Package 'GeoThinneR'

November 25, 2025

Type Package

Title Efficient Spatial Thinning of Species Occurrences

Version 2.1.0

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, doParallel, fields, foreach, graphics, grDevices, matrixStats, nabor, sf, stats, terra

Suggests ggplot2, knitr, rmarkdown, s2 (>= 1.1.0), testthat (>= 3.0.0), tibble

VignetteBuilder knitr

BuildVignettes true

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-11-25 06:11:02 UTC

Contents

`as_GeoThinned` *GeoThinned Object Constructor and Methods*

Description

Create and interact with spatial thinning results stored in a GeoThinned object.

Usage

```
as_GeoThinned(retained, method, params = list(), original_data = NULL)

new_GeoThinned(retained, method, params = list(), original_data = NULL)

## S3 method for class 'GeoThinned'
print(x, ...)

## S3 method for class 'GeoThinned'
summary(object, trial = NULL, ...)

## S3 method for class 'summary.GeoThinned'
print(x, ...)

## S3 method for class 'GeoThinned'
plot(
  x,
  trial = NULL,
```

```
  show_original = TRUE,
  col_original = "#EB714B",
  col_thinned = "#5183B3",
  pch_original = 1,
  pch_thinned = 16,
  main = NULL,
  ...
)

largest(x, ...)

## S3 method for class 'GeoThinned'
largest(x, ...)

largest_index(x, ...)

## S3 method for class 'GeoThinned'
largest_index(x, ...)

get_trial(x, trial = NULL, ...)

## S3 method for class 'GeoThinned'
get_trial(x, trial = NULL, ...)

as_sf(x, ...)

## S3 method for class 'GeoThinned'
as_sf(x, trial = NULL, crs = 4326, ...)
```

Arguments

retained	A list of logical vectors indicating retained points per trial.
method	The thinning method used (e.g., "distance", "grid", "precision").
params	A list of parameters used in thinning.
original_data	The original unmodified data.
x	An object of class GeoThinned.
...	Additional arguments (ignored).
object	An object of class GeoThinned.
trial	Integer index of the thinning trial to extract (for summary(), get_trial(), plot(), as_sf()). Default 'NULL', which will return the largest dataset.
show_original	Logical, whether to show original points.
col_original	Colors for original points.
col_thinned	Colors for thinned points.
pch_original	Point shapes for original points.
pch_thinned	Point shapes for thinned points.

main	Title of the plot.
crs	Coordinate reference system to assign to the resulting sf object (optional).

Value

A GeoThinned object or associated results (summary, plot, trial subset). When ‘thin_points()‘ is run with ‘all_trials = FALSE‘, the returned object contains only the largest trial; therefore all methods refer to this single subset.

See Also

thin_points

---

calculate_spatial_coverage

Calculate Spatial Coverage (Convex Hull Area)

Description

Computes the area of the convex hull formed by the points. Uses geodetic area (km2) if coordinates are lon/lat and distance = "haversine", otherwise computes area in squared map units.

Usage

```
calculate_spatial_coverage(coordinates, distance = "haversine")
```

Arguments

coordinates	A matrix of coordinates (longitude and latitude or planar x/y).
distance	A character string: "haversine" (default) or "euclidean".

Value

A numeric value representing the convex hull area (km2 or unit2).

Examples

```
# Geographic coordinates (lon/lat)
set.seed(456)
coords_geo <- matrix(cbind(runif(10, -10, 10), runif(10, 40, 50)), ncol = 2)
area_haversine <- calculate_spatial_coverage(coords_geo, distance = "haversine")
print(round(area_haversine, 2))  # in km2

# Projected coordinates (Euclidean/map units)
coords_proj <- matrix(runif(20), ncol = 2) * 100  # e.g., map units
area_euclidean <- calculate_spatial_coverage(coords_proj, distance = "euclidean")
print(round(area_euclidean, 2))  # in unit2
```

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

Description

This dataset contains georeferenced occurrence records of the Loggerhead Sea Turtle (*Caretta caretta*) within the Mediterranean Sea from 1962 to 2025. The data were obtained from the Global Biodiversity Information Facility (GBIF) and filtered to include only marine points falling within a defined Mediterranean polygon. Duplicated coordinates were removed.

