GeoThinneR: An R Package for Efficient Spatial Thinning of Species Occurrences and Point Data

by Jorge Mestre-Tomás

Abstract In this paper, we present GeoThinneR, an R package for efficient and flexible spatial thinning of species occurrence data. Spatial thinning is a widely used preprocessing step in species distribution modelling (SDM) that can help reduce sampling bias, but existing R implementations rely on brute-force algorithms that scale poorly with large datasets. GeoThinneR implements multiple thinning approaches, including ensuring a minimum distance between points, subsampling points on a grid, and filtering based on decimal precision. To handle large datasets, it introduces two optimized algorithms based on local *kd*-trees and adaptive neighbour estimation, which greatly reduce memory usage and execution time. Additional functionalities such as group-wise thinning and point prioritization are included to facilitate its use in SDM workflows. Here, we provide performance benchmarks using both simulated and real-world data to demonstrate substantial performance improvements over existing tools.

1 Introduction

Presence-only data are often the main source of information for modelling species distributions, as systematic survey data are rarely available (Pearce and Boyce, 2006). In recent years the availability of spatial biodiversity data has increased, with repositories such as the Global Biodiversity Information Facility (GBIF, `https://www.gbif.org`) and the Ocean Biodiversity Information System (OBIS, `https://obis.org/`) providing datasets containing hundreds of thousands of species occurrence records. These larger datasets provide new opportunities in ecological research and species distribution modelling (SDM), which are used to model the species geographic distributions based on occurrence records and environmental variables (Pulliam, 2000; Elith and Leathwick, 2009; Miller, 2010; Guisan et al., 2013). However, as dataset sizes grow, the computational requirements for processing and filtering them increase too. Species occurrence data often show spatial sampling bias due to opportunistic observations, uneven sampling effort or a combination of information from diverse sources that vary in how and where data were collected, resulting in heterogeneous datasets (Boakes et al., 2010; Lobo and Tognelli, 2011; Beck et al., 2014; Meyer et al., 2016; Hughes et al., 2021). These biases result in clusters of records that can artificially increase spatial autocorrelation among occurrences and cause the model to overfit the environmental conditions in these oversampled regions, leading to models that reflect patterns in sampling effort rather than true species distributions (Wisz et al., 2008; Phillips et al., 2009; Higa et al., 2015; Cosentino and Maiorano, 2021; Steen et al., 2021).

Several approaches exist to address sampling bias (Inman et al., 2021; Baker et al., 2024; Moudrý et al., 2024), including adjusting background points (Barber et al., 2022; Vollering et al., 2019), applying environmental filtering (Castellanos et al., 2019), or incorporating bias-related covariates (Varela et al., 2014; Chauvier et al., 2021). Some modelling frameworks attempt to account for sampling bias directly, for example by modelling preferential sampling through point processes (Diggle et al., 2010; Pennino et al., 2019; Amaral et al., 2024) or by combining opportunistically collected presence-only data with systematically sampled data within integrated SDMs (Koshkina et al., 2017; Simmonds et al., 2020; Mäkinen et al., 2024). Here we focus on one of the most widely used preprocessing methods to mitigate these biases across the geographic space, known as spatial thinning, which selectively filters points based on specified criteria to reduce the overrepresentation of the densest locations (Veloz, 2009; Boria et al., 2014). The two most common thinning methods in SDM are distance-based thinning, where points are removed if they fall within a specified minimum distance of another point (Aiello-Lammens et al., 2015), and grid-based thinning, which retains only a specified number of points per grid cell (Hijmans et al., 2023).

While existing distance-based thinning tools in R, such as **spThin** (Aiello-Lammens et al., 2015) and **enmSdmX** (Smith et al., 2023), are widely used, they have several limitations that become more challenging as dataset sizes continue to increase. These tools rely on brute-force algorithms for nearest-neighbour searches given a distance threshold, which are unfeasible to run on personal computers as they scale poorly for datasets containing hundreds of thousands of points. On the other hand, grid-based thinning, such as the one implemented in the **dismo** package (Hijmans et al., 2023), offers better computational efficiency but lacks precision when requiring a specified distance threshold. Additionally, these tools have limited functionality and usability as they often focus on a

single thinning approach and offer few customization options for specific SDM applications, such as thinning occurrences across multiple species (Noori et al., 2024). This can be useful when modelling several species or community-level patterns, for example in stacked SDMs (SSDMs) where species are modelled independently and then combined (Calabrese et al., 2014), or in joint species distribution models (JSDMs) where multiple species are analysed simultaneously accounting for the multivariate nature of the data (Pollock et al., 2014; Ovaskainen et al., 2017; Ovaskainen and Abrego, 2020). Other useful features include retaining an exact number of occurrences (Steen et al., 2021) and prioritizing points based on data quality or other features (Yu et al., 2023).

To overcome these challenges, we introduce **GeoThinneR** (Mestre-Tomás, 2025), an R package designed to provide efficient and flexible spatial thinning for large-scale datasets. **GeoThinneR** integrates multiple thinning approaches into a single package simplifying user experience (distance, grid, and decimal precision-based thinning), offers an optimized distance-based thinning using enhanced algorithms based on *kd*-tree structures, which improve nearest-neighbour search efficiency (Friedman et al., 1975), together with additional specific functionalities for SDM workflows including thinning by species or groups, the ability to retain an exact number of occurrences, and prioritization of points based on user-defined variables.

In Section 2, we present the methods and features implemented in **GeoThinneR**. Section 3 describes the optimized algorithms for large datasets together with a performance comparison between the distance-based methods available. Section 4 benchmarks the performance of the package with existing software, both with simulated and real-world datasets. Finally, Section 5 presents a short conclusion of the work and discusses future directions.

2 GeoThinneR package description

Given a set of points $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, spatial thinning aims to filter $\mathbf{S}$ to obtain a subset $\mathbf{S}' \subseteq \mathbf{S}$ that meets a specified selection criteria. Different thinning methods can be employed to apply this criteria, such as enforcing a minimum separation distance between retained points, limiting the number of points per grid cell, or rounding coordinates to a specified precision. The thinning process iteratively evaluates each point, referred to as the query point q, identifies neighbour points within $\mathbf{S}$, and applies the selection criteria to return a thinned subset $\mathbf{S}'$.

The **GeoThinneR** package is implemented in R and can be downloaded from the Comprehensive R Archive Network (CRAN) at `https://cran.r-project.org/package=GeoThinneR`. Comprehensive documentation and usage examples are available on GitHub and CRAN. One of the advantages of **GeoThinneR** is its flexibility and ease of use, allowing users to efficiently apply different spatial thinning configurations to their datasets. In this section, we demonstrate how to use the package and explore its various thinning methods and additional functionalities.

