Package 'sfdep'

April 20, 2022

Title Spatial Dependence for Simple Features

Version 0.1.0

Description An interface to 'spdep' to integrate with 'sf' objects and the 'tidyverse'.

License GPL-3

URL https://sfdep.josiahparry.com,
https://github.com/josiahparry/sfdep

Suggests broom, dbscan, dplyr, ggplot2, knitr, patchwork, purrr, rmarkdown, sfnetworks, stringr, testthat (>= 3.0.0), tibble, tidyr, vctrs, yaml

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.2

Imports sf, cli, spdep, magrittr, stats, rlang

Depends R (>= 3.5.0)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Josiah Parry [aut, cre] (<https://orcid.org/0000-0001-9910-865X>)

Maintainer Josiah Parry <josiah.parry@gmail.com>

Repository CRAN

Date/Publication 2022-04-20 08:10:02 UTC

R topics documented:

`cond_permute_nb` *Conditional permutation of neighbors*

Description

Creates a conditional permutation of neighbors list holding i fixed and shuffling it's neighbors.

Usage

```
cond_permute_nb(nb, seed = NULL)
```

Arguments

nb	a neighbor list.
seed	default null. A value to pass to `set.seed()` for reproducibility.

Value

A list of class nb where each element contains a random sample of neighbors excluding the observed regioin.

Examples

```
nb <- st_contiguity(guerry)
nb[1:5]
# conditionally permute neighbors
perm_nb <- cond_permute_nb(nb)
perm_nb[1:5]
```

---

critical_threshold *Identify critical threshold*

Description

Identifies the minimum distance in which each observation will have at least one neighbor.

Usage

```
critical_threshold(geometry)
```

Arguments

geometry	an sf geometry column

Value

a numeric scalar value.

Examples

```
critical_threshold(sf::st_geometry(guerry))
```

find_xj *Identify xj values*

Description

Find xj values given a numeric vector, x, and neighbors list, nb.

Usage

```
find_xj(x, nb)
```

Arguments

x	a vector of any class
nb	a nb object e.g. created by st_contiguity() or st_knn()

Value

A list of length x where each element is a numeric vector with the same length as the corresponding element in nb.

Examples

```
nb <- st_contiguity(sf::st_geometry(guerry))
xj <- find_xj(guerry$crime_prop, nb)
xj[1:3]
```

global_c *Compute Geary's C*

Description

Compute Geary's C

Usage

```
global_c(x, nb, wt, allow_zero = NULL)
```

Arguments

x	A numeric vector.
nb	a neighbor list object for example as created by st_contiguity().
wt	a weights list as created by st_weights().
allow_zero	If TRUE, assigns zero as lagged value to zone without neighbors.

Value

a list with two names elements C and K returning the value of Geary's C and sample kurtosis respectively.

See Also

Other global_c: global_c_perm(), global_c_test()

Examples

```
nb <- guerry_nb$nb
wt <- guerry_nb$wt
x <- guerry_nb$crime_pers
global_c(x, nb, wt)
```

---

global_c_perm *Global C Permutation Test*

Description

Global C Permutation Test

Usage

```
global_c_perm(
  x,
  nb,
  wt,
  nsim = 499,
  alternative = "greater",
  allow_zero = NULL,
  ...
)
```

Arguments

x	A numeric vector.
nb	a neighbor list object for example as created by st_contiguity().
wt	a weights list as created by st_weights().
nsim	number of simulations to run.
alternative	default "two.sided". Should be one of "greater", "less", or "two.sided" to specify the alternative hypothesis.
allow_zero	If TRUE, assigns zero as lagged value to zone without neighbors.
...	additional arguments passed to spdep::geary.mc().

Value

an object of classes `htest` and `mc.sim`

See Also

Other global_c: `global_c_test()`, `global_c()`

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb)
x <- guerry$crime_pers
global_c_perm(x, nb, wt)
```

---

global_c_test *Global C Test*

Description

Global C Test

Usage

```
global_c_test(x, nb, wt, randomization = TRUE, allow_zero = NULL, ...)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`randomization`	default `TRUE`. Calculate variance based on randomization. If `FALSE`, under the assumption of normality.
`allow_zero`	If `TRUE`, assigns zero as lagged value to zone without neighbors.
`...`	additional arguments passed to `spdep::moran.mc()`

Value

an `htest` object

See Also

Other global_c: `global_c_perm()`, `global_c()`

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb)
x <- guerry$crime_pers
global_c_test(x, nb, wt)
```

`global_g_test` *Getis-Ord Global G*

Description

Getis-Ord Global G

Usage

```
global_g_test(x, nb, wt, alternative = "greater", allow_zero = NULL, ...)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`alternative`	default `"two.sided"`. Should be one of `"greater"`, `"less"`, or `"two.sided"` to specify the alternative hypothesis.
`allow_zero`	If `TRUE`, assigns zero as lagged value to zone without neighbors.
`...`	additional methods passed to `spdep::globalG.test()`.