Usage

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

Format

A data frame with 8340 rows and 5 variables:

decimalLongitude Numeric. Longitude coordinates (WGS84).

decimalLatitude Numeric. Latitude coordinates (WGS84).

year Integer. Year in which the observation was recorded.

species Character. The species name (*Caretta caretta*).

coordinateUncertaintyInMeters Numeric. Positional uncertainty of the coordinates, in meters.

Details

The data were accessed through the GBIF API and clipped spatially using a Mediterranean Sea polygon (https://www.marineregions.org/gazetteer.php?p=details&id=1905). All records were filtered to ensure presence-only data with valid geographic coordinates. Duplicate coordinate pairs were removed to reduce redundancy.

This dataset is used as a real-world example throughout the GeoThinneR package and the accompanying article.

For reproducibility, the GBIF download script is included in `data-raw/caretta_download.R`.

The Mediterranean Sea polygon used to spatially filter the records was download from Marineregions.org: https://www.marineregions.org/gazetteer.php?p=details&id=1905. It is distributed with the package as `inst/extdata/mediterranean_sea.gpkg` and used in the `data-raw/caretta_download.R` script.

Source

Global Biodiversity Information Facility (GBIF)
GBIF Occurrence Download https://doi.org/10.15468/dl.9jcjrm
Accessed from R via **rgbif** (https://github.com/ropensci/rgbif) on 2025-02-07

`compute_nearest_neighbor_distances` *Compute Nearest Neighbor Distances*

Description

Calculates nearest neighbor distances using geodesic or Euclidean distance.

Usage

```
compute_nearest_neighbor_distances(
  coordinates,
  distance = "haversine",
  R = 6371
)
```

Arguments

`coordinates`	A matrix of coordinates with two columns.
`distance`	A character string: "haversine" (default) or "euclidean".
`R`	Radius of the Earth in kilometers. Default is 6371.

Value

A numeric vector of nearest neighbor distances, in meters (haversine) or in map units (euclidean).

Examples

```
# Example with geographic (longitude/latitude) coordinates
set.seed(123)
coords_geo <- matrix(cbind(runif(10, -10, 10), runif(10, 40, 50)), ncol = 2)
nnd_haversine <- compute_nearest_neighbor_distances(coords_geo, distance = "haversine")
print(round(nnd_haversine, 2))  # in km

# Example with projected coordinates (Euclidean)
coords_proj <- matrix(runif(20), ncol = 2) * 100  # e.g., meters or map units
nnd_euclidean <- compute_nearest_neighbor_distances(coords_proj, distance = "euclidean")
print(round(nnd_euclidean, 2))
```

compute_neighbors_brute *Compute Neighbors Using Brute-Force*

Description

Computes neighbors for each point in a set of coordinates using a greedy approach. All pairwise distances are calculated to identify neighbors within a specified distance threshold.

Usage

```
compute_neighbors_brute(
  coordinates,
  thin_dist,
  distance = c("haversine", "euclidean"),
  R = 6371
)
```

Arguments

coordinates	A matrix of coordinates to thin, with two columns representing longitude and latitude.
thin_dist	A positive numeric value representing the thinning distance in kilometers.
distance	A character string specifying the distance metric to use ‘c("haversine", "euclidean")‘.
R	A numeric value representing the radius of the Earth in kilometers. The default is 6371 km.

Value

A list where each element corresponds to a point and contains the indices of its neighbors.

Examples

```
set.seed(123)
coords <- matrix(runif(20, min = -180, max = 180), ncol = 2)

# Compute neighbors using brute fore
neighbors <- compute_neighbors_brute(coords, thin_dist = 10,)
```

compute_neighbors_kdtree *Compute Neighbors Using kd-Tree*

Description

Computes neighbors for each point in a set of coordinates using a kd-tree for efficient neighbor searches. This method is particularly useful for large datasets.