The main function to apply spatial thinning is `thin_points()`, which takes a data frame containing a set of points with longitude and latitude columns. The key arguments are the thinning `method` (`"distance"`, `"grid"`, or `"precision"`), the thinning criterion (`thin_dist`, `resolution`, or `precision`), and the number of independent thinning attempts (`trials`). By default, only the subset retaining the largest number of points is returned; to keep all thinning trials, the argument `all_trials = TRUE` must be specified. To illustrate the basic usage, we simulate a toy dataset of 100 uniformly distributed points for two species across an area of 1 x 1 degree and apply spatial thinning using the `thin_points()` function:

```
library(GeoThinneR)
# Simulate data
set.seed(2547)
sim_data <- data.frame(
  lon = runif(100, 0, 1),
  lat = runif(100, 0, 1),
  group = sample(c("species_1", "species_2"), 100, replace = TRUE)
)

thin_results <- thin_points(sim_data,
  method = "distance", thin_dist = 10,
  trials = 10, all_trials = TRUE, seed = 567
)
```

GeoThinneR returns the results structured as a `GeoThinned` object, which contains: a list of logical vectors indicating which points were retained in each trial (`x$retained`), the method used for thinning (`x$method`), the parameters used for thinning (`x$params`), and the original dataset (`x$original_data`).

The `summary()` function provides information about the thinning process, including the number of points retained after thinning, the nearest neighbour distance (NND), and the spatial coverage of the retained points. By default, the summary shows the metrics for the trial that retains the largest number of points, but users can also specify a specific trial using the `trial` argument.

```
summary(thin_results)
```

```
#> Summary of GeoThinneR Results
#> ----------------------------
#> Trial summarized  : 2
#> Method used       : distance
#> Distance metric   : haversine
#>
#> Number of points:
#>   Original               100
#>   Thinned                 46
#>   Retention ratio      0.460
#>
#> Nearest Neighbor Distances [km]
#>             Min 1st Qu. Median   Mean 3rd Qu.  Max
#> Original  1.054   3.699  5.349  5.917   7.474 16.4
#> Thinned  10.254  10.899 11.700 11.968  12.646 16.4
#>
#> Spatial Coverage (Convex Hull Area) [km2]
#>   Original         10453.599
#>   Thinned          10111.415
```

The `summary()` output shows that 46 of the 100 points were retained (a retention ratio of 0.46), increasing the minimum nearest-neighbour distance from 1.05 km in the original data to 10.25 km after thinning. Spatial coverage changes only slightly (from 10,453.59 km^2 to 10,111.42 km^2), indicating that the overall extent of the points is mostly maintained. Using the `plot()` function, we can visualize the original and thinned datasets.

```
# Visualize largest thinned trial
plot(thin_results)
# Show retained points only
plot(thin_results, show_original = FALSE)
# Change the colors
plot(thin_results, col_original = "red", col_thinned = "blue")
```

For grid-based and precision-based thinning, retaining the maximum number of points that meet the thinning constraint is simple and straightforward. In contrast, distance-based thinning is a non-deterministic problem related to the set packing problem in combinatorial mathematics. Therefore, as in **spThin**, **GeoThinneR** allows users to perform multiple independent thinning trials, which may produce different subsets of retained points. Using the `largest()` and `get_trial()` functions, users can extract the largest thinned dataset or any of the trials, respectively.

```
# Extract the largest thinned dataset
largest(thin_results)
# Extract the index of the largest dataset
largest_index(thin_results)
# Extract the second trial
get_trial(thin_results, 2)
```

2.1 Overview of thinning methods

GeoThinneR integrates three distinct methods to provide flexibility: distance-based, grid-based, and precision-based thinning (Figure 1). For better usability, all methods are implemented within a modular wrapper function, `thin_points()`, which includes the thinning methods with optimized algorithms for large datasets.

Distance-based thinning (`method="distance"`) ensures that all retained points are separated by at least a user-defined minimum distance d (Aiello-Lammens et al., 2015). However, efficiently identifying neighbour points becomes computationally challenging as dataset sizes increase. For a

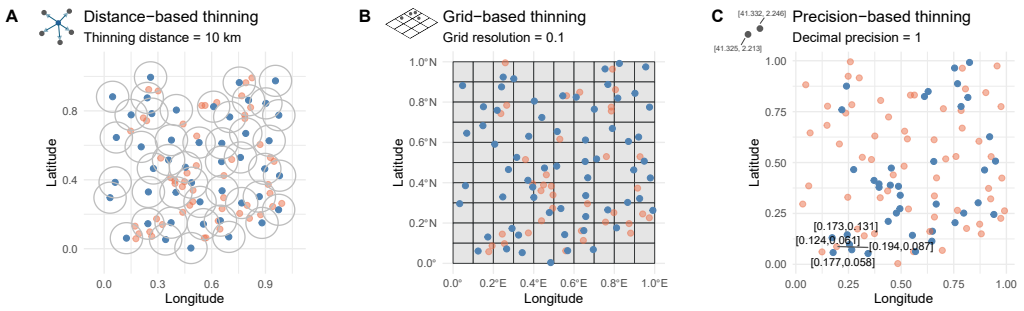

Figure 1: Spatial thinning examples using the three methods implemented in GeoThinneR: (A) distance-based thinning with a thinning distance of 10 km, (B) grid-based thinning with a grid resolution of 0.1 degrees and retaining one point per grid cell, and (C) precision-based thinning with coordinates rounded to one decimal place. Blue points indicate retained points, and red points indicate removed points.

given set of points $\mathbf{S}$, the algorithm iteratively removes points with neighbours within d based on either Haversine (geographic coordinates) or Euclidean (Cartesian coordinates) distance. By default, **GeoThinneR** uses the Haversine formula (Chopde and Nichat, 2013) to compute great-circle distances:

$$D(\mathbf{x}_i, \mathbf{x}_j) = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\text{lat}_j - \text{lat}_i}{2}\right) + \cos(\text{lat}_j) \cdot \cos(\text{lat}_i) \cdot \sin^2\left(\frac{\text{lon}_j - \text{lon}_i}{2}\right)}\right), \quad (1)$$

where R is the radius of the Earth (by default, 6371 km), and (lon, lat) are the longitude and latitude coordinates of $\mathbf{x}_i$ and $\mathbf{x}_j$. The Haversine formula assumes a smooth spherical Earth, providing sufficient accuracy for most spatial analyses (Maria et al., 2020). For example, to obtain points separated at least by 10 km we could use a similar code as before:

```
# Distance-based thinning
dist_thin <- thin_points(sim_data, method = "distance", thin_dist = 10, seed = 8237)
```

Grid-based thinning (`method="grid"`) is an alternative that stratifies points $\mathbf{S}$ into equal-sized spatial grid cells $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, selecting a subset of points $\mathbf{S}'_{G_j} \subseteq \{\mathbf{x}_i \in \mathbf{S} : \mathbf{x}_i \in G_j\}$ from each cell $G_j \in \mathcal{G}$ (Hijmans et al., 2023). This method is computationally more efficient than distance-based thinning as it does not need to compute neighbour distances, making it suitable for large datasets where strict separation distances are unnecessary.