Value

an `htest` object

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb, style = "B")
x <- guerry$crime_pers
global_g_test(x, nb, wt)
```

`global_jc_perm` *Global Join Count Permutation Test*

Description

Global Join Count Permutation Test

Global Join Count Test

Usage

```
global_jc_perm(
  fx,
  nb,
  wt,
  alternative = "greater",
  nsim = 499,
  allow_zero = FALSE,
  ...
)

global_jc_test(fx, nb, wt, alternative = "greater", allow_zero = NULL, ...)
```

Arguments

`fx`	a factor or character vector of the same length as nb.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`alternative`	default `"two.sided"`. Should be one of `"greater"`, `"less"`, or `"two.sided"` to specify the alternative hypothesis.
`nsim`	number of simulations to run.
`allow_zero`	If `TRUE`, assigns zero as lagged value to zone without neighbors.
`...`	additional arguments passed to `spdep::joincount.test()`

Value

an object of class `jclist` which is a list where each element is of class `htest` and `mc.sim`.

an object of class `jclist` which is a list where each element is of class `htest` and `mc.sim`.

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb, style = "B")
fx <- guerry$region
global_jc_perm(fx, nb, wt)
```

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb, style = "B")
fx <- guerry$region
global_jc_test(fx, nb, wt)
global_jc_perm(fx, nb, wt)
```

`global_moran` *Calculate Global Moran's I*

Description

Calculate Global Moran's I

Usage

```
global_moran(x, nb, wt, na_ok = FALSE, ...)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`na_ok`	default `FALSE`. If `FALSE` presence or `NA` or `Inf` results in an error.
`...`	additional arguments passed to `spdep::moran()`.

Value

an `htest` object

See Also

Other global_moran: `global_moran_bv()`, `global_moran_perm()`, `global_moran_test()`, `local_moran_bv()`

Examples

```
nb <- guerry_nb$nb
wt <- guerry_nb$wt
x <- guerry_nb$crime_pers
moran <- global_moran(x, nb, wt)
```

`global_moran_bv` *Compute the Global Bivariate Moran's I*

Description

Given two continuous numeric variables, calculate the bivariate Moran's I.

Usage

```
global_moran_bv(x, y, nb, wt, nsim = 99)
```

Arguments

`x`	a numeric vector of same length as `nb`.
`y`	a numeric vector of same length as `nb`.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`nsim`	the number of simulations to run.

Details

$$I_B = \frac{\Sigma_i(\Sigma_j w_{ij} y_j \times x_i)}{\Sigma_i x_i^2}$$

Value

a named list with two elements `Ib` and `p_sim` containing the bivariate Moran'sI and simulated p-value respectively.

See Also

Other global_moran: `global_moran_perm()`, `global_moran_test()`, `global_moran()`, `local_moran_bv()`

Examples

```
x <- guerry_nb$crime_pers
y <- guerry_nb$wealth
nb <- guerry_nb$nb
wt <- guerry_nb$wt
global_moran_bv(x, y, nb, wt)
```

`global_moran_perm` *Global Moran Permutation Test*

Description

Global Moran Permutation Test

Usage

```
global_moran_perm(x, nb, wt, alternative = "two.sided", nsim = 499, ...)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`alternative`	default `"two.sided"`. Should be one of `"greater"`, `"less"`, or `"two.sided"` to specify the alternative hypothesis.
`nsim`	number of simulations to run.
`...`	additional arguments passed to `spdep::moran.mc()`

Value

an object of classes `htest`, and `mc.sim`.

See Also

Other global_moran: `global_moran_bv()`, `global_moran_test()`, `global_moran()`, `local_moran_bv()`

Examples

```
nb <- guerry_nb$nb
wt <- guerry_nb$wt
x <- guerry_nb$crime_pers
moran <- global_moran_perm(x, nb, wt)
broom::tidy(moran)
```

`global_moran_test` *Global Moran Test*

Description

Global Moran Test

Usage

```
global_moran_test(
  x,
  nb,
  wt,
  alternative = "greater",
  randomization = TRUE,
  ...
)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`alternative`	default `"two.sided"`. Should be one of `"greater"`, `"less"`, or `"two.sided"` to specify the alternative hypothesis.
`randomization`	default `TRUE`. Calculate variance based on randomization. If `FALSE`, under the assumption of normality.
`...`	additional arguments passed to `spdep::moran.mc()`

Value

an object of class `htest`

See Also

Other global_moran: `global_moran_bv()`, `global_moran_perm()`, `global_moran()`, `local_moran_bv()`

Examples

```
nb <- guerry_nb$nb
wt <- guerry_nb$wt
x <- guerry_nb$crime_pers
global_moran_test(x, nb, wt)
```

guerry *"Essay on the Moral Statistics of France" data set.*

Description

This dataset has been widely used to demonstrate geospatial methods and techniques. As such it is useful for inclusion to this R package for the purposes of example. The dataset in this package is modified from Guerry by Michael Friendly.

