Package 'sdsfun'

August 30, 2024

Title Spatial Data Science Complementary Features

Version 0.1.0

Description Wrapping and supplementing commonly used functions in the R ecosystem
related to spatial data science, while serving as a basis for other packages
maintained by Wenbo Lv.

License GPL-3

Encoding UTF-8

URL https://spatlyu.github.io/sdsfun/,
https://github.com/SpatLyu/sdsfun

BugReports https://github.com/SpatLyu/sdsfun/issues

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports dplyr, geosphere, magrittr, sf, spdep, stats, tibble

Suggests ggplot2, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Wenbo Lv [aut, cre, cph] (<https://orcid.org/0009-0002-6003-3800>)

Maintainer Wenbo Lv <lyu.geosocial@gmail.com>

Repository CRAN

Date/Publication 2024-08-30 10:50:02 UTC

Contents

---

inverse_distance_swm   *construct inverse distance weight*

Description

Function for constructing inverse distance weight.

Usage

```
inverse_distance_swm(sfj, power = 1, bandwidth = NULL)
```

Arguments

sfj	Vector object that can be converted to sf by sf::st_as_sf().
power	(optional) Default is 1. Set to 2 for gravity weights.
bandwidth	(optional) When the distance is bigger than bandwidth, the corresponding part of the weight matrix is set to 0. Default is NULL, which means not use the bandwidth.

Details

The inverse distance weight formula is $w_{ij} = 1/d_{ij}^{\alpha}$

Value

A inverse distance weight matrices with class of matrix.

Examples

```
library(sf)
pts = read_sf(system.file('extdata/pts.gpkg',package = 'sdsfun'))
wt = inverse_distance_swm(pts)
wt[1:5,1:5]
```

---

normalize_vector	*normalization*

Description

normalization

Usage

```
normalize_vector(x, to_left = 0, to_right = 1)
```

Arguments

x	A continuous numeric vector.
to_left	(optional) Specified minimum. Default is 0.
to_right	(optional) Specified maximum. Default is 1.

Value

A continuous vector which has normalized.

Examples

```
normalize_vector(c(-5,1,5,0.01,0.99))
```

---

sf_geometry_name	*sf object geometry column name*

Description

Get the geometry column name of an sf object

Usage

```
sf_geometry_name(sfj)
```

Arguments

sfj	An sf object.

Value

A character.

Examples

```
library(sf)
snnu = read_sf(system.file('extdata/snnu.geojson',package = 'sdsfun'))
sf_geometry_name(snnu)
```

---

sf_geometry_type   *sf object geometry type*

Description

Get the geometry type of an sf object

Usage

```
sf_geometry_type(sfj)
```

Arguments

sfj   An sf object.

Value

A lowercase character vector

Examples

```
library(sf)
snnu = read_sf(system.file('extdata/snnu.geojson',package = 'sdsfun'))
sf_geometry_type(snnu)
```

---

sf_voronoi_diagram   *generates voronoi diagram*

Description

Generates Voronoi diagram (Thiessen polygons) for sf object

Usage

```
sf_voronoi_diagram(sfj)
```

Arguments

sfj   An sf object.

Value

An sf object of polygon geometry type.

Note

Only sf objects of (multi-)point type are supported to generate voronoi diagram and the returned result includes only the geometry column.

Examples

```
library(sf)
pts = read_sf(system.file('extdata/pts.gpkg',package = 'sdsfun'))
pts_v = sf_voronoi_diagram(pts)

library(ggplot2)
ggplot() +
  geom_sf(data = pts_v, color = 'red',
          fill = 'transparent') +
  geom_sf(data = pts, color = 'blue', size = 1.25) +
  theme_void()
```

---

spdep_contiguity_swm   *constructs spatial weight matrices based on contiguity*

Description

Constructs spatial weight matrices based on contiguity via spdep package.