We can provide the grid size in kilometers (`thin_dist`), the resolution of the grid (`resolution`), or a raster object to use as a grid template (`raster_obj`).

```
# Grid-based thinning using resolution
grid_thin <- thin_points(sim_data, method = "grid", resolution = 0.1, seed = 9137)

# Using raster layer
rast_obj <- terra::rast(xmin = 0, xmax = 1, ymin = 0, ymax = 1, res = 0.1)
grid_thin <- thin_points(sim_data, method = "grid", raster_obj = rast_obj, seed = 54898)
```

Precision-based thinning (`method="precision"`) removes duplicate points by rounding coordinates to a specified decimal precision, which is a very common procedure when dealing with georeferenced data in SDM (Kindt et al., 2023). This method is particularly useful when dealing with heterogeneous datasets with varying levels of spatial precision or when reducing density without enforcing a strict minimum distance is sufficient. Since it avoids distance calculations entirely, precision-based thinning is particularly fast and scales efficiently with dataset size.

```
# Precision-based thinning
prec_thin <- thin_points(sim_data, method = "precision", precision = 1, seed = 5674)
```

The optimized algorithms described in the next section focus on distance-based thinning (`method = "distance"`), since the other methods do not rely on neighbour searches and have lower computational cost.

2.2 Optimization for large datasets

As the size of spatial datasets increases, finding exact nearest neighbours by scanning the complete set **S** and computing all pairwise distances for each query point q becomes computationally unfeasible. This makes nearest neighbour searches one of the primary bottlenecks in spatial thinning methods with exact distance thresholds. To address this, **GeoThinneR** implements four different search strategies:

- `search_type="brute"` is a greedy algorithm similar to the one implemented in **spThin** and **enmSdmX**, which calculates all pairwise distances between points using the **fields** package (Douglas Nychka et al., 2021). While being the most straightforward approach, it scales poorly with large datasets due to its $O(n^2)$ complexity.
- `search_type="kd_tree"` uses *kd*-tree structures from the **nabor** package (Elseberg et al., 2012), which were proposed to theoretically reduce the time complexity of nearest-neighbour searches to $O(\log n)$ (Friedman et al., 1975). Although *kd*-trees can significantly improve performance for large datasets, they may suffer from the "curse of dimensionality" or in cases where exhaustive searches are required to identify exact nearest neighbours, their performance can be equivalent to or worse than brute-force algorithms (Ram and Sinha, 2019).
- `search_type="local_kd_tree"` subdivides the spatial domain into multiple subregions, each with an independent and smaller *kd*-tree. This avoids creating a single large *kd*-tree where lots of queries have to be evaluated and reduces both run time and memory usage for large datasets by building smaller and more manageable search trees.
- `search_type="k_estimation"` is an alternative approach to improve *kd*-tree performance and scalability. When searching for nearest-neighbours within a given distance, the exact number of neighbours k per query point q is unknown, so the algorithm evaluates whether all points in the tree lie within the search radius bound from q. To reduce the number of searches performed in the tree for each query point q, we propose a method to estimate the maximum possible number of neighbours $k_{\max}$. This estimation reduces the number of searches per point from n to $k_{\max}$, reducing unnecessary computations of distant points while still finding the exact number of neighbours.

These optimizations significantly improve computational efficiency, making **GeoThinneR** suitable for large-scale SDM applications. The underlying algorithms and their performance trade-offs are further detailed in Section 3.

2.3 Additional functionalities

In addition to its core thinning methods, **GeoThinneR** includes several functionalities designed for SDM workflows that users may require when running their analysis. Firstly, **GeoThinneR** allows performing group-specific thinning within a single dataset. This is useful when working with datasets containing multiple groups, such as different species or presence/absence data, where thinning needs to be applied independently within each group (Figure 2a).

```
# Thinning by group
precision_thin_group <- thin_points(sim_data,
  method = "precision", precision = 1,
  group_col = "group", seed = 2948
)
```

Secondly, in some cases, users may need to retain a fixed number of points while maintaining spatial representativeness, such as for class balancing in presence/absence datasets. **GeoThinneR** allows users to specify a target number of retained points and attempts to retain the specified number of points while guaranteeing a minimum distance threshold between them (Figure 2b). The process for retaining points selects the most spatially distant points to maximize geographical representativeness. This approach is supported only via the brute-force method as it requires all pairwise distance computations to select the most distant points.

```
# Thinning target points
target_thin <- thin_points(sim_data, target_points = 20, thin_dist = 10, seed = 675)
```

Finally, when applying the thinning algorithm and deciding which points to remove and retain, users can specify variables that influence which points are preferentially retained, making it possible to prioritize records based on ecological relevance, data quality, or uncertainty (Figure 2c). The thinning algorithm follows the standard procedure, but when multiple candidate points violate the thinning constraint, such as falling within the same grid cell or minimum distance threshold, the algorithm

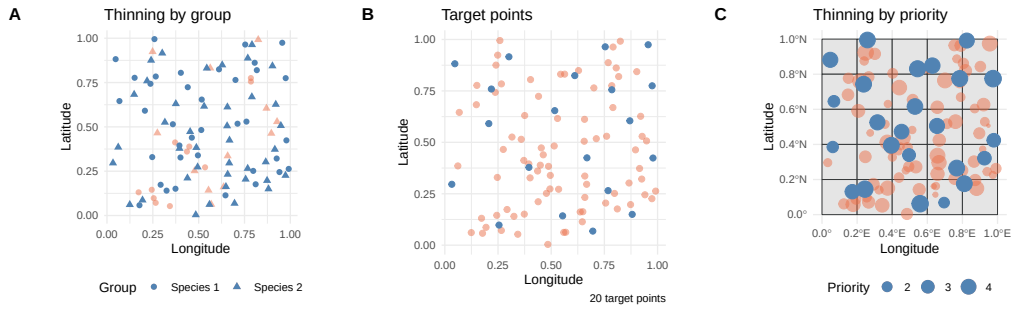

Figure 2: Additional spatial thinning functionalities available in GeoThinneR: (A) group-wise thinning applied independently for each species, (B) retaining a fixed number of points (20 points in this example) while ensuring spatial separation, and (C) prioritizing points based on larger variable values within each grid cell. Retained points are shown in blue, and removed points are in red.

retains the point with the highest priority value. This gives users more control over the characteristics of the retained dataset. For example, heterogeneous datasets from sources such as GBIF often include records of varying spatial precision. In such cases, prioritization allows users to preferentially retain higher-quality records instead of selecting points at random.

```
# Thinning priority
sim_data$priority <- runif(100, 1, 5)
rast_obj <- terra::rast(xmin = 0, xmax = 1, ymin = 0, ymax = 1, res = 0.2)
priority_thin <- thin_points(sim_data,
  method = "grid", raster_obj = rast_obj,
  priority = sim_data$priority, seed = 3455
)
```

