Package ‘GeoThinneR’

January 20, 2025

Type Package

Title Simple Spatial Thinning for Ecological and Spatial Analysis

Version 1.1.0

Description Provides efficient geospatial thinning algorithms to reduce the density of coordinate data while maintaining spatial relationships. Implements K-D Tree and brute-force distance-based thinning, as well as grid-based and precision-based thinning methods. For more information on the methods, see Elseberg et al. (2012) <https://hdl.handle.net/10446/86202>.

License MIT + file LICENSE

URL https://github.com/jmestret/GeoThinneR,
https://jmestret.github.io/GeoThinneR/

BugReports https://github.com/jmestret/GeoThinneR/issues

Depends R (>= 4.0.0)

Imports data.table, fields, matrixStats, nabor, Rcpp, stats, terra

Suggests ggplot2, knitr, rmarkdown, sf, testthat (>= 3.0.0), tibble

LinkingTo Rcpp

VignetteBuilder knitr

BuildVignettes true

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

NeedsCompilation yes

Author Jorge Mestre-Tomás [aut, cre] (<https://orcid.org/0000-0002-8983-3417>)

Maintainer Jorge Mestre-Tomás <jorge.mestre.tomas@csic.es>

Repository CRAN

Date/Publication 2024-10-03 23:10:13 UTC

Contents

`assign_coords_to_grid` *Assign Geographic Coordinates to Grid Cells*

Description

This function assigns a set of geographic coordinates (longitude and latitude) to grid cells based on a specified cell size.

Usage

```
assign_coords_to_grid(coords, cell_size)
```

Arguments

`coords`	A data frame or matrix with two columns: longitude and latitude.
`cell_size`	Numeric value representing the size of each grid cell, typically in degrees.

Value

A character vector of grid cell identifiers, where each identifier is formatted as "x_y", representing the grid cell coordinates.

Examples

```
coords <- data.frame(long = c(-122.4194, 0), lat = c(37.7749, 0))
cell_size <- 1
assign_coords_to_grid(coords, cell_size)
```

brute_force_thinning *Perform Brute Force Thinning*

Description

This function applies a brute force algorithm to thin a set of spatial coordinates, attempting to maximize the number of points retained while ensuring a minimum distance (‘thin_dist‘) between any two points.

Usage

```
brute_force_thinning(
  coordinates,
  thin_dist = 10,
  trials = 10,
  all_trials = FALSE,
  target_points = NULL,
  euclidean = FALSE,
  R = 6371
)
```

Arguments

`coordinates`	A numeric matrix or data frame with two columns representing longitude and latitude (or XY coordinates if ‘euclidean = TRUE‘).
`thin_dist`	Numeric value representing the thinning distance in kilometers (default: 10 km).
`trials`	Integer specifying the number of trials to run for thinning (default: 10).
`all_trials`	Logical value indicating whether to return the results of all trials (‘TRUE‘) or just the best attempt with the most points retained (‘FALSE‘, default).
`target_points`	Optional integer specifying the number of points to retain. If ‘NULL‘ (default), the function tries to maximize the number of points retained.
`euclidean`	Logical value indicating whether to compute the Euclidean distance (‘TRUE‘) or Haversine distance (‘FALSE‘, default).
`R`	Numeric value representing the Earth’s radius in kilometers (default: 6371 km). Only used if ‘euclidean = FALSE‘.

Value

A logical vector indicating which points are kept in the best trial if ‘all_trials = FALSE‘; otherwise, a list of logical vectors for each trial.

Examples

```
# Example with geographic coordinates (Haversine distance)
coords <- data.frame(
  long = c(-122.4194, -122.4195, -122.4196),
  lat = c(37.7749, 37.7740, 37.7741)
)
coords <- as.matrix(coords)

result <- brute_force_thinning(coords, thin_dist = 0.1, trials = 5)
print(result)

# Example computing Euclidean distance
result_euclidean <- brute_force_thinning(coords, thin_dist = 1, trials = 5, euclidean = TRUE)
print(result_euclidean)
```

`caretta` *Loggerhead Sea Turtle (*Caretta caretta*) Occurrences in the Mediterranean Sea*

Description

This dataset contains a subset of global occurrences of the Loggerhead Sea Turtle (*Caretta caretta*), filtered for records in the Mediterranean Sea. The data were sourced from the Global Biodiversity Information Facility (GBIF).

Usage

```
data("caretta")
```

Format

A data frame with 6785 rows and 5 columns:

decimalLongitude Numeric. Longitude coordinates (WGS84).

decimalLatitude Numeric. Latitude coordinates (WGS84).

year Integer. The year in which the occurrence was recorded.

species Character. The scientific name of the species, i.e., *Caretta caretta*.

coordinateUncertaintyInMeters Numeric. The uncertainty of the coordinates in meters.

