Package 'rgeoda'

February 20, 2026

Type Package

Title R Library for Spatial Data Analysis

Version 0.1.1

Date 2026-03-19

Maintainer Xun Li <`lixun910@gmail.com`>

Description Provides spatial data analysis functionalities including Exploratory Spatial Data Analysis, Spatial Cluster Detection and Clustering Analysis, Regionalization, etc. based on the C++ source code of 'GeoDa', which is an open-source software tool that serves as an introduction to spatial data analysis. The 'GeoDa' software and its documentation are available at <`https://geodacenter.github.io`>.

URL `https://github.com/geodacenter/rgeoda/`, `https://geodacenter.github.io/rgeoda/`

BugReports `https://github.com/geodacenter/rgeoda/issues/`

Depends R (>= 4.0.0), methods, digest

License GPL (>= 2)

Collate init.R rgeoda.R sf_geoda.R RcppExports.R read_geoda.R weights.R utils.R lisa.R clustering.R

Imports sf, Rcpp (>= 1.0.1)

LinkingTo Rcpp, BH (>= 1.87.0-1)

RoxygenNote 7.3.2

Encoding UTF-8

Suggests wkb, sp

SystemRequirements C++17

NeedsCompilation yes

Author Xun Li [aut, cre],
Luc Anselin [aut]

Repository CRAN

Date/Publication 2026-02-19 23:10:02 UTC

Contents

`as.data.frame.geoda` *convert rgeoda instance to data.frame*

Description

Override the as.data.frame function for rgeoda instance

Usage

```
## S3 method for class 'geoda'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A rgeoda object
`row.names`	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	optional parameters
`...`	other arguments passed to methods

Value

A data.frame object

`as.geoda` *Create an instance of geoda-class from either an 'sf' or 'sp' object*

Description

Create an instance of geoda-class from an 'sf' object returned from 'st_read()' function, or a 'sp' object returned from 'readOGR()' function. NOTE: The table content is NOT used to create an instance of geoda-class.

Usage

```
as.geoda(obj, with_table = TRUE)
```

Arguments

`obj`	An instance of 'sf' or 'sp' object
`with_table`	A boolean flag indicates if table is copied from sf object to create geoda object. Default is TRUE

Value

An instance of geoda-class

`as.matrix.Weight` *spatial weights to matrix*

Description

Convert a GeoDa spatial weights object to a Matrix object

Usage

```
## S3 method for class 'Weight'
as.matrix(x, rownames = NULL, rownames.value = NULL, ...)
```

Arguments

`x`	A weights object
`rownames`	optional, a single column name or column number to use as the rownames in the returned matrix. If TRUE the key of the data.table will be used if it is a single column, otherwise the first column in the data.table will be used.
`rownames.value`	optional, a vector of values to be used as the rownames in the returned matrix. It must be the same length as nrow(x).
`...`	Required to be present because the generic ‘as.matrix‘ generic has it. Arguments here are not currently used or passed on by this method.

Value

A matrix object

`azp_greedy` *A greedy algorithm to solve the AZP problem*

Description

The automatic zoning procedure (AZP) was initially outlined in Openshaw (1977) as a way to address some of the consequences of the modifiable areal unit problem (MAUP). In essence, it consists of a heuristic to find the best set of combinations of contiguous spatial units into p regions, minimizing the within sum of squares as a criterion of homogeneity. The number of regions needs to be specified beforehand.

Usage

```
azp_greedy(
  p,
  w,
  df,
  bound_variable = data.frame(),
  min_bound = 0,
  inits = 0,
  initial_regions = vector("numeric"),
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  rdist = numeric()
)
```

Arguments

`p`	The number of spatially constrained clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`bound_variable`	(optional) A data frame with selected bound variabl
`min_bound`	(optional) A minimum bound value that applies to all clusters
`inits`	(optional) The number of construction re-runs, which is for ARiSeL "automatic regionalization with initial seed location"
`initial_regions`	(optional) The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
```

```
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
azp_clusters <- azp_greedy(5, queen_w, data)
azp_clusters

## End(Not run)
```

`azp_sa` *A simulated annealing algorithm to solve the AZP problem*

Description

The automatic zoning procedure (AZP) was initially outlined in Openshaw (1977) as a way to address some of the consequences of the modifiable areal unit problem (MAUP). In essence, it consists of a heuristic to find the best set of combinations of contiguous spatial units into p regions, minimizing the within sum of squares as a criterion of homogeneity. The number of regions needs to be specified beforehand.

Usage

```
azp_sa(
  p,
  w,
  df,
  cooling_rate,
  sa_maxit = 1,
  bound_variable = data.frame(),
  min_bound = 0,
  inits = 0,
  initial_regions = vector("numeric"),
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  rdist = numeric()
)
```

Arguments

`p`	The number of spatially constrained clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`cooling_rate`	The cooling rate of a simulated annealing algorithm. Defaults to 0.85
`sa_maxit`	(optional): The number of iterations of simulated annealing. Defaults to 1

`bound_variable`	(optional) A data frame with selected bound variabl
`min_bound`	(optional) A minimum bound value that applies to all clusters
`inits`	(optional) The number of construction re-runs, which is for ARiSeL "automatic regionalization with initial seed location"
`initial_regions`	(optional) The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
azp_clusters <- azp_sa(5, queen_w, data, cooling_rate = 0.85)
azp_clusters

## End(Not run)
```

`azp_tabu` *A tabu algorithm to solve the AZP problem*

Description

The automatic zoning procedure (AZP) was initially outlined in Openshaw (1977) as a way to address some of the consequences of the modifiable areal unit problem (MAUP). In essence, it consists of a heuristic to find the best set of combinations of contiguous spatial units into p regions, minimizing the within sum of squares as a criterion of homogeneity. The number of regions needs to be specified beforehand.

Usage

```
azp_tabu(
  p,
  w,
  df,
  tabu_length = 10,
  conv_tabu = 10,
  bound_variable = data.frame(),
  min_bound = 0,
  inits = 0,
  initial_regions = vector(”numeric”),
  scale_method = ”standardize”,
  distance_method = ”euclidean”,
  random_seed = 123456789,
  rdist = numeric()
)
```

Arguments

`p`	The number of spatially constrained clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`tabu_length`	The length of a tabu search heuristic of tabu algorithm. e.g. 10.
`conv_tabu`	(optional): The number of non-improving moves. Defaults to 10.
`bound_variable`	(optional) A data frame with selected bound variabl
`min_bound`	(optional) A minimum bound value that applies to all clusters
`inits`	(optional) The number of construction re-runs, which is for ARiSeL "automatic regionalization with initial seed location"
`initial_regions`	(optional) The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
azp_clusters <- azp_tabu(5, queen_w, data, tabu_length=10, conv_tabu=10)
azp_clusters

## End(Not run)
```

`create_weights` *Create an empty weights*

Description

Create an empty weights

Usage

```
create_weights(num_obs)
```

Arguments

`num_obs`	The number of observations for this empty weights

Value

An instance of Weight-class

`distance_weights` *Distance-based Spatial Weights*

Description

Create a distance-based weights

Usage

```
distance_weights(
  sf_obj,
  dist_thres,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`dist_thres`	A positive numeric value of distance threshold
`power`	(optional) The power (or exponent) of a number indicates how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
dist_thres <- min_distthreshold(guerry)
dist_w <- distance_weights(guerry, dist_thres)
summary(dist_w)
```

`eb_rates` *Empirical Bayes(EB) Rate*

Description

The function to compute EB Rate from an event variable and a base variable.

