover))

landcover_ext = landcover[ext]
plot(landcover_ext)

ecoregions = read_sf(system.file("vector/ecoregionss.gpkg", package = "motif"))
plot(ecoregions["id"])

s1 = lsp_search(landcover_ext, landcover, type = "cove",
  dist_fun = "jensen-shannon", threshold = 0.9, window = 100)
plot(s1["dist"])

ecoregions = st_transform(ecoregions, st_crs(landcover))
s2 = lsp_search(landcover_ext, landcover, type = "cove",
  dist_fun = "jensen-shannon", threshold = 0.5, window = ecoregions["id"])
plot(s2["dist"])


# larger data example
library(stars)

landcover = read_stars(system.file("raster/landcover2015.tif", package = "motif"))
plot(landcover)

ext = st_bbox(c(xmin = -249797.344531127, xmax = -211162.693944285,
                ymin = -597280.143035389, ymax = -558645.492448547),
                crs = st_crs(landcover))

landcover_ext = landcover[ext]
plot(landcover_ext)

ecoregions = read_sf(system.file("vector/ecoregions.gpkg", package = "motif"))
plot(ecoregions["id"])

s1 = lsp_search(landcover_ext, landcover, type = "cove",
  dist_fun = "jensen-shannon", threshold = 0.9, window = 1000)
plot(s1["dist"])

ecoregions = st_transform(ecoregions, st_crs(landcover))
s2 = lsp_search(landcover_ext, landcover, type = "cove",
  dist_fun = "jensen-shannon", threshold = 0.5, window = ecoregions["id"])
plot(s2["dist"])
```

`lsp_signature` *Creates a spatial signature*

Description

Calculates selected spatial signatures based on categorical raster data. It also allows for calculations for any defined regular and irregular areas. It has several built-in signatures but also allows for any user-defined functions.

Usage

```
lsp_signature(
  x,
  type,
  window = NULL,
  neighbourhood = 4,
  threshold = 0.9,
  ordered = TRUE,
  repeated = TRUE,
  normalization = "pdf",
  wecoma_fun = "mean",
  wecoma_na_action = "replace",
  classes = NULL
)
```

Arguments

x	Object of class stars, stars_proxy, or terra's SpatRaster. It should have one attribute (for "coma", "cove"), two attributes ("cocoma", "cocove", "wecoma", "wecove"), two or more attributes ("incoma", "incove"), or any number of attributes suitable for user-defined functions.
type	Type of the calculated signature. It can be "coma" (co-occurrence matrix), "cove" (co-occurrence vector), "cocoma" (co-located co-occurrence matrix), "cocove" (co-located co-occurrence vector), "wecoma" (weighted co-occurrence matrix), "wecove" (weighted co-occurrence vector), "incoma" (integrated co-occurrence matrix), "incove" (integrated co-occurrence vector), "composition" or any function that can summarize stars objects.
window	Specifies areas for analysis. It can be either: NULL, a numeric value, or an sf object. If window=NULL calculations are performed for a whole area. If the window argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an sf object is provided, each feature (row) defines the extent of a local pattern. The sf object should have one attribute (otherwise, the first attribute is used as an id).
neighbourhood	The number of directions in which cell adjacencies are considered as neighbours: 4 (rook's case) or 8 (queen's case). The default is 4.
threshold	The share of NA cells (0-1) to allow metrics calculation.
ordered	For "cove", "cocove", "wecove" and "incove" only. The type of pairs considered. Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.
repeated	For "incove" only. Should the repeated co-located co-occurrence matrices be used? Either "ordered" (TRUE) or "unordered" (FALSE). The default is TRUE.

normalization For "cove", "cocove", "wecove", "incove", "composition", or user-provided functions only. Should the output vector be normalized? Either "none" or "pdf". The "pdf" option normalizes a vector to sum to one. The default is "pdf".

wecoma_fun For "wecoma" and "wecove" only. Function to calculate values from adjacent cells to contribute to exposure matrix, "mean" - calculate average values of local population densities from adjacent cells, "geometric_mean" - calculate geometric mean values of local population densities from adjacent cells, or "focal" assign a value from the focal cell

wecoma_na_action For "wecoma" and "wecove" only. Decides on how to behave in the presence of missing values in w. Possible options are "replace", "omit", "keep". The default, "replace", replaces missing values with 0, "omit" does not use cells with missing values, and "keep" keeps missing values.

classes Which classes (categories) should be analyzed? This parameter expects a list of the same length as the number of attributes in x, where each element of the list contains integer vector. The default is NULL, which means that the classes are calculated directly from the input data and all of them are used in the calculations.

Value

Object of class lsp. It has three columns: (1) id - an id of each window. For irregular windows, it is the values provided in the window argument, (2) na_prop - share (0-1) of NA cells for each window, (3) signature - a list-column containing calculated signatures

Examples