3 Modified distance-based thinning algorithm based on *kd*-tree

The distance-based thinning process can be conceptualized as a two-step procedure: (1) identifying neighbouring points within a given distance threshold, and (2) selectively removing points to meet the selection criteria. Current methods in SDM workflows use greedy approaches in the first step, computing all pairwise distances between points and resulting in a time complexity of $O(n^2)$ and high memory usage. **GeoThinneR**, like **spThin** (Aiello-Lammens et al., 2015) and **enmSdmX** (Smith et al., 2023), implements a brute-force method using the `rdist.earth()` function from the **fields** package (Douglas Nychka et al., 2021). To improve performance, we propose two minor modifications to *kd*-tree nearest-neighbour search algorithms that improve scalability for large datasets (Figure 3). These methods reduce computational complexity while ensuring exact thinning distances, so they do not change which points are retained.

3.1 Traditional *kd*-tree nearest neighbour search

A *kd*-tree (Friedman et al., 1975) is a data structure that organizes points in a k-dimensional space by recursively partitioning the data along hyperplanes creating a binary search tree. Each node in the *kd*-tree represents a data point in the multidimensional space, and internal (non-leaf) nodes define the splitting hyperplanes that divide the space into two subsets of points based on their position relative to the split (Ram and Sinha, 2019). Many *kd*-tree variants exist, but the essence is that they recursively partition the space containing the set of points $\mathbf{S}$ into hyper-rectangles that contain a small subset $\mathbf{S}'_i \subset \mathbf{S}$ of the input points (Panigrahy, 2008).

GeoThinneR implements *kd*-tree construction and nearest-neighbour searches using the **nabor** package, which wraps the `libnabo` library (Elseberg et al., 2012). Since these *kd*-trees are built on Euclidean distances, geographic coordinates (latitude and longitude) are transformed into Cartesian coordinates (x, y, z) before constructing the binary tree:

$$\begin{aligned} x &= R \cdot \cos(\text{lat}) \cdot \cos(\text{lon}) \\ y &= R \cdot \cos(\text{lat}) \cdot \sin(\text{lon}) \\ z &= R \cdot \sin(\text{lat}), \end{aligned} \tag{2}$$

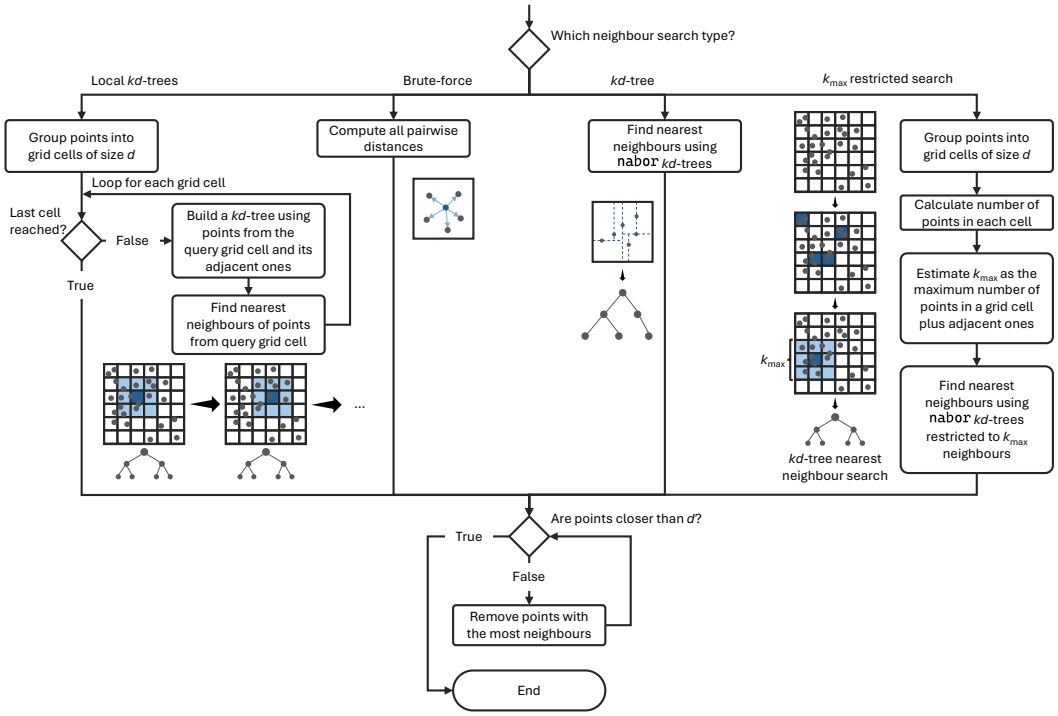

Figure 3: Diagram illustrating the four nearest neighbour search strategies implemented in GeoThinneR for distance-based thinning. The methods include brute-force, standard *kd*-trees, local *kd*-trees, and restricted *kd*-trees using estimated k_{max}. Each approach computes neighbouring points within a thinning distance before proceeding with point removal.

where lat and lon are the latitude and longitude in radians, respectively ($\frac{\pi}{180}(\text{lon}, \text{lat})$), and R is the Earth's radius. This results in a three-dimensional *kd*-tree.

However, performance may be worsened in higher dimensions or when many points must be evaluated as potential neighbours of the query point q. The time required to find the neighbours of q is affected by the search for candidate neighbours and the time it takes to evaluate among all candidates which ones are actually neighbours (Ram and Sinha, 2019). While approximate searches can reduce computational time by tuning the precision-recall trade-off, exact searches in large datasets require exhaustive searches, leading to higher computational complexity. To improve scalability, we introduce two additional optimizations: local *kd*-trees partitioning and restricted neighbour searches.

3.2 Local *kd*-trees for scalable searches

To reduce the complexity of searching within a single *kd*-tree constructed for the entire set of points $\mathbf{S}$, which requires evaluating all points to identify neighbours within the distance threshold, we propose constructing local *kd*-trees by subdividing the spatial domain into multiple smaller regions, each with its own tree. Although creating multiple *kd*-trees is time-consuming, memory usage and overall runtime are improved for large datasets, as neighbour searches are less exhaustive when evaluating smaller subsets $\mathbf{S}'_i \subset \mathbf{S}$.

To find exact neighbours using this approach, we first divide the spatial domain into a grid with cell sizes equal to the thinning distance and assign each point to a grid cell. Then, for each grid cell, we identify points within that cell and its adjacent cells, as these are potential neighbours. Recursively, an independent *kd*-tree is constructed for each grid cell using the **nabor** package, and neighbours are identified for each query point within the cell.