Usage

```
eb_rates(df)
```

Arguments

`df`	A data frame with two selected variable: one is "event", anothor is "base" variable. E.g. guerry[c("hr60", "po60")]

Value

A data.frame with two columns "EB Rate" and "IsNull".

Examples

```
## Not run:
library(sf)
nat <- st_read("natregimes.shp")
ebr <- eb_rates(nat[c("HR60", "PO60")])
ebr

## End(Not run)
```

`eb_rates_standardization`

Empirical Bayes(EB) Rate Standardization

Description

The function to compute EB Rate Standardization from an event variable and a base variable.

Usage

```
eb_rates_standardization(df)
```

Arguments

`df`	A data frame with two selected variable: one is "event", anothor is "base" variable. E.g. guerry[c("hr60", "po60")]

Value

A data.frame with two columns "EB Rate" and "IsNull".

Examples

```
## Not run:
library(sf)
nat <- st_read("natregimes.shp")
ebr <- eb_rates_standardization(nat[c("HR60", "PO60")])
ebr

## End(Not run)
```

`gda_distance_weights` *(For internally use and test only) Distance-based Spatial Weights*

Description

Create a distance-based weights

Usage

```
gda_distance_weights(
  geoda_obj,
  dist_thres,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`geoda_obj`	An instance of geoda-class
`dist_thres`	A positive numeric value of distance threshold
`power`	(optional) The power (or exponent) of a number indicates how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
dist_thres <- gda_min_distthreshold(guerry)
dist_w <- gda_distance_weights(guerry, dist_thres)
summary(dist_w)

## End(Not run)
```

`gda_kernel_knn_weights` *(For internally use and test only) K-NN Kernel Spatial Weights*

Description

Create a kernel weights by specifying k-nearest neighbors and a kernel method

Usage

```
gda_kernel_knn_weights(
  geoda_obj,
  k,
  kernel_method,
  adaptive_bandwidth = TRUE,
  use_kernel_diagonals = FALSE,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`geoda_obj`	An instance of geoda
`k`	a positive integer number for k-nearest neighbors
`kernel_method`	a string value, which has to be one of 'triangular', 'uniform', 'epanechnikov', 'quartic', 'gaussian'
`adaptive_bandwidth`	(optional) TRUE (default) or FALSE: TRUE use adaptive bandwidth calculated using distance of k-nearest neithbors, FALSE use max distance of all observation to their k-nearest neighbors
`use_kernel_diagonals`	(optional) FALSE (default) or TRUE, apply kernel on the diagonal of weights matrix
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
adptkernel_w = gda_kernel_knn_weights(guerry, 6, "uniform")
summary(adptkernel_w)

## End(Not run)
```

`gda_kernel_weights` *(For internally use and test only) Distance-based Kernel Spatial Weights*

Description

Create a kernel weights by specifying a bandwidth and a kernel method

Usage

```
gda_kernel_weights(
  geoda_obj,
  bandwidth,
  kernel_method,
  use_kernel_diagonals = FALSE,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`geoda_obj`	An instance of geoda-class
`bandwidth`	A positive numeric value of bandwidth
`kernel_method`	a string value, which has to be one of 'triangular', 'uniform', 'epanechnikov', 'quartic', 'gaussian'
`use_kernel_diagonals`	(optional) FALSE (default) or TRUE, apply kernel on the diagonal of weights matrix
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
bandwidth <- gda_min_distthreshold(guerry)
kernel_w <- gda_kernel_weights(guerry, bandwidth, kernel_method = "uniform")
summary(kernel_w)

## End(Not run)
```

`gda_knn_weights` *(For internally use and test only) K-Nearest Neighbors-based Spatial Weights*

Description

Create a k-nearest neighbors based spatial weights

Usage

```
gda_knn_weights(
  geoda_obj,
  k,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`geoda_obj`	An instance of geoda
`k`	a positive integer number for k-nearest neighbors
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
knn6_w <- gda_knn_weights(guerry, 6)
summary(knn6_w)

## End(Not run)
```

`gda_min_distthreshold` *(For internally use and test only) Minimum Distance Threshold for Distance-based Weights*

Description

Get minimum threshold of distance that makes sure each observation has at least one neighbor

Usage

```
gda_min_distthreshold(geoda_obj, is_arc = FALSE, is_mile = TRUE)
```

Arguments

`geoda_obj`	An instance of geoda-class
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, if 'is_arc' option is TRUE, then 'is_mile' will set distance unit to 'mile' or 'km'.

Value

A numeric value of minimum threshold of distance

`gda_queen_weights` *(For internally use and test only) Queen Contiguity Spatial Weights*

Description

Create a Queen contiguity weights with options of "order", "include lower order" and "precision threshold"

Usage

```
gda_queen_weights(
  geoda_obj,
  order = 1,
  include_lower_order = FALSE,
  precision_threshold = 0
)
```

Arguments

`geoda_obj`	An object of [geoda] class
`order`	(Optional) Order of contiguity
`include_lower_order`	(Optional) Whether or not the lower order neighbors should be included in the weights structure
`precision_threshold`	(Optional) The precision of the underlying shape file is insufficient to allow for an exact match of coordinates to determine which polygons are neighbors

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
queen_w <- gda_queen_weights(guerry)
summary(queen_w)

## End(Not run)
```

`gda_rook_weights` *(For internally use and test only) Rook Contiguity Spatial Weights*

Description

Create a Rook contiguity weights with options of "order", "include lower order" and "precision threshold"

Usage