Usage

```
spdep_contiguity_swm(
  sfj,
  queen = TRUE,
  k = NULL,
  order = 1L,
  cumulate = TRUE,
  style = "W",
  zero.policy = TRUE
)
```

Arguments

sfj	An sf object.
queen	(optional) if TRUE, using queen contiguity, otherwise rook contiguity. Default is TRUE.
k	(optional) The number of nearest neighbours. Ignore this parameter when not using distance based neighbours to construct spatial weight matrices.

order	(optional) The order of the adjacency object. Default is 1.
cumulate	(optional) Whether to accumulate adjacency objects. Default is TRUE.
style	(optional) `style` can take values W, B, C, and S. More to see `spdep::nb2mat()`. Default is W.
zero.policy	(optional) if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors. Default is TRUE.

Value

A matrix

Note

When `k` is set to a positive value, using K-Nearest Neighbor Weights.

Examples

```
library(sf)
pts = read_sf(system.file('extdata/pts.gpkg',package = 'sdsfun'))

wt1 = spdep_contiguity_swm(pts, k = 6, style = 'B')
wt2 = spdep_contiguity_swm(pts, queen = TRUE, style = 'B')
wt2 = spdep_contiguity_swm(pts, queen = FALSE, order = 2, style = 'B')
```

---

spdep_distance_swm   *constructs spatial weight matrices based on distance*

Description

Constructs spatial weight matrices based on distance via `spdep` package.

Usage

```
spdep_distance_swm(
  sfj,
  kernel = NULL,
  k = NULL,
  bandwidth = NULL,
  power = 1,
  style = "W",
  zero.policy = TRUE
)
```

Arguments

`sfj`	An sf object.
`kernel`	(optional) The kernel function, can be one of `uniform`, `triangular`,`quadratic`(epanechnikov),`quartic` and `gaussian`. Default is `NULL`.
`k`	(optional) The number of nearest neighbours. Default is `NULL`. Only useful when `kernel` is provided.
`bandwidth`	(optional) The bandwidth, default is `NULL`.
`power`	(optional) Default is `1`. Useful when `kernel` is not provided.
`style`	(optional) `style` can take values `W`, `B`, `C`, and `S`. More to see `spdep::nb2mat()`. Default is `W`.
`zero.policy`	(optional) if `FALSE` stop with error for any empty neighbour sets, if `TRUE` permit the weights list to be formed with zero-length weights vectors. Default is `TRUE`.

Details

five different kernel weight functions:

- uniform: $K_{(z)} = 1/2$,for $|z| < 1$
- triangular $K_{(z)} = 1 - |z|$,for $|z| < 1$
- quadratic (epanechnikov) $K_{(z)} = \frac{3}{4}\left(1 - z^2\right)$,for $|z| < 1$
- quartic $K_{(z)} = \frac{15}{16}\left(1 - z^2\right)^2$,for $|z| < 1$
- gaussian $K_{(z)} = (2\pi)^{\frac{1}{2}}e^{-\frac{z^2}{2}}$

For the equation above, $z = d_{ij}/h_i$ where h_i is the bandwidth

Value

A matrix

Note

When `kernel` is setting, using distance weight based on kernel function.

For spatial weights based on distance functions, a style of B means using the original value of the calculated distance function.

Examples

```
library(sf)
pts = read_sf(system.file('extdata/pts.gpkg',package = 'sdsfun'))

wt1 = spdep_distance_swm(pts, style = 'B')
wt2 = spdep_distance_swm(pts, kernel = 'gaussian')
wt2 = spdep_distance_swm(pts, k = 3, kernel = 'gaussian')
```

standardize_vector   *standardization*

Description

To calculate the Z-score using variance normalization, the formula is as follows:
$Z = \frac{(x - mean(x))}{sd(x)}$

Usage

```
standardize_vector(x)
```

Arguments

x   A numeric vector

Value

A standardized numeric vector

Examples

```
standardize_vector(1:10)
```

Index