This strategy maintains computational efficiency while avoiding memory overloads of constructing and querying large trees. Furthermore, **GeoThinneR** uses parallel processing when `n_cores` is set to a value greater than one, providing a straightforward way to scale thinning operations across large datasets.

3.3 Restricted neighbour searches

When performing the nearest-neighbour searches using *kd*-trees from **nabor**, we have to specify the number of neighbours to find k which increases or decreases the search complexity. As the number of potential neighbours for a point q is unknown, we set k equal to the total number of points n in the dataset, ensuring that all points are checked to determine whether they are neighbours of the query point. However, for large datasets, this approach introduces unnecessary computational costs by evaluating distant points. Instead, we propose an adaptive neighbour search that estimates $k_{\max}$ based on local point densities to reduce the number of nearest neighbours to find for each query point.

The process involves dividing the spatial domain into grid cells of size equal to the thinning distance d, then counting the number of points in each grid cell and identifying the densest cells. The maximum number of neighbours a point may have $k_{\max}$ is estimated as the maximum number of points found within any dense cell and its adjacent cells. This value provides a practical upper bound on the neighbour search and is used to limit the number of points evaluated during the *kd*-tree query, reducing unnecessary searches while still ensuring exact neighbour identification.

This method significantly reduces computational time, particularly in datasets where the maximum number of neighbours per point is much smaller than the total number of points. However, one limitation is that densely populated areas or large thinning distances may inflate $k_{\max}$, making the method less effective. In such cases, the local *kd*-tree method with parallelization provides a suitable alternative.

3.4 Performance comparison of neighbour search methods

In this section, we evaluate the computational efficiency of the different neighbour search methods implemented in **GeoThinneR** using simulated datasets of varying sizes and spatial structures. All tests were conducted using R 4.3.3 on a Windows 11 computer with 128 GB of RAM and an Intel(R) Xeon(R) Gold 6240R processor featuring 24 cores, 48 threads, and a base clock speed of 2.40 GHz. The benchmark code is available in the supplementary R files.

We simulated datasets within a unit square domain $[0,1] \times [0,1]$, ranging from 1,000 to 50,000 points, under three spatial processes to mimic different real-world scenarios: (i) a homogeneous Poisson process simulating complete spatial randomly distributed points, (ii) a Matern clustered point process generated using the `rMatClust` function from the **spatstat** package (Baddeley et al., 2015) with 10 cluster centers (`kappa`), a cluster radius of 0.15 (`scale`), and a varying number of points per cluster (`mu`), and (iii) a mixed spatial pattern dataset combining the randomly distributed points and the clustered processes. Finally, we benchmarked the methods using two thinning distances (1 km and 10 km) to highlight the impact of increasing the number of points that need to be removed and how each method scales with different search complexities.

We compared the neighbour search methods in **GeoThinneR** using execution time (seconds) and peak RAM usage (MB) to evaluate the computational speed and memory requirements, which are critical for large-scale spatial datasets. Each method was tested three times, and the mean values were calculated.

The two modified algorithms ($k_{\max}$ estimation and local *kd*-trees) consistently outperform the brute-force and traditional *kd*-tree methods in both peak memory usage (Figure 4a) and execution time (Figure 4b) across all spatial data types. The traditional *kd*-tree method provides better performance compared to the brute-force method for small datasets; however, for larger datasets, memory usage and execution time increase due to exhaustive searches in deeper trees to find all neighbours of each point. However, the two optimized methods are orders of magnitude faster and less memory-intensive. For small thinning distances, the $k_{\max}$ estimation algorithm shows the best performance in terms of both time and memory usage, because the estimated number of neighbours $k_{\max}$ remains low, reducing unnecessary searches. However, at larger thinning distances, and particularly for the clustered and mixed datasets, as the dataset size increases, this algorithm becomes slower and uses more memory than the local *kd*-trees approach. This is because the algorithm estimates a high value for $k_{\max}$ and, therefore, increases search complexity.

Alternatively, the local *kd*-tree approach, which generates an independent *kd*-tree for each subregion, offers improved memory efficiency and outperforms the other methods as thinning distances increase, as fewer trees need to be built. This method can also be executed in parallel, which can further improve run time for large datasets, especially when many partitions are created due to a small thinning distance or a large spatial extent (Figure 4c). However, if the number of partitions is not very large, as in the case of the 1 km thinning distance in this example, parallel execution may not bring any improvement.