Usage

```
compute_neighbors_kdtree(
  coordinates,
  thin_dist,
  k = NULL,
  distance = c("haversine", "euclidean"),
  R = 6371
)
```

Arguments

coordinates	A matrix of coordinates to thin, with two columns representing longitude and latitude.
thin_dist	A positive numeric value representing the thinning distance in kilometers.
k	An integer specifying the maximum number of neighbors to consider for each point.
distance	A character string specifying the distance metric to use ‘c("haversine", "euclidean")‘.
R	A numeric value representing the radius of the Earth in kilometers. The default is 6371 km.

Details

This function uses kd-tree (via ‘nabor‘ package) for efficient spatial searches. The kd-tree inherently works with Euclidean distances. If ‘"haversine"‘ is selected, the function first converts geographic coordinates to 3D Cartesian coordinates before constructing the kd-tree.

Value

A list where each element corresponds to a point and contains the indices of its neighbors, excluding the point itself.

Examples

```
set.seed(123)
coords <- matrix(runif(20, min = -180, max = 180), ncol = 2)

# Compute neighbors using kd-tree
neighbors <- compute_neighbors_kdtree(coords, thin_dist = 10,)
```

`compute_neighbors_local_kdtree`

Compute Neighbors Using Local kd-Trees

Description

Divides the search area into a grid of local regions and constructs kd-trees for each region to compute neighbors efficiently. Neighbor regions are also considered to ensure a complete search.

Usage

```
compute_neighbors_local_kdtree(
  coordinates,
  thin_dist,
  distance = c("haversine", "euclidean"),
  R = 6371,
  n_cores = 1
)
```

Arguments

`coordinates`	A matrix of coordinates to thin, with two columns representing longitude and latitude.
`thin_dist`	A positive numeric value representing the thinning distance in kilometers.
`distance`	A character string specifying the distance metric to use ‘c("haversine", "euclidean")‘.
`R`	A numeric value representing the radius of the Earth in kilometers. The default is 6371 km.
`n_cores`	An integer specifying the number of cores to use for parallel processing. The default is 1.

Value

A list where each element corresponds to a point and contains the indices of its neighbors, excluding the point itself.

Examples

```
set.seed(123)
coords <- matrix(runif(20, min = -180, max = 180), ncol = 2)

# Compute neighbors using local kd-trees with Euclidean distance
neighbors <- compute_neighbors_local_kdtree(coords, thin_dist = 10, n_cores = 1)
```

`distance_thinning` *Perform Distance-Based Thinning*

Description

This function applies a distance-based thinning algorithm using a kd-tree or brute-force approach. Two modified algorithms based on kd-trees (local kd-trees and estimating the maximum number of neighbors) are implemented which scale better for large datasets. The function removes points that are closer than a specified distance to each other while maximizing spatial representation.

Usage

```
distance_thinning(
  coordinates,
  thin_dist = 10,
  trials = 10,
  all_trials = FALSE,
  search_type = c("local_kd_tree", "k_estimation", "kd_tree", "brute"),
  target_points = NULL,
  priority = NULL,
  distance = c("haversine", "euclidean"),
  R = 6371,
  n_cores = 1
)
```

Arguments

`coordinates`	A matrix of coordinates to thin, with two columns representing longitude and latitude.
`thin_dist`	A positive numeric value representing the thinning distance in kilometers.
`trials`	An integer specifying the number of trials to run for thinning. Default is 10.
`all_trials`	A logical indicating whether to return results of all attempts (‘TRUE‘) or only the best attempt with the most points retained (‘FALSE‘). Default is ‘FALSE‘.
`search_type`	A character string indicating the neighbor search method ‘c("local_kd_tree", "k_estimation", "kd_tree", "brute")‘. The default value is ‘local_kd_tree‘. See details.
`target_points`	Optional integer specifying the number of points to retain. If ‘NULL‘ (default), the function tries to maximize the number of points retained.

priority	A numeric vector of the same length as the number of points, specifying a priority weight for each point. Higher values indicate higher importance and are favored when selecting which points to retain. Priority is used to guide selection when multiple candidate points are otherwise equally valid (e.g., points in the same grid cell, with the same rounded coordinates, or with the same number of neighbors).
distance	Distance metric to use ‘c("haversine", "euclidean")‘. Default is Haversine for geographic coordinates.
R	Radius of the Earth in kilometers (default: 6371 km).
n_cores	Number of cores for parallel processing (only for ‘"local_kd_tree"‘). Default is 1.