```
gda_rook_weights(
  geoda_obj,
  order = 1,
  include_lower_order = FALSE,
  precision_threshold = 0
)
```

Arguments

`geoda_obj`	An object of [geoda] class
`order`	(Optional) Order of contiguity
`include_lower_order`	(Optional) Whether or not the lower order neighbors should be included in the weights structure
`precision_threshold`	(Optional) The precision of the underlying shape file is insufficient to allow for an exact match of coordinates to determine which polygons are neighbors

Value

An instance of Weight-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
rook_w <- gda_rook_weights(guerry)
summary(rook_w)

## End(Not run)
```

`geoda-class` *'geoda' class*

Description

'geoda' is a RefClass that wraps the C++ GeoDa class (via p_GeoDa defines in rgeoda.R)

Fields

`gda`	The pointer to the instance of p_GeoDa-class
`map_type`	The map type, could be either Point or Polygon
`n_cols`	The number of columns
`n_obs`	The number of observations
`field_names`	A string vector of field names
`field_types`	A string vector of field types (integer, real, string)

Methods

`GetFieldNames(...)`	Get the field names of all columns
`GetFieldTypes(...)`	Get the field types (integer, real, string) of all columns
`GetIntegerCol(col_name)`	Get the integer values from a column
`GetMapType(...)`	Get the map type
`GetNumCols(...)`	Get the number of columns
`GetNumObs(...)`	Get the number of observations
`GetPointer()`	Get the C++ object pointer (internally used)
`GetRealCol(col_name)`	Get the real values from a column
`GetUndefinedVals(col_name)`	Get the undefined flags from a column
`initialize(o_gda)`	Constructor with a geoda object (internally used)

`geoda_open` *Create an instance of geoda-class by reading from an ESRI Shapefile dataset*

Description

Create an instance of geoda-class by reading from an ESRI Shapefile dataset.

Usage

```
geoda_open(ds_path)
```

Arguments

`ds_path`	(character) The path of the spatial dataset

Value

An instance of geoda-class

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- geoda_open(guerry_path)
guerry_df <- as.data.frame(guerry) # access as a data.frame
head(guerry_df)

## End(Not run)
```

`get_neighbors` *Neighbors of one observation*

Description

Get neighbors for idx-th observation, idx starts from 1

Usage

```
get_neighbors(gda_w, idx)
```

Arguments

`gda_w`	A Weight object
`idx`	A value indicates idx-th observation, idx start from 1

Value

A numeric vector of the neighbor indices, which start from 1

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
nbrs <- get_neighbors(queen_w, idx = 1)
cat("\nNeighbors of the 1-st observation are:", nbrs)

## End(Not run)
```

`get_neighbors_weights` *Weights values of the neighbors of one observation*

Description

Get the associated weights values of neighbors for idx-th observation

Usage

```
get_neighbors_weights(gda_w, idx)
```

Arguments

`gda_w`	A Weight object
`idx`	A value indicates idx-th observation, idx start from 1

Value

A numeric vector of the neighbor indices, which start from 1

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
nbrs <- get_neighbors_weights(queen_w, idx = 1)
cat("\nNeighbors of the 1-st observation are:", nbrs)

## End(Not run)
```

`has_isolates` *Isolation/Island in Spatial Weights*

Description

Check if weights matrix has isolates, or if any observation has no neighbors

Usage

```
has_isolates(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

A boolean value indicates if weights matrix is symmetric

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
has_isolates(queen_w)

## End(Not run)
```

`hinge15_breaks` *(Box) Hinge15 Breaks*

Description

Hinge15 breaks data into 6 groups like box plot groups (Lower outlier, < 25

Usage

```
hinge15_breaks(df)
```

Arguments

`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
hinge15_breaks(guerry['Crm_prs'])
```

`hinge30_breaks` *(Box) Hinge30 Breaks*

Description

Hinge30 breaks data into 6 groups like box plot groups (Lower outlier, < 25

Usage

```
hinge30_breaks(df)
```

Arguments

`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
hinge30_breaks(guerry['Crm_prs'])
```

`is_symmetric` *Symmetry of Weights Matrix*

Description

Check if weights matrix is symmetric

Usage

```
is_symmetric(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

A boolean value indicates if weights matrix is symmetric

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
is_symmetric(queen_w)

## End(Not run)
```

`join_count_ratio` *Join Count Ratio*

Description

Join count ratio is the join counts, the number of times a category is surrounded by neighbors of the same category, over the total number of neighbors after converting each category to a dummy variable.

Usage

```
join_count_ratio(clusters, w)
```

Arguments

`clusters`	A cluster classification variable (categorical values from a dataframe or values returned from cluster functions)
`w`	An instance of Weight class

Value

A data.frame with names "Cluster", "N", "Neighbors", "Join Count", "Ratio"

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
clusters <- skater(5, queen_w, data)
results <- join_count_ratio(clusters, queen_w)
results

## End(Not run)
```

`kernel_knn_weights` *K-NN Kernel Spatial Weights*

Description

Create a kernel weights by specifying k-nearest neighbors and a kernel method

Usage

```
kernel_knn_weights(
  sf_obj,
  k,
  kernel_method,
  adaptive_bandwidth = TRUE,
  use_kernel_diagonals = FALSE,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`k`	a positive integer number for k-nearest neighbors
`kernel_method`	a string value, which has to be one of 'triangular', 'uniform', 'epanechnikov', 'quartic', 'gaussian'
`adaptive_bandwidth`	(optional) TRUE (default) or FALSE: TRUE use adaptive bandwidth calculated using distance of k-nearest neithbors, FALSE use max distance of all observation to their k-nearest neighbors
`use_kernel_diagonals`	(optional) FALSE (default) or TRUE, apply kernel on the diagonal of weights matrix
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
adptkernel_w = kernel_knn_weights(guerry, 6, "uniform")
summary(adptkernel_w)
```

`kernel_weights` *Distance-based Kernel Spatial Weights*

Description

Create a kernel weights by specifying a bandwidth and a kernel method

Usage

```
kernel_weights(
  sf_obj,
  bandwidth,
  kernel_method,
  use_kernel_diagonals = FALSE,
```

```
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`bandwidth`	A positive numeric value of bandwidth
`kernel_method`	a string value, which has to be one of 'triangular', 'uniform', 'epanechnikov', 'quartic', 'gaussian'
`use_kernel_diagonals`	(optional) FALSE (default) or TRUE, apply kernel on the diagonal of weights matrix
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
bandwidth <- min_distthreshold(guerry)
kernel_w <- kernel_weights(guerry, bandwidth, kernel_method = "uniform")
summary(kernel_w)
```

`knn_weights` *K-Nearest Neighbors-based Spatial Weights*

Description

Create a k-nearest neighbors based spatial weights

Usage

```
knn_weights(
  sf_obj,
  k,
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`k`	a positive integer number for k-nearest neighbors
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
knn6_w <- knn_weights(guerry, 6)
summary(knn6_w)
```

`LISA-class` *LISA class (Internally Used)*

Description

A LISA-class that wrappers the statistics of LISA computation

Fields

`gda_lisa` An object of GeoDaLISA

`p_vals` The pseudo-p values of significance of LISA computation

`c_vals` The cluster indicators of LISA computation

`lisa_vals` The local spatial autocorrelation values of LISA computation

`nn_vals` The number of neighbors of every observations in LISA computation

`labels` The cluster labels of LISA

`colors` The cluster colors (HEX format) of LISA

Methods

`GetBO(current_p)` Get the Bonferroni bound value

`GetClusterIndicators()` Get the local cluster indicators returned from LISA computation.