Usage

```
guerry

guerry_nb
```

Format

An object of class `sf` (inherits from `tbl_df`, `tbl`, `data.frame`) with 85 rows and 27 columns.

`guerry` an sf object with 85 observations and 27 variables. `guerry_nb` has 2 additional variables created by `sfdep`.

Details

`guerry` and `guerry_nb` objects are sf class objects. These are polygons of the boundaries of France (excluding Corsica) as they were in 1830.

Source

`Guerry::gfrance85`

include_self *Includes self in neighbor list*

Description

Includes observed region in list of own neighbors. For some neighbor lists, it is important to include the ith observation (or self) in the neighbors list, particulalry for kernel weights.

Usage

```
include_self(nb)

remove_self(nb)
```

Arguments

nb	an object of class nb e.g. made by st_contiguity()

Value

An object of class nb.

Examples

```
nb <- st_contiguity(guerry)
self_included <- include_self(nb)
self_included
remove_self(self_included)
```

---

local_c *Compute Local Geary statistic*

Description

The Local Geary is a local adaptation of Geary's C statistic of spatial autocorrelation. The Local Geary uses squared differences to measure dissimilarity unlike the Local Moran. Low values of the Local Geary indicate positive spatial autocorrelation and large refers to negative spatial autocorrelation. Inference for the Local Geary is based on a permutation approach which compares the observed value to the reference distribution under spatial randomness. The Local Geary creates a pseudo p-value. This is not an analytical p-value and is based on the number of permutations and as such should be used with care.

Usage

```
local_c(x, nb, wt, ...)

local_c_perm(x, nb, wt, nsim = 499, alternative = "two.sided", ...)
```

Arguments

x	a numeric vector, or list of numeric vectors of equal length.
nb	a neighbor list
wt	a weights list
...	other arguments passed to spdep::localC_perm(), e.g. zero.policy = TRUE to allow for zones without neighbors.
nsim	The number of simulations used to generate reference distribution.
alternative	A character defining the alternative hypothesis. Must be one of "two.sided", "less" or "greater".

Details

Overview:

The Local Geary can be extended to a multivariate context. When `x` is a numeric vector, the univariate Local Geary will be calculated. To calculate the multivariate Local Moran provide either a list or a matrix. When `x` is a list, each element must be a numeric vector of the same length and of the same length as the neighbours in `listw`. In the case that `x` is a matrix the number of rows must be the same as the length of the neighbours in `listw`.

While not required in the univariate context, the standardized Local Geary is calculated. The multivariate Local Geary is *always* standardized.

The univariate Local Geary is calculated as $c_i = \sum_j w_{ij}(x_i - x_j)^2$ and the multivariate Local Geary is calculated as $c_{k,i} = \sum_{v=1}^{k} c_{v,i}$ as described in Anselin (2019).

Implementation:

These functions are based on the implementations of the local Geary statistic in the development version of spdep. They are based on spdep::localC and spdep::localC_perm.

spdep::localC_perm and thus local_c_perm utilize a conditional permutation approach to approximate a reference distribution where each observation `i` is held fixed, randomly samples neighbors, and calculated the local C statistic for that tuple (`ci`). This is repeated `nsim` times. From the simulations 3 different types of p-values are calculated—all of which have their potential flaws. So be *extra judicious* with using p-values to make conclusions.

- `p_ci`: utilizes the sample mean and standard deviation. The p-value is then calculated using `pnorm()`–assuming a normal distribution which isn't always true.
- `p_ci_sim`: uses the rank of the observed statistic.
- `p_folded_sim`: follows the pysal implementation where p-values are in the range of [0, 0.5]. This excludes 1/2 of all p-values and should be used with caution.

Value

a `data.frame` with columns

- `ci`: Local Geary statistic
- `e_ci`: expected value of the Local Geary based on permutations
- `z_ci`: standard deviation based on permutations
- `var_ci`: variance based on permutations
- `p_ci`: p-value based on permutation sample standard deviation and means
- `p_ci_sim`: p-value based on rank of observed statistic
- `p_folded_sim`: p-value based on the implementation of Pysal which always assumes a two-sided test taking the minimum possible p-value
- `skewness`: sample skewness
- `kurtosis`: sample kurtosis

Author(s)

Josiah Parry, <josiah.parry@gmail.com>

References

Anselin, L. (1995), Local Indicators of Spatial Association—LISA. Geographical Analysis, 27: 93-115. doi: 10.1111/j.15384632.1995.tb00338.x

Anselin, L. (2019), A Local Indicator of Multivariate Spatial Association: Extending Geary's c. Geogr Anal, 51: 133-150. doi: 10.1111/gean.12164