Details

- ‘"kd_tree"‘: Uses a single kd-tree for efficient nearest-neighbor searches. - ‘"local_kd_tree"‘: Builds multiple smaller kd-trees for better scalability. - ‘"k_estimation"‘: Approximates a maximum number of neighbors per point to reduce search complexity. - ‘"brute"‘: Computes all pairwise distances (inefficient for large datasets).

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)
result  <- matrix(runif(20, min = -180, max = 180), ncol = 2) # 10 random points

# Perform thinning with local kd-trees
result_partitioned <- distance_thinning(result , thin_dist = 5000, trials = 5,
                                        search_type = "local_kd_tree", all_trials = TRUE)
print(result_partitioned)

# Perform thinning estimating max number of neighbors
result_estimated <- distance_thinning(result , thin_dist = 5000, trials = 5,
                                        search_type = "k_estimation", all_trials = TRUE)
print(result_estimated)
```

---

estimate_k_max *Estimate Maximum Neighbors for kd-Tree Thinning*

Description

This function estimates the maximum value of k (the number of nearest neighbors) for kd-tree-based thinning by evaluating the densest regions of a spatial dataset. The function uses a histogram-based binning approach for efficiency and low memory usage.

Usage

```
estimate_k_max(coordinates, thin_dist, distance = c("haversine", "euclidean"))
```

Arguments

coordinates	A matrix of spatial coordinates with two columns for longitude and latitude.
thin_dist	A positive numeric value representing the thinning distance in kilometers. This defines the resolution of the grid used for density calculations.
distance	Distance metric used ‘c("haversine", "euclidean")‘.

Details

The function divides the spatial domain into grid cells based on the specified thinning distance. Grid cell sizes are determined assuming approximately 111.32 km per degree (latitude/longitude). The function identifies the densest grid cells and their immediate neighbors to compute the maximum k value.

Value

A numeric value representing the maximum k (number of nearest neighbors) required for the densest regions in the dataset.

Examples

```
# Generate sample data
set.seed(123)
coordinates <- matrix(runif(200, min = -10, max = 10), ncol = 2)

# Estimate k for kd-tree thinning
k_max <- estimate_k_max(coordinates, thin_dist = 50)
print(k_max)
```

---

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 ‘terra::SpatRaster‘ object.

Usage

```
grid_thinning(
  coordinates,
  thin_dist = NULL,
  resolution = NULL,
  origin = NULL,
  raster_obj = NULL,
  n = 1,
  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 (e.g., '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.
n	A positive integer specifying the maximum number of points to retain per grid cell (default: 1).
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, 'epsg:4326'). This can be an EPSG code, a PROJ.4 string, or a 'terra::crs' object.
priority	A numeric vector of the same length as the number of points, specifying a priority weight for each point. Higher values indicate higher importance and are favored when selecting which points to retain. Priority is used to guide selection when multiple candidate points are otherwise equally valid (e.g., points in the same grid cell, with the same rounded coordinates, or with the same number of neighbors).

Value

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

Examples

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

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

# Example: Grid thinning using a custom resolution
result_res <- grid_thinning(coords, resolution = 0.01, n = 2, 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(coords, raster_obj = rast_obj, trials = 5)
print(result_raster)
```

---

is_lonlat *Check for Longitude/Latitude Coordinates*

Description

This function checks whether a pair of coordinate vectors represent geographic longitude and latitude values. The function returns 'TRUE' if all longitude values are within -180 (- tolerance) and 180 (+ tolerance) and the latitude is within -90 (- tolerance) and 90 (+ tolerance).

Usage

```
is_lonlat(lon, lat, tolerance = 0.1)
```

Arguments

lon	Numeric vector of longitudes in degrees.
lat	Numeric vector of latitudes in degrees.
tolerance	Numeric tolerance (in degrees) for checking the global longitude/latitude bounds. Default is 0.1.

Value

A logical value. 'TRUE' if values are within the ranges and 'FALSE' otherwise.

Examples