Details

The dataset has been filtered to include only records within the Mediterranean Sea. The occurrence data cover multiple years, which provides information on the temporal distribution of the species in this region.

Source

Global Biodiversity Information Facility (GBIF), https://www.gbif.org/species/8894817

grid_thinning *Perform Grid-Based Thinning of Spatial Points*

Description

This function performs thinning of spatial points by assigning them to grid cells based on a specified resolution or thinning distance. It can either create a new raster grid or use an existing raster object.

Usage

```
grid_thinning(
  coordinates,
  thin_dist = NULL,
  resolution = NULL,
  origin = NULL,
  raster_obj = NULL,
  trials = 10,
  all_trials = FALSE,
  crs = "epsg:4326",
  priority = NULL
)
```

Arguments

`coordinates` A numeric matrix or data frame with two columns representing the x (longitude) and y (latitude) coordinates of the points.

`thin_dist` A numeric value representing the thinning distance in kilometers. It will be converted to degrees if ‘resolution‘ is not provided.

`resolution` A numeric value representing the resolution (in degrees) of the raster grid. If provided, this takes priority over ‘thin_dist‘.

`origin` A numeric vector of length 2 (for example, ‘c(0, 0)‘), specifying the origin of the raster grid (optional).

`raster_obj` An optional ‘terra‘ SpatRaster object to use for grid thinning. If provided, the raster object will be used instead of creating a new one.

`trials` An integer specifying the number of trials to perform for thinning (default: 10).

`all_trials` A logical value indicating whether to return results for all trials (‘TRUE‘) or just the first trial (‘FALSE‘, default).

`crs` An optional CRS (Coordinate Reference System) to project the coordinates and raster (default WGS84). This can be an EPSG code, a PROJ.4 string, or a ‘terra::crs‘ object.

`priority` A of the same length as the number of points with numerical values indicating the priority of each point. Instead of eliminating points randomly, the points are preferred according to these values.

Value

A list of logical vectors indicating which points to keep for each trial.

Examples

```
# Example: Grid thinning using thin_dist
coordinates <- matrix(c(-122.4194, 37.7749,
                        -122.4195, 37.7740,
                        -122.4196, 37.7741), ncol = 2, byrow = TRUE)

result <- grid_thinning(coordinates, thin_dist = 10, trials = 5, all_trials = TRUE)
print(result)

# Example: Grid thinning using a custom resolution
result_res <- grid_thinning(coordinates, resolution = 0.01, trials = 5)
print(result_res)

# Example: Using a custom raster object
library(terra)
rast_obj <- terra::rast(nrows = 100, ncols = 100, xmin = -123, xmax = -121, ymin = 36, ymax = 38)
result_raster <- grid_thinning(coordinates, raster_obj = rast_obj, trials = 5)
print(result_raster)
```

kd_tree_thinning *Perform K-D Tree ANN Thinning*

Description

This function applies the K-D tree Approximate Nearest Neighbors (ANN) thinning algorithm on a set of spatial coordinates. It can optionally use space partitioning to improve the thinning process, which is particularly useful for large datasets.

Usage

```
kd_tree_thinning(
  coordinates,
  thin_dist = 10,
  trials = 10,
  all_trials = FALSE,
  space_partitioning = FALSE,
  euclidean = FALSE,
  R = 6371
)
```

Arguments

`coordinates` A matrix of coordinates to thin, with two columns representing longitude and latitude.

`thin_dist` A numeric value representing the thinning distance in kilometers. Points closer than this distance to each other are considered redundant and may be removed.

`trials` An integer specifying the number of trials to run for thinning. Multiple trials can help achieve a better result by randomizing the thinning process. Default is 10.

`all_trials` A logical value indicating whether to return results of all attempts (‘TRUE‘) or only the best attempt with the most points retained (‘FALSE‘). Default is ‘FALSE‘.

`space_partitioning`
A logical value indicating whether to use space partitioning to divide the coordinates into grid cells before thinning. This can improve efficiency in large datasets. Default is ‘FALSE‘.

`euclidean` Logical value indicating whether to compute the Euclidean distance (‘TRUE‘) or Haversine distance (‘FALSE‘, default).

`R` A numeric value representing the radius of the Earth in kilometers. The default is 6371 km.

Value

A list. If ‘all_trials‘ is ‘FALSE‘, the list contains a single logical vector indicating which points are kept in the best trial. If ‘all_trials‘ is ‘TRUE‘, the list contains a logical vector for each trial.