Examples

```
guerry %>%
  dplyr::transmute(nb = st_contiguity(geometry),
                   wt = st_weights(nb),
                   geary = local_c_perm(
                     x = list(crime_pers, literacy),
                     nb, wt
                   )) %>%
  tidyr::unnest(geary)
```

`local_g` *Local G*

Description

Local G

Usage

```
local_g(x, nb, wt, alternative = "two.sided", ...)

local_g_perm(x, nb, wt, nsim = 499, alternative = "two.sided", ...)
```

Arguments

`x`	A numeric vector.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`alternative`	default `"two.sided"`. Should be one of `"greater"`, `"less"`, or `"two.sided"` to specify the alternative hypothesis.
`...`	methods passed to `spdep::localG()` or `spdep::localG_perm()`
`nsim`	The number of simulations to run.

Value

a data.frame with columns:

- gi: the observed statistic
- e_gi: the permutation sample mean
- var_gi: the permutation sample variance
- p_value: the p-value using sample mean and standard deviation
- p_folded_sim: p-value based on the implementation of Pysal which always assumes a two-sided test taking the minimum possible p-value
- skewness: sample skewness
- kurtosis: sample kurtosis

Examples

```
x <- guerry$crime_pers
nb <- st_contiguity(guerry)
wt <- st_weights(nb)

res <- local_g_perm(x, nb, wt)

head(res)
```

---

local_gstar *Local G**

Description

Local G*

Usage

```
local_gstar(x, nb, wt, alternative = "two.sided", ...)

local_gstar_perm(x, nb, wt, nsim = 499, alternative = "two.sided", ...)
```

Arguments

x	A numeric vector.
nb	a neighbor list object for example as created by st_contiguity().
wt	a weights list as created by st_weights().
alternative	default "two.sided". Should be one of "greater", "less", or "two.sided" to specify the alternative hypothesis.
...	methods passed to spdep::localG() or spdep::localG_perm()
nsim	The number of simulations to run.

Value

a `data.frame` with columns:

- `gi`: the observed statistic
- `e_gi`: the permutation sample mean
- `var_gi`: the permutation sample variance
- `p_value`: the p-value using sample mean and standard deviation
- `p_folded_sim`: p-value based on the implementation of Pysal which always assumes a two-sided test taking the minimum possible p-value
- `skewness`: sample skewness
- `kurtosis`: sample kurtosis

Examples

```
nb <- st_contiguity(guerry)
wt <- st_weights(nb)
x <- guerry$crime_pers

res <- local_gstar_perm(x, nb, wt)
head(res)

res <- local_gstar(x, nb, wt)
head(res)
```

---

local_jc_bv *Bivariate local join count*

Description

Bivariate local join count

Usage

```
local_jc_bv(x, z, nb, wt, nsim = 499)
```

Arguments

`x`	a binary variable either numeric or logical
`z`	a binary variable either numeric or logical
`nb`	a neighbors list object.
`wt`	default `st_weights(nb,style = "B")`. A binary weights list as created by `st_weights(nb,style = "B")`.
`nsim`	the number of conditional permutation simulations

Value

a data.frame with two columns join_count and p_sim and number of rows equal to the length of arguments x, z, nb, and wt.

Examples

```
x <- as.integer(guerry$infants > 23574)
z <- as.integer(guerry$donations > 10973)
nb <- st_contiguity(guerry)
wt <- st_weights(nb, style = "B")
local_jc_bv(x, z, nb, wt)
```

---

local_jc_uni *Compute local univariate join count*

Description

The univariate local join count statistic is used to identify clusters of rarely occurring binary variables. The binary variable of interest should occur less than half of the time.

Usage

```
local_jc_uni(
  x,
  nb,
  wt = st_weights(nb, style = "B"),
  nsim = 499,
  alternative = "two.sided"
)
```

Arguments

x	a binary variable either numeric or logical
nb	a neighbors list object.
wt	default st_weights(nb,style = "B"). A binary weights list as created by st_weights(nb,style = "B").
nsim	the number of conditional permutation simulations
alternative	default "greater". One of "less" or "greater".

Details

The local join count statistic requires a binary weights list which can be generated with st_weights(nb,style = "B"). Additionally, ensure that the binary variable of interest is rarely occurring in no more than half of observations.

P-values are estimated using a conditional permutation approach. This creates a reference distribution from which the observed statistic is compared. For more see Geoda Glossary.

Value

a data.frame with two columns join_count and p_sim and number of rows equal to the length of arguments x, nb, and wt.

Examples

```
guerry %>%
  dplyr::transmute(top_crime = crime_prop > 9000,
                   nb = st_contiguity(geometry),
                   wt = st_weights(nb, style = "B"),
                   jc = local_jc_uni(top_crime, nb, wt)) %>%
  tidyr::unnest(jc)
```

---

local_moran *Calculate the Local Moran's I Statistic*

Description

Moran's I is calculated for each polygon based on the neighbor and weight lists.