`GetColors()` Get the cluster colors of LISA computation.

`GetFDR(current_p)` Get the False Discovery Rate value

`GetLISAValues()` Get the local spatial autocorrelation values returned from LISA computation.

`GetLabels()` Get the cluster labels of LISA computation.

`GetLocalSignificanceValues()` Get the local pseudo-p values of significance returned from LISA computation.

`GetNumNeighbors()` Get the number of neighbors of every observations in LISA computation.

`Run()` Call to run LISA computation

`SetPermutations(num_perm)` Set the number of permutations for the LISA computation

`SetSignificanceCutoff(cutoff)` Set the cutoff value of significance values

`SetThreads(num_threads)` Set the number of CPU threads for the LISA computation

`initialize(lisa_obj)` Constructor with a LISA object (internally used)

`lisa_bo` *Bonferroni bound value of local spatial autocorrelation*

Description

Get Bonferroni bound value based on current LISA computation and current significat p-value

Usage

```
lisa_bo(gda_lisa, current_p)
```

Arguments

`gda_lisa`	An instance of LISA object
`current_p`	A value of current siginificant p-value

Value

A numeric value of Bonferroni bound

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
bo <- lisa_bo(lisa, 0.05)
bo

## End(Not run)
```

`lisa_clusters` *Get local cluster indicators*

Description

Get the local cluster indicators returned from LISA computation.

Usage

```
lisa_clusters(gda_lisa, cutoff = 0)
```

Arguments

`gda_lisa`	An instance of LISA object
`cutoff`	A value of cutoff for significance p-values to filter not-significant clusters, default=0.0, means not used

Value

A numeric vector of LISA cluster indicator

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
clsts <- lisa_clusters(lisa)
clsts

## End(Not run)
```

`lisa_colors` *Get cluster colors*

Description

Get the cluster colors of LISA computation.

Usage

```
lisa_colors(gda_lisa)
```

Arguments

`gda_lisa`	An instance of LISA object

Value

A string vector of cluster colors

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
clrs <- lisa_colors(lisa)
clrs

## End(Not run)
```

`lisa_fdr` *False Discovery Rate value of local spatial autocorrelation*

Description

Get False Discovery Rate value based on current LISA computation and current significant p-value

Usage

```
lisa_fdr(gda_lisa, current_p)
```

Arguments

`gda_lisa`	An instance of LISA object
`current_p`	A value of current siginificant p-value

Value

A numeric vector of False Discovery Rate

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
fdr <- lisa_fdr(lisa, 0.05)
fdr

## End(Not run)
```

`lisa_labels` *Get cluster labels*

Description

Get cluster labels of LISA computation.

Usage

```
lisa_labels(gda_lisa)
```

Arguments

`gda_lisa`	An instance of LISA object

Value

A string vector of cluster labels

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
lbls <- lisa_labels(lisa)
lbls

## End(Not run)
```

`lisa_num_nbrs` *Get numbers of neighbors for all observations*

Description

Get numbers of neighbors for all observations

Usage

```
lisa_num_nbrs(gda_lisa)
```

Arguments

`gda_lisa`	An instance of LISA object

Value

A numeric vector of the number of neighbors

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
nn <- lisa_num_nbrs(lisa)
nn

## End(Not run)
```

`lisa_pvalues` *Get pseudo-p values of LISA*

Description

Get the local pseudo-p values of significance returned from LISA computation.

Usage

```
lisa_pvalues(gda_lisa)
```

Arguments

`gda_lisa`	An instance of LISA object

Value

A numeric vector of pseudo-p values of local spatial autocorrelation

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
pvals <- lisa_pvalues(lisa)
pvals

## End(Not run)
```

`lisa_values` *Get LISA values*

Description

Get the local spatial autocorrelation values returned from LISA computation

Usage

```
lisa_values(gda_lisa)
```

Arguments

`gda_lisa`	An instance of LISA object

Value

A numeric vector of local spatial autocorrelation

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
lms <- lisa_values(lisa)
lms

## End(Not run)
```

`local_bijoincount` *Bivariate Local Join Count Statistics*

Description

The function to apply local Bivariate Join Count statistics

Usage

```
local_bijoincount(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with two selected variable. E.g. guerry[c("TopCrm", "InvCrm")]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
guerry["InvCrm"] <-  1 - guerry[["TopCrm"]]
lisa <- local_bijoincount(queen_w, guerry[c("TopCrm", "InvCrm")])
clsts<- lisa_clusters(lisa)
clsts
```

`local_bimoran` *Bivariate Local Moran Statistics*

Description

The function to apply bivariate local Moran statistics

Usage

```
local_bimoran(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with two selected variable. E.g. guerry[c('Crm_prs','Litercy')]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_bimoran(queen_w, guerry[c('Crm_prs','Litercy')])
lms <- lisa_values(lisa)
lms
```

`local_g` *Local Getis-Ord's G Statistics*

Description

The function to apply Getis-Ord's local G statistics

Usage

```
local_g(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_g(queen_w, guerry["Crm_prs"])
lms <- lisa_values(lisa)
lms
```

`local_geary` *Local Geary Statistics*

Description

The function to apply local Geary statistics

Usage

```
local_geary(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_geary(queen_w, guerry["Crm_prs"])
lms <- lisa_values(lisa)
lms
```

`local_gstar` *Local Getis-Ord's G* Statistics*

Description

The function to apply Getis-Ord's local G* statistics

Usage

```
local_gstar(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_gstar(queen_w,  guerry["Crm_prs"])
lms <- lisa_values(lisa)
lms
```

`local_joincount` *Local Join Count Statistics*

Description

The function to apply local Join Count statistics

Usage

```
local_joincount(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_joincount(queen_w, guerry['TopCrm'])
clsts<- lisa_clusters(lisa)
clsts
```

`local_moran` *Local Moran Statistics*

Description

The function to apply local Moran statistics

Usage

```
local_moran(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with only selected variable. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_moran(queen_w, guerry["Crm_prs"])
lms <- lisa_values(lisa)
lms
```

`local_moran_eb` *Local Moran with Empirical Bayes(EB) Rate*

Description

The function to apply local Moran with EB Rate statistics. The EB rate is first computed from "event" and "base" variables, and then used in local moran statistics.