```
library(stars)

landcover = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))

landcover_coma = lsp_signature(landcover, type = "coma", threshold = 0.9, window = 2000)
landcover_coma

landcover_comp = lsp_signature(landcover, type = "composition", threshold = 0.9)
landcover_comp


# larger data example
library(stars)

landcover = read_stars(system.file("raster/landcover2015.tif", package = "motif"))

landcover_coma = lsp_signature(landcover, type = "coma", threshold = 0.9, window = 2000)
landcover_coma

landcover_comp = lsp_signature(landcover, type = "composition", threshold = 0.9)
landcover_comp
```

`lsp_to_dist` *Calculate Distance Matrix*

Description

Calculates a distance matrix based on an object of class `lsp`.

Usage

```
lsp_to_dist(x, dist_fun, unit = "log2", p = NULL)
```

Arguments

`x`	An object of class `lsp` - usually the output of the `lsp_signature()` function
`dist_fun`	A distance/dissimilarity method used. All possible values can be found using the `philentropy::getDistMethods()` function
`unit`	A character string specifying the logarithm unit that should be used to compute distances that depend on log computations: "log", "log2", "log10". The default is `"log"`
`p`	Power of the Minkowski distance. Used only when the `dist_fun = "minkowski"`

Value

An object of class ‘"dist"‘

Examples

```
library(stars)
landcover = read_stars(system.file("raster/landcover2015s.tif", package = "motif"))

landcover_cove = lsp_signature(landcover, type = "cove", threshold = 0.9, window = 400)
landcover_cove

dist_cov = lsp_to_dist(landcover_cove, dist_fun = "jensen-shannon")
dist_cov


# larger data example
library(stars)
landcover = read_stars(system.file("raster/landcover2015.tif", package = "motif"))

landcover_cove = lsp_signature(landcover, type = "cove", threshold = 0.9, window = 2000)
landcover_cove

dist_cov = lsp_to_dist(landcover_cove, dist_fun = "jensen-shannon")
dist_cov
```

`lsp_transform` *Transforms lsp objects*

Description

It allows for transforming spatial signatures (outputs of the `lsp_signature()` function) using user-provided functions. See examples for more details.

Usage

```
lsp_transform(x, fun, ...)
```

Arguments

`x`	Object of class `lsp` - usually the output of the `lsp_signature()` function.
`fun`	A user-provided function.
`...`	Additional arguments for `fun`.

Value

Object of class `lsp`. It has three columns: (1) `id` - an id of each window. For irregular windows, it is the values provided in the `window` argument, (2) `na_prop` - share (0-1) of `NA` cells for each window, (3) `signature` - a list-column containing with calculated signatures

Examples

```
library(stars)
landform = read_stars(system.file("raster/landforms.tif", package = "motif"))
result_coma500 = lsp_signature(landform, type = "coma", threshold = 0.5, window = 500)

#see how the first signature looks
result_coma500$signature[[1]]

my_function = function(mat){
    mat_c = colSums(mat)
    freqs = mat_c / sum(mat)
    # entropy
    -sum(freqs * log2(freqs), na.rm = TRUE)
}

result_coma500_2 = lsp_transform(result_coma500, my_function)

#see how the first signature looks after transformation
result_coma500_2$signature[[1]]


# larger data example
library(stars)
landform = read_stars(system.file("raster/landform.tif", package = "motif"))
```

```
result_coma500 = lsp_signature(landform, type = "coma", threshold = 0.5, window = 500)

#see how the first signature looks
result_coma500$signature[[1]]

my_function = function(mat){
    mat_c = colSums(mat)
    freqs = mat_c / sum(mat)
    # entropy
    -sum(freqs * log2(freqs), na.rm = TRUE)
}

result_coma500_2 = lsp_transform(result_coma500, my_function)

#see how the first signature looks after transformation
result_coma500_2$signature[[1]]
```

`prepare_window` *Prepares window* arguments (internal function)*

Description

Prepares window* arguments (internal function)

Usage

```
prepare_window(x, window)
```

Arguments

`x`	Object of class `stars` or `stars_proxy`
`window`	Specifies areas for analysis. It can be either: `NULL`, a numeric value, or an `sf` object. If `window=NULL` calculations are performed for a whole area. If the `window` argument is numeric, it is a length of the side of a square-shaped block of cells. Expressed in the numbers of cells, it defines the extent of a local pattern. If an `sf` object is provided, each feature (row) defines the extent of a local pattern. The `sf` object should have one attribute (otherwise, the first attribute is used as an id).

Value

A list with `window`, `window_size`, and `window_shift`

Index