Usage

```
local_moran(x, nb, wt, alternative = "two.sided", nsim = 499, ...)
```

Arguments

x	A numeric vector.
nb	a neighbor list object for example as created by st_contiguity().
wt	a weights list as created by st_weights().
alternative	default "two.sided". Should be one of "greater", "less", or "two.sided" to specify the alternative hypothesis.
nsim	The number of simulations to run.
...	See ?spdep::localmoran_perm() for more options.

Details

local_moran() calls spdep::localmoran_perm() and calculates the Moran I for each polygon. As well as provide simulated p-values.

Value

a data.frame containing the columns ii, eii, var_ii, z_ii, p_ii, p_ii_sim, and p_folded_sim. For more details please see spdep::localmoran_perm().

See Also

Other stats: st_lag()

Examples

```
lisa <- guerry %>%
  dplyr::mutate(nb = st_contiguity(geometry),
         wt = st_weights(nb),
         moran = local_moran(crime_pers, nb, wt))

# unnest the dataframe column
tidyr::unnest(lisa, moran)
```

`local_moran_bv` *Compute the Local Bivariate Moran's I Statistic*

Description

Given two continuous numeric variables, calculate the bivariate Local Moran's I.

Usage

```
local_moran_bv(x, y, nb, wt, nsim = 499)
```

Arguments

`x`	a numeric vector of same length as `nb`.
`y`	a numeric vector of same length as `nb`.
`nb`	a neighbor list object for example as created by `st_contiguity()`.
`wt`	a weights list as created by `st_weights()`.
`nsim`	the number of simulations to run.

Details

$I_i^B = cx_i \Sigma_j w_{ij} y_j$

Value

a `data.frame` containing two columns `Ib` and `p_sim` containing the local bivariate Moran's I and simulated p-values respectively.

See Also

Other global_moran: `global_moran_bv()`, `global_moran_perm()`, `global_moran_test()`, `global_moran()`

Examples

```
x <- guerry_nb$crime_pers
y <- guerry_nb$wealth
nb <- guerry_nb$nb
wt <- guerry_nb$wt
local_moran_bv(x, y, nb, wt)
```

losh *Local spatial heteroscedacity*

Description

Local spatial heteroscedacity

Usage

```
losh(x, nb, wt, a = 2, ...)

losh_perm(x, nb, wt, a = 2, nsim = 499, ...)
```

Arguments

x	a numeric vector.
nb	a neighbor list for example created by st_contiguity()
wt	a weights list for example created by st_weights()
a	the exponent applied to the local residuals
...	methods passed to spdep::LOSH
nsim	number of simulations to run

Value

a data.frame with columns

- hi: the observed statistic
- e_hi: the sample average
- var_hi: the sample variance
- z_hi the approximately Chi-square distributed test statistic
- x_bar_i: the local spatially weight mean for observation i
- ei: residuals

Examples

```
nb <- st_contiguity(guerry)
wt <- st_weights(nb)
x <- guerry$crime_pers
losh(x, nb, wt)
losh(x, nb, wt, var_hi = FALSE)
losh_perm(x, nb, wt, nsim = 49)
```

nb_match_test *Local Neighbor Match Test*

Description

Implements the Local Neighbor Match Test as described in *Tobler's Law in a Multivariate World* (Anselin and Li, 2020).

Usage

```
nb_match_test(
  x,
  nb,
  wt = st_weights(nb),
  k = 10,
  nsim = 499,
  scale = TRUE,
  .method = "euclidian",
  .p = 2
)
```

Arguments

x	a numeric vector or a list of numeric vectors of equal length.
nb	a neighbor list object for example as created by st_contiguity().
wt	a weights list as created by st_weights().
k	the number of neighbors to identify in attribute space. Should be the same as number of neighbors provided in st_knn.
nsim	the number of simulations to run for calculating the simulated p-value.
scale	default TRUE. Whether x should be scaled or not. Note that measures should be standardized.
.method	default "euclidian". The distance measure passed to stats::dist().
.p	default 2. The power of Minkowski distance passed to the p argument in stats::dist().

Value

a data.frame with columns

- n_shared (integer): the number of shared neighbors between geographic and attribute space
- nb_matches (list): matched neighbor indexes. Each element is an iteger vector of same length as the ith observation of n_shared
- knn_nb (list): the neighbors in attribute space
- probability (numeric): the geometric probability of observing the number of matches
- p_sim (numeric): a folded simulated p-value

Examples

```
guerry %>%
  dplyr::transmute(nb = st_knn(geometry, k = 10),
         nmt = nb_match_test(list(crime_pers, literacy, suicides),
                             nb, nsim = 999)) %>%
  tidyr::unnest(nmt)
```

`st_as_edges` *Convert to an edge lines object*

Description

Given geometry and neighbor and weights lists, create an edge list `sf` object.