Usage

```
local_moran_eb(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with two selected variable: one is "event", anothor is "base" variable. E.g. guerry[c("hr60", "po60")]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
## Not run:
library(sf)
nat <- st_read("natregimes.shp")
nat_w <- queen_weights(nat)
lisa <- local_moran_eb(queen_w, guerry[c("hr60", "po60")])
```

```
lms <- lisa_values(lisa)
lms

## End(Not run)
```

`local_multigeary` *Local Multivariate Geary Statistics*

Description

The function to apply local Multivariate Geary statistics

Usage

```
local_multigeary(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variables only. E.g. guerry["Crm_prs"]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants',
'Suicids')]
lisa <- local_multigeary(queen_w, data)
lms <- lisa_clusters(lisa)
lms
```

`local_multijoincount` *(Multivariate) Colocation Local Join Count Statistics*

Description

The function to apply (multivariate) colocation local Join Count statistics

Usage

```
local_multijoincount(
  w,
  df,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variables only. E.g. guerry[c("TopCrm", "TopWealth", "TopLit")]
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_multijoincount(queen_w,
guerry[c('TopWealth','TopWealth', 'TopLit')])
clsts <- lisa_clusters(lisa)
clsts
```

`local_multiquantilelisa` *Multivariate Quantile LISA Statistics*

Description

The function to apply multivariate quantile LISA statistics

Usage

```
local_multiquantilelisa(
  w,
  df,
  k,
  q,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variables only. E.g. guerry[c("TopCrm", "TopWealth", "TopLit")]
`k`	A vector of "k" values indicate the number of quantiles for each variable. Value range e.g. [1, 10]
`q`	A vector of "q" values indicate which quantile or interval for each variable used in local join count statistics. Value stars from 1.
`permutations`	(optional) The number of permutations for the LISA computation

`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_multiquantilelisa(queen_w, guerry[c("Crm_prp", "Litercy")],
k=c(4,4), q=c(1,1))
clsts <- lisa_clusters(lisa)
clsts
```

`local_quantilelisa` *Quantile LISA Statistics*

Description

The function to apply quantile LISA statistics

Usage

```
local_quantilelisa(
  w,
  df,
  k,
  q,
  permutations = 999,
  permutation_method = "complete",
  significance_cutoff = 0.05,
  cpu_threads = 6,
  seed = 123456789
)
```

Arguments

`w`	An instance of Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]
`k`	A value indicates the number of quantiles. Value range e.g. [1, 10]
`q`	A value indicates which quantile or interval used in local join count statistics. Value stars from 1.
`permutations`	(optional) The number of permutations for the LISA computation
`permutation_method`	(optional) The permutation method used for the LISA computation. Options are ('complete', 'lookup'). Default is 'complete'.
`significance_cutoff`	(optional) A cutoff value for significance p-values to filter not-significant clusters
`cpu_threads`	(optional) The number of cpu threads used for parallel LISA computation
`seed`	(optional) The seed for random number generator

Value

An instance of LISA-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
lisa <- local_quantilelisa(queen_w, guerry["Crm_prs"], k=4, q=1)
clsts <- lisa_clusters(lisa)
clsts
```

`make_spatial` *Make Spatial*

Description

Make spatially constrained clusters from spatially non-constrained clusters using the contiguity information from the input weights

Usage

```
make_spatial(clusters, w)
```

Arguments

`clusters`	A cluster classification variable (categorical values from a dataframe or values returned from cluster functions)
`w`	An instance of Weight class

Value

A vector of categorical values (cluster classification)

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
clusters <- kmeans(5, data)
queen_w <- queen_weights(guerry)
results <- make_spatial(clusters, queen_w)
results

## End(Not run)
```

`maxp_greedy` *A greedy algorithm to solve the max-p-region problem*

Description

The max-p-region problem is a special case of constrained clustering where a finite number of geographical areas are aggregated into the maximum number of regions (max-p-regions), such that each region is geographically connected and the clusters could maximize internal homogeneity.

Usage

```
maxp_greedy(
  w,
  df,
  bound_variable,
  min_bound,
  iterations = 99,
  initial_regions = vector("numeric"),
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  cpu_threads = 6,
  rdist = numeric()
)
```

Arguments

`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]

`bound_variable`	A numeric vector of selected bounding variable
`min_bound`	A minimum value that the sum value of bounding variable int each cluster should be greater than
`iterations`	(optional): The number of iterations of greedy algorithm. Defaults to 99.
`initial_regions`	(optional): The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`cpu_threads`	(optional) The number of cpu threads used for parallel computation
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
bound_variable <- guerry['Pop1831']
min_bound <- 3236.67 # 10% of Pop1831
maxp_clusters <- maxp_greedy(queen_w, data, bound_variable, min_bound, iterations=99)
maxp_clusters

## End(Not run)
```

`maxp_sa` *A simulated annealing algorithm to solve the max-p-region problem*

Description

The max-p-region problem is a special case of constrained clustering where a finite number of geographical areas are aggregated into the maximum number of regions (max-p-regions), such that each region is geographically connected and the clusters could maximize internal homogeneity.

Usage

```
maxp_sa(
  w,
  df,
  bound_variable,
  min_bound,
  cooling_rate,
  sa_maxit = 1,
  iterations = 99,
  initial_regions = vector("numeric"),
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  cpu_threads = 6,
  rdist = numeric()
)
```

Arguments

`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`bound_variable`	A numeric vector of selected bounding variable
`min_bound`	A minimum value that the sum value of bounding variable int each cluster should be greater than
`cooling_rate`	The cooling rate of a simulated annealing algorithm. Defaults to 0.85
`sa_maxit`	(optional): The number of iterations of simulated annealing. Defaults to 1
`iterations`	(optional): The number of iterations of SA algorithm. Defaults to 99.
`initial_regions`	(optional): The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`cpu_threads`	(optional) The number of cpu threads used for parallel computation
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
bound_variable <- guerry['Pop1831']
min_bound <- 3236.67 # 10% of Pop1831
maxp_clusters <- maxp_sa(queen_w, data, bound_variable, min_bound, cooling_rate=0.85, sa_maxit=1)
maxp_clusters

## End(Not run)
```

`maxp_tabu` *A tabu-search algorithm to solve the max-p-region problem*

Description

The max-p-region problem is a special case of constrained clustering where a finite number of geographical areas are aggregated into the maximum number of regions (max-p-regions), such that each region is geographically connected and the clusters could maximize internal homogeneity.