```
is_lonlat(lon = c(-3, 10, 179), lat = c(40, -20, 5))
is_lonlat(lon = c(100000, 150000), lat = c(4500000, 4600000))
```

lon_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

```
lon_lat_to_cartesian(lon, lat, R = 6371)
```

Arguments

lon	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

```
lon <- c(-122.4194, 0)
lat <- c(37.7749, 0)
lon_lat_to_cartesian(lon, 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 iteratively removes the 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,
  trials,
  all_trials = FALSE,
  priority = NULL
)
```

Arguments

neighbor_indices	A list of integer vectors where each element contains the indices of the neighboring points for each point in the dataset.
trials	A positive integer specifying the number of thinning trials to perform. 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'.
priority	A numeric vector of the same length as the number of points, specifying a priority weight for each point. Higher values indicate higher importance and are favored when selecting which points to retain. Priority is used to guide selection when multiple candidate points are otherwise equally valid (e.g., points in the same grid cell, with the same rounded coordinates, or with the same number of neighbors).

Value

A list of logical vectors indicating which points are kept in each trial if all_trials is TRUE; otherwise, a list with 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))
trials <- 5
all_trials <- FALSE
kept_points <- max_thinning_algorithm(neighbor_indices, trials, all_trials)
print(kept_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	A positive integer specifying the number of decimal places to which coordinates should be rounded. Default is 4.
trials	A positive 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 numeric vector of the same length as the number of points, specifying a priority weight for each point. Higher values indicate higher importance and are favored when selecting which points to retain. Priority is used to guide selection when multiple candidate points are otherwise equally valid (e.g., points in the same grid cell, with the same rounded coordinates, or with the same number of neighbors).

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

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

`select_target_points` *Select Target Number of Points for Spatial Thinning*

Description

This function selects a specified number of points from a spatial dataset while maximizing the distance between selected points.

Usage

```
select_target_points(
  distance_matrix,
  target_points,
  thin_dist,
  trials,
  all_trials = FALSE
)
```

Arguments

`distance_matrix`	A matrix of pairwise distances between points.
`target_points`	An integer specifying the number of points to retain.
`thin_dist`	A positive numeric value representing the thinning distance in kilometers.
`trials`	A positive integer specifying the number of thinning trials to perform. 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‘.

Value

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

Examples

```
# Example distance matrix (3 points)
dist_matrix <- matrix(c(0, 2, 5,
                        2, 0, 3,
                        5, 3, 0), ncol = 3)

# Select 2 points maximizing distance
result <- select_target_points(dist_matrix, target_points = 2,
                               thin_dist = 4, trials = 5, all_trials = TRUE)
```

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 decimal precision, with support for multiple trials and optional grouping. By default, only the largest subset is returned. To return all trials, set ‘all_trials = TRUE‘.

Usage

```
thin_points(
  data,
  lon_col = "lon",
  lat_col = "lat",
  group_col = NULL,
  method = c("distance", "grid", "precision"),
  trials = 10,
  all_trials = FALSE,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

data	A data frame or tibble containing the input points to thin. Must contain longitude and latitude columns.
lon_col	Character name of the column with longitude coordinates (default: ‘"lon"‘).
lat_col	Character name of the column with latitude coordinates (default: ‘"lat"‘).
group_col	Character name of the column for grouping points (e.g., species name, year). If ‘NULL‘, no grouping is applied.
method	Thinning method to use. One of ‘"distance", "grid", "precision"‘.
trials	Number of thinning iterations to perform (default: ‘10‘). Must be a positive integer.
all_trials	If ‘TRUE‘, returns results of all attempts; if ‘FALSE‘, returns the best attempt with the most points retained (default: ‘FALSE‘).
seed	Optional; an integer seed for reproducibility of results.
verbose	If ‘TRUE‘, prints progress messages (default: ‘FALSE‘).
...	Additional arguments passed to specific thinning methods. See Details.

Details

The following thinning methods are available:

‘"distance"‘ Forces a specific minimum distance between points.

‘"grid"‘ Applies a grid-based thinning method.