Usage

```
st_as_edges(x, nb, wt)

## S3 method for class 'sf'
st_as_edges(x, nb, wt)

## S3 method for class 'sfc'
st_as_edges(x, nb, wt)
```

Arguments

`x`	object of class `sf` or `sfc`.
`nb`	a neighbor list. If `x` is class `sf`, the unquote named of the column. If `x` is class `sfc`, an object of class `nb` as created from `st_contiguity()`.
`wt`	optional. A weights list as generated by `st_weights()`. . If `x` is class `sf`, the unquote named of the column. If `x` is class `sfc`, the weights list itself.

Details

Creating an edge list creates a column for each `i` position and `j` between an observation and their neighbors. You can recreate these values by expanding the `nb` and `wt` list columns.

```
guerry_nb %>%
  tibble::as_tibble() %>%
  dplyr::select(nb, wt) %>%
  dplyr::mutate(i = dplyr::row_number(), .before = 1) %>%
  tidyr::unnest(c(nb, wt))

## # A tibble: 420 × 3
##        i    nb    wt
##    <int> <int> <dbl>
##  1     1    36 0.25
```

```
##  2     1    37 0.25
##  3     1    67 0.25
##  4     1    69 0.25
##  5     2     7 0.167
##  6     2    49 0.167
##  7     2    57 0.167
##  8     2    58 0.167
##  9     2    73 0.167
## 10     2    76 0.167
## # . . .  with 410 more rows
```

Value

Returns an `sf` object with edges represented as a `LINESTRING`.

- `from`: node index. This is the row position of `x`.
- `to`: node index. This is the neighbor value stored in `nb`.
- `i`: node index. This is the row position of `x`.
- `j`: node index. This is the neighbor value stored in `nb`.
- `wt`: the weight value of `j` stored in `wt`.

Examples

```
guerry %>%
  dplyr::mutate(nb = st_contiguity(geometry),
         wt = st_weights(nb)) %>%
  st_as_edges(nb, wt)
```

`st_as_graph` *Create an sfnetwork*

Description

Given an `sf` or `sfc` object and neighbor and weights lists, create an `sfnetwork` object.

Usage

```
st_as_graph(x, nb, wt)

## S3 method for class 'sf'
st_as_graph(x, nb, wt)

## S3 method for class 'sfc'
st_as_graph(x, nb, wt)
```

Arguments

x	object of class sf or sfc.
nb	a neighbor list. If x is class sf, the unquote named of the column. If x is class sfc, an object of class nb as created from st_contiguity().
wt	optional. A weights list as generated by st_weights(). . If x is class sf, the unquote named of the column. If x is class sfc, the weights list itself.

See Also

st_as_nodes() and st_as_edges()

Examples

```
guerry_nb %>%
  st_as_graph(nb, wt)
```

---

st_as_nodes *Convert to a node point object*

Description

Given geometry and a neighbor list, creates an sf object to be used as nodes in an sfnetworks::sfnetwork(). If the provided geometry is a polygon, sf::st_point_on_surface() will be used to create the node point.

Usage

```
st_as_nodes(x, nb)

## S3 method for class 'sf'
st_as_nodes(x, nb)

## S3 method for class 'sfc'
st_as_nodes(x, nb)
```

Arguments

x	object of class sf or sfc.
nb	a neighbor list. If x is class sf, the unquote named of the column. If x is class sfc, an object of class nb as created from st_contiguity().

Details

st_as_node() adds a row i based on the attribute "region.id" in the nb object. If the nb object is created with sfdep, then the values will always be row indexes.

Value

An object of class `sf` with `POINT` geometry.

Examples

```
guerry %>%
  dplyr::transmute(nb = st_contiguity(geometry)) %>%
  st_as_nodes(nb)
```

`st_cardinalties` *Calculate neighbor cardinalities*

Description

Identify the cardinality of a neighbor object. Utilizes `spdep::card()` for objects with class `nb`, otherwise returns `lengths(nb)`.

Usage

```
st_cardinalties(nb)
```

Arguments

`nb`	A neighbor list object as created by `st_neighbors()`.

Value

an integer vector with the same length as `nb`.

See Also

Other other: `st_nb_lag_cumul()`, `st_nb_lag()`

Examples

```
nb <- st_contiguity(sf::st_geometry(guerry))
st_cardinalties(nb)
```

st_contiguity *Identify polygon neighbors*

Description

Given an sf geometry of type POLYGON or MULTIPOLYGON identify contiguity based neighbors.