Usage

```
maxp_tabu(
  w,
  df,
  bound_variable,
  min_bound,
  tabu_length = 10,
  conv_tabu = 10,
  iterations = 99,
  initial_regions = vector("numeric"),
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  cpu_threads = 6,
  rdist = numeric()
)
```

Arguments

`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`bound_variable`	A numeric vector of selected bounding variable

`min_bound`	A minimum value that the sum value of bounding variable int each cluster should be greater than
`tabu_length`	(optional): The length of a tabu search heuristic of tabu algorithm. Defaults to 10.
`conv_tabu`	(optional): The number of non-improving moves. Defaults to 10.
`iterations`	(optional): The number of iterations of Tabu algorithm. Defaults to 99.
`initial_regions`	(optional): The initial regions that the local search starts with. Default is empty. means the local search starts with a random process to "grow" clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(optional) The seed for random number generator. Defaults to 123456789.
`cpu_threads`	(optional) The number of cpu threads used for parallel computation
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
bound_variable <- guerry['Pop1831']
min_bound <- 3236.67 # 10% of Pop1831
maxp_clusters <- maxp_tabu(queen_w, data, bound_variable, min_bound, tabu_length=10, conv_tabu=10)
maxp_clusters

## End(Not run)
```

`max_neighbors` *Maximum Neighbors of Spatial Weights*

Description

Get the number of maximum neighbors of spatial weights

Usage

```
max_neighbors(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

The number of maximum neighbors of spatial weights

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
max_neighbors(queen_w)

## End(Not run)
```

`mean_neighbors` *Mean Neighbors of Spatial Weights*

Description

Get the number of mean neighbors of spatial weights

Usage

```
mean_neighbors(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

The number of mean neighbors of spatial weights

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
mean_neighbors(queen_w)

## End(Not run)
```

`median_neighbors` *Median Neighbors of Spatial Weights*

Description

Get the number of median neighbors of spatial weights

Usage

```
median_neighbors(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

The number of median neighbors of spatial weights

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
median_neighbors(queen_w)

## End(Not run)
```

`min_distthreshold` *Minimum Distance Threshold for Distance-based Weights*

Description

Get minimum threshold of distance that makes sure each observation has at least one neighbor

Usage

```
min_distthreshold(sf_obj, is_arc = FALSE, is_mile = TRUE)
```

Arguments

`sf_obj`	An sf (simple feature) object
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations
`is_mile`	(optional) TRUE (default) or FALSE, if 'is_arc' option is TRUE, then 'is_mile' will set distance unit to 'mile' or 'km'.

Value

A numeric value of minimum threshold of distance

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
dist_thres <- min_distthreshold(guerry)
dist_thres

## End(Not run)
```

`min_neighbors` *Minimum Neighbors of Spatial Weights*

Description

Get the number of minimum neighbors of spatial weights

Usage

```
min_neighbors(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

The number of minimum neighbors of spatial weights

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
min_neighbors(queen_w)

## End(Not run)
```

`natural_breaks` *Natural Breaks (Jenks)*

Description

Natural Breaks group data whose boundaries are set where there are relatively big differences.

Usage

```
natural_breaks(k, df)
```

Arguments

`k`	A numeric value indicates how many breaks
`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
natural_breaks(k=5, guerry['Crm_prs'])
```

`neighbor_match_test` *Local Neighbor Match Test*

Description

The local neighbor match test is to assess the extent of overlap between k-nearest neighbors in geographical space and k-nearest neighbors in multi-attribute space.

Usage

```
neighbor_match_test(
  df,
  k,
  scale_method = "standardize",
  distance_method = "euclidean",
  power = 1,
  is_inverse = FALSE,
  is_arc = FALSE,
  is_mile = TRUE
)
```

Arguments

`df`	A subset of sf object with selected variables. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`k`	a positive integer number for k-nearest neighbors searching.
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The type of distance metrics used to measure the distance between input data. Options are ('euclidean', 'manhattan'). Default is 'euclidean'.
`power`	(optional) The power (or exponent) of a number says how many times to use the number in a multiplication.
`is_inverse`	(optional) FALSE (default) or TRUE, apply inverse on distance value.
`is_arc`	(optional) FALSE (default) or TRUE, compute arc distance between two observations.
`is_mile`	(optional) TRUE (default) or FALSE, convert distance unit from mile to km.

Value

A data.frame with two columns "Cardinality" and "Probability".

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
nbr_test <- neighbor_match_test(data, 6)
nbr_test
```

`percentile_breaks` *Percentile Breaks*

Description

Percentile breaks data into 6 groups: the lowest 1 10-50

Usage

```
percentile_breaks(df)
```

Arguments

`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
percentile_breaks(guerry['Crm_prs'])
```

`p_GeoDa-class` *p_GeoDa*

Description

p_GeoDa class is a RefClass that wraps the C++ 'GeoDa' class. See C++ functions in rcpp_rgeoda.cpp

---

`p_GeoDaTable-class` *p_GeoDaTable*

Description

p_GeoDaTable class is a RefClass that wraps the C++ 'GeoDaTable' class. See C++ functions in rcpp_rgeoda.cpp

---

`p_GeoDaWeight-class` *p_GeoDaWeight*

Description

p_GeoDaWeight class is a RefClass that wraps the C++ GeoDaWeight class. See C++ functions in rcpp_weights.cpp

---

`p_LISA-class` *p_LISA*

Description

p_LISA class is a RefClass that wraps the C++ LISA class. See C++ functions in rcpp_lisa.cpp

---

`quantile_breaks` *Quantile Breaks*

Description

Quantile breaks data into groups that each have the same number of observations

Usage

```
quantile_breaks(k, df)
```

Arguments

`k`	A numeric value indicates how many breaks
`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
quantile_breaks(k=5, guerry['Crm_prs'])
```

`queen_weights` *Queen Contiguity Spatial Weights*

Description

Create a Queen contiguity weights with options of "order", "include lower order" and "precision threshold"

Usage

```
queen_weights(
  sf_obj,
  order = 1,
  include_lower_order = FALSE,
  precision_threshold = 0
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`order`	(Optional) Order of contiguity
`include_lower_order`	(Optional) Whether or not the lower order neighbors should be included in the weights structure
`precision_threshold`	(Optional) The precision of the underlying shape file is insufficient to allow for an exact match of coordinates to determine which polygons are neighbors

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
summary(queen_w)
```

`read_gal` *Read a .GAL file*

Description

Create a spatial weights object from a .GAL file

Usage

```
read_gal(file_path, id_vec = c())
```

Arguments

`file_path`	The file paht of the .GAL file
`id_vec`	The id_vec is the id values used in the .GAL file. Default is empty.