‘"precision"‘ Utilizes precision-based thinning.

Distance-based thinning

The specific parameters for distance-based thinning are:

‘thin_dist‘ A positive numeric value representing the thinning distance in kilometers.

‘search_type‘ A character string indicating the neighbor search method ’c("local_kd_tree", "k_estimation", "kd_tree", "brute")’. The defult value is ’local_kd_tree’.

‘distance‘ Distance metric to use ’c("haversine", "euclidean")’. Default is Haversine for geographic coordinates.

‘R‘ The radius of the Earth in kilometers. Default is 6371 km.

‘target_points‘ Optional integer specifying the number of points to retain. If ’NULL’ (default), the function tries to maximize the number of points retained.

‘n_cores‘ Number of cores for parallel processing (only for ’"local_kd_tree"’). Default is 1.

Grid-based thinning

The specific parameters for grid-based thinning are:

‘thin_dist‘ A positive numeric value representing the thinning distance in kilometers.

‘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 (e.g., ’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.

‘n‘ A positive integer specifying the maximum number of points to retain per grid cell (default: 1).

‘crs‘ An optional CRS (Coordinate Reference System) to project the coordinates and raster (default WGS84, ’epsg:4326’). This can be an EPSG code, a PROJ.4 string, or a ’terra::crs’ object.

‘priority‘ A numeric vector of the same length as the number of points with numerical values indicating the priority of each point. Instead of eliminating points randomly, higher values are preferred during thinning.

Precision-based thinning

The specific parameters for precision-based thinning are:

‘precision‘ A positive integer specifying the number of decimal places to which coordinates should be rounded. Default is 4.

‘priority‘ A numeric vector of the same length as the number of points with numerical values indicating the priority of each point. Instead of eliminating points randomly, higher values are preferred during thinning.

For more information on specific thinning methods and inputs, refer to their respective documentation:

- ‘distance_thinning()‘
- ‘grid_thinning()‘
- ‘precision_thinning()‘

Value

A ‘GeoThinned‘ object (S3 class), which contains:

- ‘retained‘: A list of logical vectors (one per trial) indicating retained points.
- ‘original_data‘: The original input dataset.
- ‘method‘: The thinning method used.
- ‘params‘: A list of the thinning parameters used.

By default, ‘thin_points()‘ returns only the trial with the largest number of retained points. To access all thinning trials, set ‘all_trials = TRUE‘; otherwise, functions such as ‘largest()‘ and ‘get_trial()‘ will always refer to the same subset.

Examples

```
# Basic usage
set.seed(123)
sample_data <- data.frame(
  lon = runif(100, -10, 10),
  lat = runif(100, -5, 5)
)

result <- thin_points(sample_data, method = "distance", thin_dist = 100)

# Grouped thinning
sample_data$species <- sample(c("A", "B"), 100, replace = TRUE)
grouped_result <- thin_points(sample_data, group_col = "species",
                              method = "distance", thin_dist = 100)
```

---

thunnus *Yellowfin Tuna (*Thunnus albacares*) Worldwide Occurrence Records*

Description

This dataset contains georeferenced occurrence records of the Yellowfin Tuna (*Thunnus albacares*) from 1950 to 2025, obtained globally from the Global Biodiversity Information Facility (GBIF). The dataset was filtered to include presence-only records with valid geographic coordinates and year, and duplicate coordinates were removed.

Usage

```
data("thunnus")
```

Format

A data frame with 80,163 rows and 3 variables:

decimalLongitude Numeric. Longitude coordinates (WGS84).

decimalLatitude Numeric. Latitude coordinates (WGS84).

year Integer. Year in which the observation was recorded.

Details

The data were accessed through the GBIF API and used for benchmarking thinning algorithms in the GeoThinneR package. Coordinates include marine records, and the data were globally sampled across many decades.

For reproducibility, the GBIF download script is included in `data-raw/thunnus_download.R`.

Source

Global Biodiversity Information Facility (GBIF)
GBIF Occurrence Download https://doi.org/10.15468/dl.xsyrkh
Accessed from R via **rgbif** (https://github.com/ropensci/rgbif) on 2025-02-07

Index