Examples

```
# Generate sample coordinates
set.seed(123)
coordinates <- matrix(runif(20, min = -180, max = 180), ncol = 2) # 10 random points

# Perform K-D Tree thinning without space partitioning
result <- kd_tree_thinning(coordinates, thin_dist = 10, trials = 5, all_trials = FALSE)
print(result)

# Perform K-D Tree thinning with space partitioning
result_partitioned <- kd_tree_thinning(coordinates, thin_dist = 5000, trials = 5,
                                       space_partitioning = TRUE, all_trials = TRUE)
print(result_partitioned)

# Perform K-D Tree thinning with Cartesian coordinates
cartesian_coordinates <- long_lat_to_cartesian(coordinates[, 1], coordinates[, 2])
result_cartesian <- kd_tree_thinning(cartesian_coordinates, thin_dist = 10, trials = 5,
                                     euclidean = TRUE)
print(result_cartesian)
```

long_lat_to_cartesian *Convert Geographic Coordinates to Cartesian Coordinates*

Description

This function converts geographic coordinates, given as longitude and latitude in degrees, to Cartesian coordinates (x, y, z) assuming a spherical Earth model.

Usage

```
long_lat_to_cartesian(long, lat, R = 6371)
```

Arguments

long	Numeric vector of longitudes in degrees.
lat	Numeric vector of latitudes in degrees.
R	Radius of the Earth in kilometers (default: 6371 km).

Value

A numeric matrix with three columns (x, y, z) representing Cartesian coordinates.

Examples

```
long <- c(-122.4194, 0)
lat <- c(37.7749, 0)
long_lat_to_cartesian(long, lat)
```

max_thinning_algorithm *Thinning Algorithm for Spatial Data*

Description

This function performs the core thinning algorithm used to reduce the density of points in spatial data while maintaining spatial representation. It works by iteratively removing points with the most neighbors until no points with neighbors remain. The algorithm supports multiple trials to find the optimal thinning solution.

Usage

```
max_thinning_algorithm(neighbor_indices, n, trials, all_trials = FALSE)
```

Arguments

neighbor_indices
A list of integer vectors where each element contains the indices of the neighboring points for each point in the dataset.

n	The number of points in the dataset.
trials	The number of thinning trials to run.
all_trials	If TRUE, returns the results of all trials; if FALSE, returns the best trial with the most points retained (default: FALSE).

Value

A list of logical vectors indicating which points are kept in each trial if all_trials is TRUE; otherwise, a single logical vector indicating the points kept in the best trial.

Examples

```
# Example usage within a larger thinning function
neighbor_indices <- list(c(2, 3), c(1, 3), c(1, 2))
n <- 3
trials <- 5
all_trials <- FALSE
keep_points <- max_thinning_algorithm(neighbor_indices, n, trials, all_trials)
print(keep_points)
```

precision_thinning *Precision Thinning of Spatial Points*

Description

This function performs thinning of spatial points by rounding their coordinates to a specified precision and removing duplicates. It can perform multiple trials of this process and return the results for all or just the best trial.

Usage

```
precision_thinning(
  coordinates,
  precision = 4,
  trials = 10,
  all_trials = FALSE,
  priority = NULL
)
```

Arguments

`coordinates`	A numeric matrix or data frame with two columns representing the longitude and latitude of points.
`precision`	An integer specifying the number of decimal places to which coordinates should be rounded. Default is 4.
`trials`	An integer specifying the number of thinning trials to perform. Default is 10.
`all_trials`	A logical value indicating whether to return results for all trials (‘TRUE‘) or just the first/best trial (‘FALSE‘). Default is ‘FALSE‘.
`priority`	A of the same length as the number of points with numerical values indicating the priority of each point. Instead of eliminating points randomly, the points are preferred accoridng to these values.

Details

The function performs multiple trials to account for randomness in the order of point selection. By default, it returns the first trial, but setting ‘all_trials = TRUE‘ will return the results of all trials.

Value

If ‘all_trials‘ is ‘FALSE‘, returns a logical vector indicating which points were kept in the first trial. If ‘all_trials‘ is ‘TRUE‘, returns a list of logical vectors, one for each trial.

Examples

```
# Example usage
coordinates <- matrix(c(-123.3656, 48.4284, -123.3657, 48.4285, -123.3658, 48.4286), ncol = 2)
result <- precision_thinning(coordinates, precision = 3, trials = 5, all_trials = TRUE)
print(result)

# Example with a single trial and lower precision
result_single <- precision_thinning(coordinates, precision = 2, trials = 1, all_trials = FALSE)
print(result_single)
```

`rounding_hashing_thinning` *Rounding Hashing Thinning*

Description

Performs thinning of geographical coordinates using a hashing approach and rounds the coordinates to create a grid.