Value

A weights object

`read_gwt` *Read a .GWT file*

Description

Create a spatial weights object from a .GWT file

Usage

```
read_gwt(file_path, id_vec = c())
```

Arguments

`file_path`	The file paht of the .GWT file
`id_vec`	The id_vec is the id values used in the .GWT file. Default is empty.

Value

A weights object

---

read_swm *Read a .SWM file*

Description

Create a spatial weights object from a .SWM file

Usage

```
read_swm(file_path, id_vec = numeric())
```

Arguments

`file_path`	The file paht of the .SWM file
`id_vec`	The id_vec is the id values used in the .SWM file. e.g. c(0,1,2,3,...)

Value

A weights object

---

redcap *Regionalization with dynamically constrained agglomerative clustering and partitioning*

Description

REDCAP (Regionalization with dynamically constrained agglomerative clustering and partitioning) is developed by D. Guo (2008). Like SKATER, REDCAP starts from building a spanning tree with 4 different ways (single-linkage, average-linkage, ward-linkage and the complete-linkage). The single-linkage way leads to build a minimum spanning tree. Then,REDCAP provides 2 different ways (first-order and full-order constraining) to prune the tree to find clusters. The first-order approach with a minimum spanning tree is exactly the same with SKATER. In GeoDa and pygeoda, the following methods are provided: * First-order and Single-linkage * Full-order and Complete-linkage * Full-order and Average-linkage * Full-order and Single-linkage * Full-order and Ward-linkage

Usage

```
redcap(
  k,
  w,
  df,
  method = "fullorder-averagelinkage",
  bound_variable = data.frame(),
  min_bound = 0,
```

```
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  cpu_threads = 6,
  rdist = numeric()
)
```

Arguments

`k`	The number of clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`method`	"firstorder-singlelinkage", "fullorder-completelinkage", "fullorder-averagelinkage","fullorder-singlelinkage", "fullorder-wardlinkage"
`bound_variable`	(optional) A data frame with selected bound variabl
`min_bound`	(optional) A minimum bound value that applies to all clusters
`scale_method`	(optional) One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(int,optional) The seed for random number generator. Defaults to 123456789.
`cpu_threads`	(optional) The number of cpu threads used for parallel computation
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
guerry_clusters <- redcap(4, queen_w, data, "fullorder-completelinkage")
guerry_clusters

## End(Not run)
```

`rook_weights` *Rook Contiguity Spatial Weights*

Description

Create a Rook contiguity weights with options of "order", "include lower order" and "precision threshold"

Usage

```
rook_weights(
  sf_obj,
  order = 1,
  include_lower_order = FALSE,
  precision_threshold = 0
)
```

Arguments

`sf_obj`	An sf (simple feature) object
`order`	(Optional) Order of contiguity
`include_lower_order`	(Optional) Whether or not the lower order neighbors should be included in the weights structure
`precision_threshold`	(Optional) The precision of the underlying shape file is insufficient to allow for an exact match of coordinates to determine which polygons are neighbors

Value

An instance of Weight-class

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
rook_w <- rook_weights(guerry)
summary(rook_w)
```

`save_weights` *Save Spatial Weights*

Description

Save spatial weights to a file

Usage

```
save_weights(gda_w, id_variable, out_path, layer_name = "")
```

Arguments

`gda_w`	A Weight object
`id_variable`	The id variable (a data.frame) that defines the unique value of each observation when saving a weights file
`out_path`	The path of an output weights file
`layer_name`	(optional) The name of the layer of input dataset

Value

A boolean value indicates if save successfully or failed

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
save_weights(quen_w, guerry_df['CODE_DE'], out_path = '/path/Guerry_r.gal')

## End(Not run)
```

`schc` *Spatially Constrained Hierarchical Clucstering (SCHC)*

Description

Spatially constrained hierarchical clustering is a special form of constrained clustering, where the constraint is based on contiguity (common borders). The method builds up the clusters using agglomerative hierarchical clustering methods: single linkage, complete linkage, average linkage and Ward's method (a special form of centroid linkage). Meanwhile, it also maintains the spatial contiguity when merging two clusters.

Usage

```
schc(
  k,
  w,
  df,
  method = "average",
  bound_variable = data.frame(),
  min_bound = 0,
  scale_method = "standardize",
  distance_method = "euclidean",
  rdist = numeric()
)
```

Arguments

`k`	The number of clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`method`	"single", "complete", "average","ward"
`bound_variable`	(optional) A data frame with selected bound variabl
`min_bound`	(optional) A minimum bound value that applies to all clusters
`scale_method`	One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
guerry_clusters <- schc(4, queen_w, data, "complete")
guerry_clusters
```

`set_neighbors` *Set neighbors of an observation*

Description

Set neighbors for idx-th observation, idx starts from 1

Usage

```
set_neighbors(gda_w, idx, nbrs)
```

Arguments

`gda_w`	A Weight object
`idx`	A value indicates idx-th observation, idx start from 1
`nbrs`	A list indicates the neighbors of idx-th observation (id start from 1)

Examples

```
## Not run:
new_w <- create_weights(10)
set_neighbors(new_w, 1, c(2,3))
update_weights(new_w)

## End(Not run)
```

`set_neighbors_with_weights` *Set neighbors and weights values of an observation*

Description

Set neighbors and the associated weights values for idx-th observation, idx starts from 1

Usage

```
set_neighbors_with_weights(gda_w, idx, nbrs, wvals)
```

Arguments

`gda_w`	A Weight object
`idx`	A value indicates idx-th observation, idx start from 1
`nbrs`	A list indicates the neighbors of idx-th observation (id start from 1)
`wvals`	A list indicates the associated weights values of the neighbors

Examples

```
## Not run:
new_w <- create_weights(10)
set_neighbors(new_w, 1, c(2,3))
update_weights(new_w)

## End(Not run)
```

`sf_to_geoda` *Create an instance of geoda-class from a ’sf’ object*

Description

Create an instance of geoda-class from a ’sf’ object returned from ’st_read()’ function. NOTE: The table content is NOT used to create an instance of geoda-class.

Usage

```
sf_to_geoda(sf_obj, with_table = TRUE)
```

Arguments

`sf_obj`	An instance of ’sf’ object
`with_table`	A boolean flag indicates if table is copied from sf object to create geoda object. Default is TRUE.

Value

An instance of geoda-class

`skater` *Spatial C(K)luster Analysis by Tree Edge Removal*

Description

SKATER forms clusters by spatially partitioning data that has similar values for features of interest.