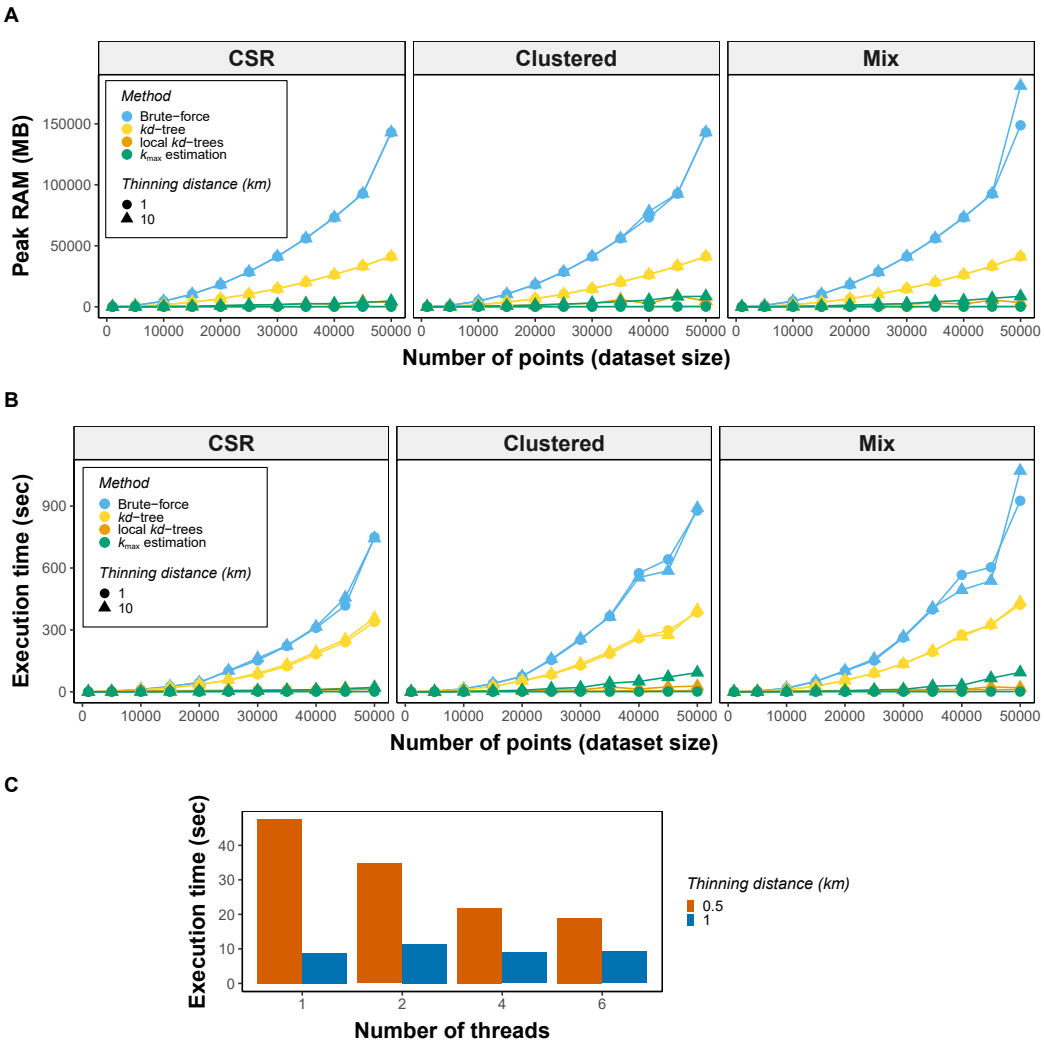

Figure 4: Performance benchmark for neighbour searches implemented in GeoThinneR distance-based thinning. (A) Peak RAM usage (MB) and (B) execution time (seconds) across three spatial data types (CSR, clustered, and mixed spatial pattern) for dataset sizes ranging from 1,000 to 50,000 points. (C) Performance comparison of local *kd*-trees with parallelization for 50,000 randomly distributed points using two thinning distances (0.5 km and 1 km). The optimized algorithms (local *kd*-trees and *k*-estimation) scale better and provide superior computational efficiency than brute-force and standard *kd*-tree, especially for large datasets.

4 Performance benchmark

In this section, we compare the performance of **GeoThinneR** with two widely used R packages for spatial thinning: **spThin** and **enmSdmX**. We compare the execution time and memory usage of the distance-based thinning algorithms available in **GeoThinneR** version 2.0.0 against the `thin()` function from **spThin** version 0.2.0 and the `geoThin()` function from **enmSdmX** version 1.2.10. In Section 4.1, we compare the performance using simulated data, and in Section 4.2, we benchmark the methods using a real-world large dataset.

```
brute <- thin_points(data,
  method = "distance", search_type = "brute",
  thin_dist = thin_dist, trials = trials
)
kd_tree <- thin_points(data,
  method = "distance", search_type = "kd_tree",
  thin_dist = thin_dist, trials = trials
)
local_kd_tree <- thin_points(data,
  method = "distance", search_type = "local_kd_tree",
  thin_dist = thin_dist, trials = trials
)
k_estimation <- thin_points(data,
  method = "distance", search_type = "k_estimation",
  thin_dist = thin_dist, trials = trials
)
spThin <- thin(data,
  lat.col = "lat", long.col = "lon", spec.col = "group",
  thin.par = thin_dist, reps = trials, locs.thinned.list.return = TRUE,
  write.files = FALSE, write.log.file = FALSE, verbose = FALSE
)
enmSdmX <- geoThin(data,
  minDist = thin_dist * 1000, longLat = c("lon", "lat"), method = "complete"
)
```

4.1 Benchmark with simulated data

The benchmark tests were performed on the complete thinning workflow using the simulated datasets described in Section 3.4. Each thinning method was executed with a single trial to obtain the maximum number of retained points per run, and each test was repeated three times to ensure reliable timing estimates. Here, we compared all methods across three spatial data types (random, clustered, and mixed) using again two thinning distances of 1 and 10 kilometers.

The optimized algorithms implemented in **GeoThinneR** demonstrate significant improvements in both memory efficiency (Figure 5a) and execution time (Figure 5b) compared to existing tools. The **enmSdmX** package exhibited the longest execution times, making it unsuitable for large datasets, so we limited its tests to 20,000 points. **spThin**, although it's more efficient, shows similar performance to the brute-force method from **GeoThinneR**. In contrast, our new modified methods demonstrate better scalability in both runtime and peak memory usage, making them more practical for large-scale spatial thinning.

The local *kd*-tree and $k_{\max}$ estimation algorithms in **GeoThinneR** present the best scalability for large datasets. The patterns are similar to those observed when comparing the neighbour search methods. The $k_{\max}$ estimation method is best suited for small thinning distances and for datasets without relatively dense regions, while the local *kd*-tree method offers a good alternative for highly clustered datasets or large thinning distances. This flexibility ensures that **GeoThinneR** can adapt to various real-world SDM workflows more efficiently than existing tools.

4.2 Application to real-world data

To illustrate the performance of **GeoThinneR** on real-world datasets, we applied its optimized thinning methods and compared them with **spThin**. We used a small dataset of 8,340 occurrence records for the loggerhead sea turtle (*Caretta caretta*) in the Mediterranean Sea (GBIF.org, 2025b), and a large and highly clustered dataset of 80,163 points for the yellowfin tuna (*Thunnus albacares*) collected globally from 1950 to 2025 (GBIF.org, 2025c). Both datasets were downloaded from GBIF (GBIF.org, 2025a) and