Usage

```
st_contiguity(x, queen = TRUE, ...)
```

Arguments

x	an sf or sfc object.
queen	default TRUE. For more see ?spdep::poly2nb
...	additional arguments passed to spdep::poly2nb()

Details

Utilizes spdep::poly2nb()

Value

a list of class nb

See Also

Other neighbors: st_dist_band(), st_knn()

Examples

```
# on basic polygons
geo <- sf::st_geometry(guerry)
st_contiguity(geo)

# in a pipe
guerry %>%
  dplyr::mutate(nb = st_contiguity(geometry), .before = 1)
```

st_dist_band *Neighbors from a distance band*

Description

Creates neighbors based on a distance band. By default, creates a distance band with the maximum distance of k-nearest neighbors where k = 1 (the critical threshold) to ensure that there are no regions that are missing neighbors.

Usage

```
st_dist_band(geometry, lower = 0, upper = critical_threshold(geometry), ...)
```

Arguments

geometry	An sf or sfc object.
lower	The lower threshold of the distance band. It is recommended to keep this as 0.
upper	The upper threshold of the distance band. By default is set to a critical threshold using critical_threshold() ensuring that each region has a minimum of one neighbor.
...	Passed to spdep::dnearneigh().

Value

a list of class nb

See Also

Other neighbors: st_contiguity(), st_knn()

Examples

```
geo <- sf::st_geometry(guerry)
st_dist_band(geo, upper = critical_threshold(geo))
```

st_inverse_distance *Calculate inverse distance weights*

Description

From a neighbor list and sf geometry column, calculate inverse distance weight.

Usage

```
st_inverse_distance(nb, geometry, scale = 100, alpha = 1)
```

Arguments

nb	a neighbors list object e.g. created by st_knn() or st_contiguity()
geometry	sf geometry
scale	default 100.a value to scale distances by before exponentiating by alpha
alpha	default 1. Set to 2 for gravity weights.

Details

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

Value

a list where each element is a numeric vector

See Also

Other weights: st_kernel_weights(), st_nb_dists(), st_weights()

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wts <- st_inverse_distance(nb, geo)
head(wts, 3)
wts <- st_inverse_distance(nb, geo, scale = 10000)
head(wts, 3)
```

---

st_kernel_weights *Calculate Kernel Weights*

Description

Create a weights list using a kernel function.

Usage

```
st_kernel_weights(
  nb,
  geometry,
  kernel = "uniform",
  threshold = critical_threshold(geometry),
  adaptive = FALSE,
  self_kernel = FALSE
)
```

Arguments

nb	an object of class nb e.g. created by st_contiguity() or st_knn().
geometry	the geometry an sf object.
kernel	One of "uniform", "gaussian", "triangular", "epanechnikov", or "quartic". See kernels for more.
threshold	a scaling threshold to be used in calculating
adaptive	default FALSE. If TRUE uses the maximum neighbor distance for each region as the threshold. Suppresses the threshold argument.
self_kernel	default FALSE. If TRUE applies the kernel function to the observed region.

Details

By default st_kernel_weight() utilizes a critical threshold of the maximum neighbor distance using critical_threshold(). If desired, the critical threshold can be specified manually. The threshold will be passed to the underlying kernel.

Value

a list where each element is a numeric vector.

See Also

Other weights: st_inverse_distance(), st_nb_dists(), st_weights()

Examples

```
geometry <- sf::st_geometry(guerry)
nb <- st_contiguity(geometry)
nb <- include_self(nb)
res <- st_kernel_weights(nb, geometry)
head(res, 3)
```

---

st_knn *Calculate K-Nearest Neighbors*

Description

Identifies the k nearest neighbors for given point geometry. If polygon geometry is provided, the centroids of the polygon will be used and a warning will be emitted.

Usage

```
st_knn(x, k = 1, symmetric = FALSE, ...)
```

Arguments

x	an sf or sfc object.
k	number of nearest neighbours to be returned
symmetric	default FALSE. Whether to force output of neighbours to be symmetric.
...	additional arguments to be passed to knearneigh().

Details

This function utilizes spdep::knearneigh() and spdep::knn2nb().

Value

a list of class nb

See Also

Other neighbors: st_contiguity(), st_dist_band()

Examples

```
st_knn(sf::st_geometry(guerry), k = 8)
```

---

st_lag *Calculate spatial lag*

Description

Calculates the spatial lag of a numeric variable given a neighbor and weights list.

Usage

```
st_lag(x, nb, wt, na_ok = FALSE, allow_zero = NULL, ...)
```

Arguments

x	A numeric vector
nb	A neighbor list object as created by st_neighbors().
wt	A weights list as created by st_weights().
na_ok	Default FALSE. If, TRUE missing values return a lagged NA.
allow_zero	If TRUE, assigns zero as lagged value to zone without neighbors.
...	See ?spdep::lag.listw for more.

Value

a numeric vector with same length as x

See Also

Other stats: `local_moran()`

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
wt <- st_weights(nb)

st_lag(guerry$crime_pers, nb, wt)
```

`st_nb_apply` *Apply a function to neighbors*

Description

Sometimes one may want to create custom lag variables or create some other neighborhood level metric that may not be defined yet. This `st_nb_apply()` enables you to apply a function to each observation's (xi) neighbors (xij).