Usage

```
skater(
  k,
  w,
  df,
  bound_variable = data.frame(),
  min_bound = 0,
  scale_method = "standardize",
  distance_method = "euclidean",
  random_seed = 123456789,
  cpu_threads = 6,
  rdist = numeric()
)
```

Arguments

`k`	The number of clusters
`w`	An instance of Weight class
`df`	A data frame with selected variables only. E.g. guerry[c("Crm_prs", "Crm_prp", "Litercy")]
`bound_variable`	(optional) A data frame with selected bound variable
`min_bound`	(optional) A minimum bound value that applies to all clusters
`scale_method`	One of the scaling methods ('raw', 'standardize', 'demean', 'mad', 'range_standardize', 'range_adjust') to apply on input data. Default is 'standardize' (Z-score normalization).
`distance_method`	(optional) The distance method used to compute the distance betwen observation i and j. Defaults to "euclidean". Options are "euclidean" and "manhattan"
`random_seed`	(int,optional) The seed for random number generator. Defaults to 123456789.
`cpu_threads`	(optional) The number of cpu threads used for parallel computation
`rdist`	(optional) The distance matrix (lower triangular matrix, column wise storage)

Value

A names list with names "Clusters", "Total sum of squares", "Within-cluster sum of squares", "Total within-cluster sum of squares", and "The ratio of between to total sum of squares".

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
guerry_clusters <- skater(4, queen_w, data)
guerry_clusters
```

`spatial_lag` *Spatial Lag*

Description

Compute the spatial lag for idx-th observation using selected variable and current weights matrix

Usage

```
spatial_lag(gda_w, df)
```

Arguments

`gda_w`	A Weight object
`df`	A data frame with selected variable only. E.g. guerry["Crm_prs"]

Value

A data.frame with one column "Spatial Lag"

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
crm_lag <- spatial_lag(queen_w, guerry["Crm_prs"])
crm_lag

## End(Not run)
```

`spatial_validation` *Spatial Validation*

Description

Spatial validation provides a collection of validation measures including 1. fragmentations (entropy, simpson), 2. join count ratio, 3. compactness (isoperimeter quotient) and 4. diameter.

Usage

```
spatial_validation(sf_obj, clusters, w)
```

Arguments

`sf_obj`	An sf (simple feature) object
`clusters`	A cluster classification variable (categorical values from a dataframe or values returned from cluster functions)
`w`	An instance of Weight class

Value

A list with names "Is Spatially Constrained", "Fragmentation", "Join Count Ratio", "Compactness", and "Diameter".

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
data <- guerry[c('Crm_prs','Crm_prp','Litercy','Donatns','Infants','Suicids')]
clusters <- skater(5, queen_w, data)
results <- spatial_validation(guerry, clusters, queen_w)
results

## End(Not run)
```

`sp_to_geoda` *Create an instance of geoda-class from a 'sp' object*

Description

Create an instance of geoda-class from a 'sp' object. NOTE: The table content is NOT used to create an instance of geoda-class.

Usage

```
sp_to_geoda(sp_obj, with_table = TRUE)
```

Arguments

`sp_obj`	An instance of 'sp' object
`with_table`	A boolean flag indicates if table is copied from sf object to create geoda object. Default is TRUE

Value

An instance of geoda-class

`stddev_breaks` *Standard Deviation Breaks*

Description

Standard deviation breaks first transforms data to standard deviation units (mean=0, stddev=1), and then divide the range of values into 6 groups.

Usage

```
stddev_breaks(df)
```

Arguments

`df`	A data frame with selected variable. E.g. guerry["Crm_prs"]

Value

A vector of numeric values of computed breaks

Examples

```
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
stddev_breaks(guerry['Crm_prs'])
```

`summary.Weight` *Summary of Spatial Weights*

Description

Override the summary() function for spatial weights

Usage

```
## S3 method for class 'Weight'
summary(object, ...)
```

Arguments

`object`	A Weight object
`...`	summary optional parameters

Value

A summary description of an instance of Weight-class

Examples

```
## Not run:
library(sf)
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
summary(queen_w)

## End(Not run)
```

`update_weights` *Update meta data of a spatial weights*

Description

Update meta data of a spatial weights. This function can be used after calling ‘set_neighbor()‘ function .

Usage

```
update_weights(gda_w)
```

Arguments

`gda_w`	A Weight object

Examples

```
## Not run:
new_w <- create_weights(10)
set_neighbors(new_w, 1, c(2,3))
update_weights(new_w)

## End(Not run)
```

`Weight-class` *Weight class (Internally Used)*

Description

A wrapper class for p_GeoDaWeight class

Fields

`gda_w` An object of p_GeoDaWeight-class

`is_symmetric` If weights matrix is symmetric

`sparsity` Sparsity of weights matrix

`min_neighbors` Minimum number of neighbors

`max_neighbors` Maximum number of neighbors

`num_obs` Number of observations

`mean_neighbors` Mean number of neighbors

`median_neighbors` Median number of neighbors

`has_isolates` If the weights matrix has any isolates

Methods

`GetNeighborWeights(idx)` Get weights values of neighbors for idx-th observation, idx starts from 0

`GetNeighbors(idx)` Get neighbors for idx-th observation, idx starts from 0

`GetPointer()` Get the C++ object pointer (internally used)

`GetSparsity()` Get sparsity computed from weights matrix

`HasIsolates()` Check if weights matrix has isolates, or if any observation has no neighbors

`IsSymmetric()` Check if weights matrix is symmetric

`SaveToFile(out_path, layer_name, id_name, id_values)` Save current spatial weights to a file.

out_path: The path of an output weights file
layer_name : The name of the layer of input dataset
id_name : The id name (or field name), which is an associated column contains unique values, that makes sure that the weights are connected to the correct observations in the data table.
id_values : The tuple of values of selected id_name (column/field)

`SetNeighbors(idx, nbrs)` Set neighbors for one observation

`SetNeighborsAndWeights(idx, nbrs, nbr_w)` Set neighbors with weights values for one observation

`SpatialLag(values)` Compute spatial lag values for values of selected variable

`Update(updateStats = TRUE)` Update the weights meta data

`initialize(o_gda_w)` Constructor with a GeoDaWeight object (internally used)

`weights_sparsity` *Sparsity of Spatial Weights*

Description

Get sparsity (

Usage

```
weights_sparsity(gda_w)
```

Arguments

`gda_w`	A Weight object

Value

A numeric value of spatial weights sparsity

Examples

```
## Not run:
guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
guerry <- st_read(guerry_path)
queen_w <- queen_weights(guerry)
weights_sparsity(queen_w)

## End(Not run)
```

Index