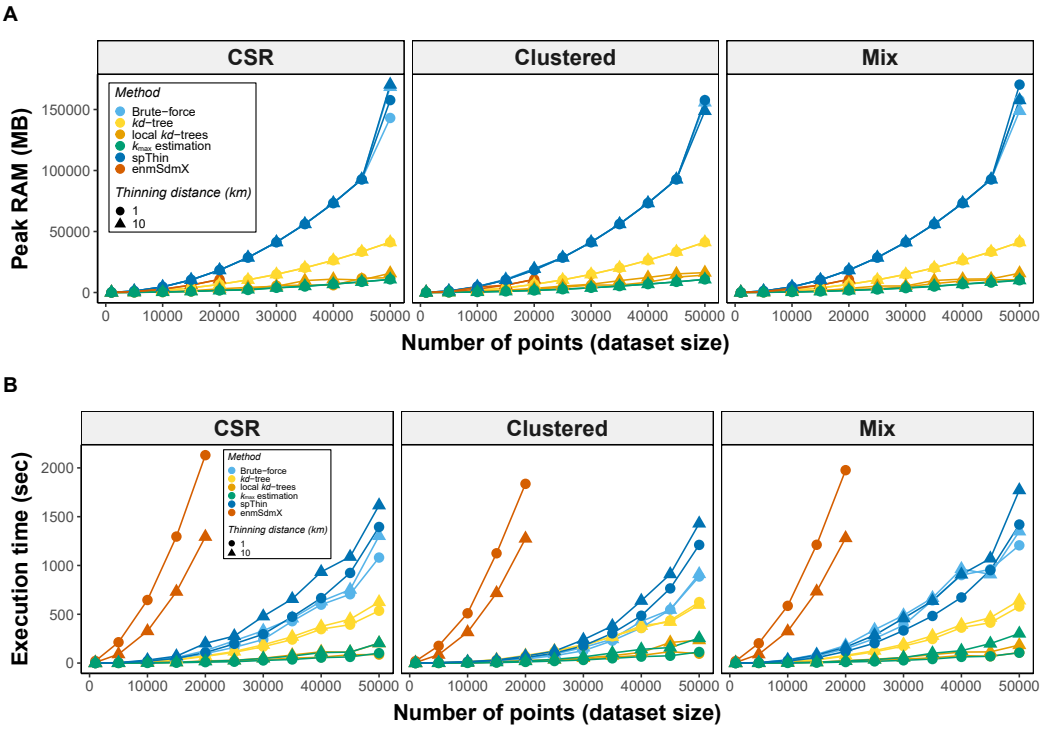

Figure 5: Performance benchmark for distance-based thinning across GeoThinneR, spThin, and enmSdmX. (A) Peak RAM usage (MB) and (B) execution time (seconds) across three spatial data types for varying dataset sizes. The enmSdmX method was only evaluated up to 20,000 points due to its high computational cost on larger datasets. The optimized methods of GeoThinneR (local *kd*-trees and *k*-estimation) offer better memory and runtime performance across all settings.

are heterogeneous, including data from different sources, locations, and sampling methods. The data are available in the **GeoThinneR** package and can be loaded using the `data()` function.

```
# Loggerhead sea turtle
data(caretta)
# Yellowfin tuna
data(thunnus)
```

Here we show the performance of the local *kd*-tree and $k_{\max}$ estimation algorithms implemented in **GeoThinneR**, comparing their computational performance against **spThin** using varying thinning distances. The objective is to show the trade-offs, best- and worst-case scenarios for each method, that's why we use varying thinning distances and dataset sizes. So we are not trying to identify the optimal thinning distance for SDM applications, this comparison highlights the scalability and efficiency of the algorithms. It is important to note that choosing an appropriate thinning distance for SDM is challenging and often requires being tuned empirically (Veloz, 2009; Soley-Guardia et al., 2019).

The results in Figure 6 show significant performance differences even for the smaller dataset. **GeoThinneR** not only executed faster but also demonstrated substantially lower memory usage, addressing one of the key limitations of current spatial thinning methods in SDM (Figure 6a). The $k_{\max}$ estimation algorithm was particularly efficient, with run times between one and two seconds due to low neighbour density that minimizes search complexity.

In contrast, the large yellowfin tuna dataset presented a different scenario (Figure 6b). **spThin** failed to execute due to excessive memory demands, surpassing our 128 GB RAM limit after several minutes, which highlights the scalability challenges of brute-force methods. Both algorithms in **GeoThinneR** maintained low memory usage, even at this dataset size. However, the $k_{\max}$ estimation method experienced slower performance as thinning distances increased because the dataset presents highly clustered areas which inflate $k_{\max}$ estimates and increase search complexity. To address this, the local *kd*-tree approach, executed here with multithreading (6 threads), proved both faster and less memory-intensive, being the method with best performance in this context. These results illustrate that while $k_{\max}$ estimation is highly efficient for moderately sized or uniformly distributed datasets,

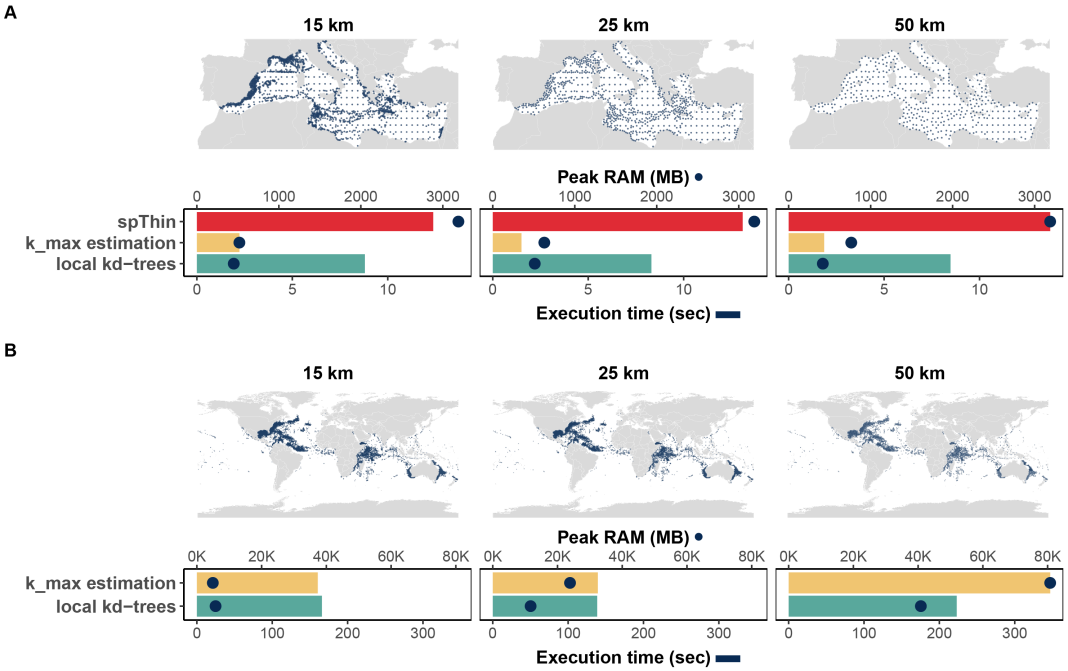

Figure 6: Execution time (bars) and memory usage (points) of GeoThinneR and spThin across varying thinning distances for occurrence data of (A) loggerhead turtles and (B) yellowfin tuna from GBIF. For the yellowfin tuna dataset, the results for spThin are not shown because the run could not be completed due to excessive memory usage.

the local *kd*-tree method with parallelization offers a memory-efficient solution for large and dense datasets.

Here we present the code used to generate the thinned datasets shown in Figure 6:

```
caretta_thin <- thin_points(
  caretta,
  lon_col = "decimalLongitude",
  lat_col = "decimalLatitude",
  method = "distance",
  thin_dist = thin_dist, # One of 10, 25, 50
  search_type = "k_estimation",
  trials = 10
)

thunnus_thin <- thin_points(
  thunnus,
  lon_col = "decimalLongitude",
  lat_col = "decimalLatitude",
  method = "distance",
  thin_dist = thin_dist, # One of 10, 25, 50
  search_type = "local_kd_tree",
  trials = 1,
  n_cores = 6
)
```

5 Conclusions

The GeoThinneR R package offers a flexible and scalable implementation of spatial thinning methods, addressing key limitations of existing tools. It integrates multiple thinning approaches (distance-, grid-, and precision-based) into a single wrapper function, enhancing usability. By incorporating optimized *kd*-tree algorithms for distance-based thinning, such as local *kd*-tree partitioning and adaptive $k_{\max}$ estimation, the package significantly reduces execution time and memory usage for

large datasets. Additionally, **GeoThinneR** includes customization features designed for species distribution modelling workflows, such as group-wise thinning and point prioritization. While future developments should focus on improving the computational performance of distance-based methods, incorporating additional thinning approaches, and estimating an optimal thinning distance, **GeoThinneR** represents a step forward in improving the efficiency of spatial thinning methods in SDM.