Usage

```
rounding_hashing_thinning(
  coordinates,
  thin_dist = 10,
  trials = 10,
  all_trials = FALSE,
  euclidean = FALSE,
  R = 6371,
  seed = NULL
)
```

Arguments

`coordinates`	A numeric matrix of size (n x 2) containing the longitude and latitude of points, where each row represents a coordinate pair.
`thin_dist`	A numeric value specifying the distance (in kilometers) within which points should be considered for thinning.
`trials`	An integer indicating the number of trials to run for the thinning process. More trials may yield better results.
`all_trials`	A logical value indicating whether to return all trials (‘TRUE‘) or only the best trial (‘FALSE‘).
`euclidean`	Logical value indicating whether to compute the Euclidean distance (‘TRUE‘) or Haversine distance (‘FALSE‘, default).
`R`	A numeric value representing the radius of the Earth in kilometers. Default is set to 6371.0 km.
`seed`	Optional; an integer seed for reproducibility of results.

Details

This function applies a hashing technique to group coordinates into grid cells, allowing for efficient thinning based on a specified distance. It can run multiple trials to determine the best set of points to keep, or return all trials if specified.

Value

A logical vector indicating which points are kept after the thinning process. If ‘all_trials‘ is ‘TRUE‘, a list of logical vectors will be returned, one for each trial.

Examples

```
# Generate random coordinates
set.seed(123)
coordinates <- matrix(runif(20, min = -180, max = 180), ncol = 2) # 10 random points

# Perform rounding hashing thinning
result <- rounding_hashing_thinning(coordinates, thin_dist = 10, trials = 5)
print(result)
```

```
# Perform thinning with all trials
all_results <- rounding_hashing_thinning(coordinates, thin_dist = 5000, trials = 5,
                                         all_trials = TRUE)
print(all_results)

# Perform thinning with euclidean distance
result_euclidean <- rounding_hashing_thinning(coordinates, thin_dist = 10,
                                              trials = 5, euclidean = TRUE)
print(result_euclidean)
```

thin_points *Spatial Thinning of Points*

Description

This function performs spatial thinning of geographic points to reduce point density while maintaining spatial representation. Points are thinned based on a specified distance, grid, or precision, and multiple trials can be performed to identify the best thinned dataset.

Usage

```
thin_points(
  data,
  long_col = NULL,
  lat_col = NULL,
  group_col = NULL,
  method = c("brute_force", "kd_tree", "round_hash", "grid", "precision"),
  trials = 10,
  all_trials = FALSE,
  target_points = NULL,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`data` A data frame or tibble containing the points to thin. Must contain longitude and latitude columns.

`long_col` Name of the column with longitude coordinates (default: "decimalLongitude").

`lat_col` Name of the column with latitude coordinates (default: "decimalLatitude").

`group_col` Name of the column for grouping points (e.g., species name, year). If NULL, no grouping is applied.

`method` Thinning method to use ‘c("brute_force", "kd_tree", "round_hash", "grid", "precision")‘.

`trials` Number of thinning iterations to perform (default: 10).

`all_trials` If TRUE, returns results of all attempts; if FALSE, returns the best attempt with the most points retained (default: FALSE).

`target_points` Optional; a numeric value specifying the exact number of points to keep. If NULL (default), maximizes the number of kept points.

`seed` Optional; an integer seed for reproducibility of results.

`verbose` If TRUE, prints progress messages (default: FALSE).

`...` Additional parameters passed to specific thinning methods (e.g., thin_dist, precision, resolution, origin, R).

Details

The thinning methods available are: - ‘brute_force‘: Uses a brute force approach to thin points. - ‘kd_tree‘: Uses K-D trees for thinning. - ‘round_hash‘: Uses rounding and hashing for efficient thinning. - ‘grid‘: Applies a grid-based thinning method. - ‘precision‘: Utilizes precision-based thinning.

For more information on specific thinning methods and inputs, refer to their respective documentation: - ‘brute_force_thinning()‘ - ‘grid_thinning()‘ - ‘kd_tree_thinning()‘ - ‘rounding_hashing_thinning()‘ - ‘precision_thinning()‘

Value

A tibble of thinned points, or a combined result of all attempts if ‘all_trials‘ is TRUE.

Examples

```
# Generate sample data
set.seed(123)
sample_data <- data.frame(
  decimalLongitude = runif(100, -180, 180),
  decimalLatitude = runif(100, -90, 90)
)

# Perform thinning using K-D tree method
thinned_data <- thin_points(sample_data,
                             long_col = "decimalLongitude",
                             lat_col = "decimalLatitude",
                             method = "kd_tree",
                             trials = 5,
                             verbose = TRUE)

# Perform thinning with grouping
sample_data$species <- sample(c("species_A", "species_B"), 100, replace = TRUE)
thinned_grouped_data <- thin_points(sample_data,
                                     long_col = "decimalLongitude",
                                     lat_col = "decimalLatitude",
                                     group_col = "species",
                                     method = "kd_tree",
                                     trials = 10)
```

Index