Usage

```
st_nb_apply(x, nb, wt, .f, suffix = "dbl", ...)
```

Arguments

`x`	A vector that will be used for neighbor xij values.
`nb`	A neighbor list object as created by `st_neighbors()`.
`wt`	A weights list as created by `st_weights()`.
`.f`	A function definition. There are three default objects that can be used inside of the function definition: • `.xij`: neighbor values of `x` for the ith observation. This is simply the subset of `x` based on the corresponding `nb` list values for each element. • `.nb`: neighbor positions. • `.wt`: neighbor weights value. If any of these three function arguments are omitted from `.f`, dots (`...`) must be supplied.
`suffix`	The `map` variant to use. Options are "dbl", "int", "lgl", "chr", "list".
`...`	arguments to pass to `.f`

Details

The below example calculates the spatial lag using `st_nb_apply()` and `st_lag()` to illustrate how we can apply functions to neighbors.

Currently questioning the use case. `find_xj()` is now exported and may negate the need for this function.

Value

a vector or list of with same length as `x`.

Examples

```
guerry %>%
  dplyr::transmute(
    nb = st_contiguity(geometry),
    wt = st_weights(nb),
    lag_apply = st_nb_apply(
      crime_pers, nb, wt,
      .f = function(.xij, .wt, ...) sum(.xij *.wt)
    ),
    lag = st_lag(crime_pers, nb, wt)
  )
```

`st_nb_dists` *Calculate neighbor distances*

Description

From an nb list and point geometry, return a list of distances for each observation's neighbors list.

Usage

```
st_nb_dists(x, nb, longlat = NULL)
```

Arguments

`x`	an object of class `sfc`.
`nb`	a neighbor list for example created by `st_contiguity()`
`longlat`	`TRUE` if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers. See `?spdep::nbdists()` for more.

Details

Utilizes `spdep::nbdists()` for distance calculation.

Value

a list where each element is a numeric vector.

See Also

Other weights: `st_inverse_distance()`, `st_kernel_weights()`, `st_weights()`

Examples

```
geo <- sf::st_geometry(guerry)
nb <- st_contiguity(geo)
dists <- st_nb_dists(geo, nb)

head(dists)
```

`st_nb_lag` *Pure Higher Order Neighbors*

Description

Identify higher order neighbors from a neighbor list. `order` must be greater than 1. When order equals 2 then the neighbors of the neighbors list is returned and so forth. See Anselin's slides for an example.

Usage

```
st_nb_lag(nb, order)
```

Arguments

`nb`	A neighbor list object as created by `st_contiguity()`.
`order`	The order of neighbors.

Details

Utilizes `spdep::nblag()`

Value

a list of class `nb`

See Also

Other other: `st_cardinalties()`, `st_nb_lag_cumul()`

Examples

```
nb <- st_contiguity(sf::st_geometry(guerry))
st_nb_lag(nb, 3)
```

st_nb_lag_cumul *Encompassing Higher Order Neighbors*

Description

Creates an encompassing neighbor list of the order specified. For example, if the order is 2 the result contains both 1st and 2nd order neighbors.

Usage

```
st_nb_lag_cumul(nb, order)
```

Arguments

nb	A neighbor list object as created by st_contiguity().
order	The order of neighbors.

Details

Utilizes spdep::nblag_cumul()

Value

a list of class nb

See Also

Other other: st_cardinalties(), st_nb_lag()

Examples

```
nb <- st_contiguity(sf::st_geometry(guerry))
st_nb_lag_cumul(nb, 3)
```

st_weights *Calculate spatial weights*

Description

Calculate polygon spatial weights from a nb list. See spdep::nb2listw() for further details.

Usage

```
st_weights(nb, style = "W", allow_zero = NULL, ...)
```

Arguments

nb	A neighbor list object as created by st_neighbors().
style	Default "W" for row standardized weights. This value can also be "B", "C", "U", "minmax", and "S". See spdep::nb2listw() for details.
allow_zero	If TRUE, assigns zero as lagged value to zone without neighbors.
...	additional arguments passed to spdep::nb2listw().

Details

Under the hood, st_weights() creates a listw object and then extracts the weights elements from it as the neighbours element is already–presumably–already existent in the neighbors list you've already created. listw objects are recreated using recreate_listw() when calculating other statistics.

Value

a list where each element is a numeric vector

See Also

Other weights: st_inverse_distance(), st_kernel_weights(), st_nb_dists()

Examples

```
guerry %>%
 dplyr::mutate(nb = st_contiguity(geometry),
               wt = st_weights(nb),
               .before = 1)

# using geometry column directly
nb <- st_contiguity(guerry$geometry)
wt <- st_weights(nb)
wt[1:3]
```

Index