6 Acknowledgments

This package has been developed as part of the ProOceans (PID2020-118097RB-I00) and SOSPen (PID2021-124831OA-I00) projects funded by the Spanish Ministry of Science and Innovation. The author also acknowledges the support from the MOIRAI project (grant agreement no. 101180994) funded by the European Union under the Horizon Europe program. The author acknowledges the institutional support of the "Severo Ochoa Center of Excellence" accreditation (CEX2024-001494-S funded by AEI 10.13039/501100011033) to the Institute of Marine Science (ICM-CSIC). Additionally, this work is part of the Integrated Marine Ecosystem Assessments (iMARES) research group, funded by the Agència de Gestió d'Ajuts Universitaris i de Recerca of the Generalitat de Catalunya (2021 SGR 00435).

7 Supplementary materials

Supplementary materials are available in addition to this article:

- *GeoThinneR_replication_small.R*: reproduces examples and figures from Sections 2 and 3. Includes a short benchmark and demonstrations of package features using small datasets.
- *GeoThinneR_replication_large.R*: reproduces figures and results from Section 4. Includes a larger benchmark to evaluate performance of GeoThinneR and other packages using simulated and real-world datasets.
- *caretta_download.R*: downloads occurrence data for loggerhead turtles (*Caretta caretta*) from GBIF.
- *thunnus_download.R*: downloads occurrence data for yellowfin tuna (*Thunnus albacares*) from GBIF.
- *data/*: contains benchmark results and the cleaned loggerhead turtle and yellowfin tuna datasets used.

References

M. E. Aiello-Lammens, R. A. Boria, A. Radosavljevic, B. Vilela, and R. P. Anderson. spThin: An R package for spatial thinning of species occurrence records for use in ecological niche models. *Ecography*, 38(5):541–545, 2015. doi: 10.1111/ecog.01132. [p299, 301, 304]

A. V. R. Amaral, E. T. Krainski, R. Zhong, and P. Moraga. Model-based geostatistics under spatially varying preferential sampling. *Journal of Agricultural, Biological and Environmental Statistics*, 29(4): 766–792, 2024. doi: 10.1007/s13253-023-00571-0. [p299]

A. Baddeley, E. Rubak, and R. Turner. *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press, London, 2015. ISBN 9781482210200. URL https://www.routledge.com/Spatial-Point-Patterns-Methodology-and-Applications-with-R/Baddeley-Rubak-Turner/p/book/9781482210200/. [p306]

D. J. Baker, I. M. Maclean, and K. J. Gaston. Effective strategies for correcting spatial sampling bias in species distribution models without independent test data. *Diversity and Distributions*, 30(3):e13802, 2024. doi: 10.1111/ddi.13802. [p299]

R. A. Barber, S. G. Ball, R. K. Morris, and F. Gilbert. Target-group backgrounds prove effective at correcting sampling bias in Maxent models. *Diversity and Distributions*, 28(1):128–141, 2022. doi: 10.1111/ddi.13442. [p299]

J. Beck, M. Böller, A. Erhardt, and W. Schwanghart. Spatial bias in the GBIF database and its effect on modeling species' geographic distributions. *Ecological Informatics*, 19:10–15, 2014. doi: 10.1016/j.ecoinf.2013.11.002. [p299]

E. H. Boakes, P. J. McGowan, R. A. Fuller, D. Chang-qing, N. E. Clark, K. O'Connor, and G. M. Mace. Distorted views of biodiversity: Spatial and temporal bias in species occurrence data. *PLoS Biology*, 8(6):e1000385, 2010. doi: 10.1371/journal.pbio.1000385. [p299]

R. A. Boria, L. E. Olson, S. M. Goodman, and R. P. Anderson. Spatial filtering to reduce sampling bias can improve the performance of ecological niche models. *Ecological Modelling*, 275:73–77, 2014. doi: 10.1016/j.ecolmodel.2013.12.012. [p299]

J. M. Calabrese, G. Certain, C. Kraan, and C. F. Dormann. Stacking species distribution models and adjusting bias by linking them to macroecological models. *Global Ecology and Biogeography*, 23(1): 99–112, 2014. doi: 10.1111/geb.12102. [p300]

A. A. Castellanos, J. W. Huntley, G. Voelker, and A. M. Lawing. Environmental filtering improves ecological niche models across multiple scales. *Methods in Ecology and Evolution*, 10(4):481–492, 2019. doi: 10.1111/2041-210X.13142. [p299]

Y. Chauvier, N. E. Zimmermann, G. Poggiato, D. Bystrova, P. Brun, and W. Thuiller. Novel methods to correct for observer and sampling bias in presence-only species distribution models. *Global Ecology and Biogeography*, 30(11):2312–2325, 2021. doi: 10.1111/geb.13383. [p299]

N. R. Chopde and M. Nichat. Landmark based shortest path detection by using A* and Haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1 (2):298–302, 2013. URL https://scispace.com/pdf/landmark-based-shortest-path-detection-byusing-a-and-2nsrbowy4m.pdf. [p302]

F. Cosentino and L. Maiorano. Is geographic sampling bias representative of environmental space? *Ecological Informatics*, 64:101369, 2021. doi: 10.1016/j.ecoinf.2021.101369. [p299]

P. J. Diggle, R. Menezes, and T.-l. Su. Geostatistical inference under preferential sampling. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 59(2):191–232, 2010. doi: 10.1111/j.1467-9876.2009.00701.x. [p299]

Douglas Nychka, Reinhard Furrer, John Paige, and Stephan Sain. fields: Tools for spatial data, 2021. URL https://github.com/dnychka/fieldsRPackage. R package version 16.3. [p303, 304]

J. Elith and J. R. Leathwick. Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, 40(1):677–697, 2009. doi: 10.1146/annurev.ecolsys.110308.120159. [p299]

J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics (JOSER)*, 3(1):2–12, 2012. ISSN 2035-3928. URL https://robotik.informatik.uni-wuerzburg.de/telematics/download/joser2012.pdf. [p303, 304]

J. H. Friedman, J. L. Bentley, and R. A. Finkel. *An Algorithm for Finding Best Matches in Logarithmic Time*. Department of Computer Science, Stanford University, 1975. doi: 10.1145/355744.355745. [p300, 303, 304]

GBIF.org. GBIF Home Page, 2025a. URL https://www.gbif.org/. [26 November 2025]. [p308]

GBIF.org. GBIF Occurrence Download. 2025b. doi: 10.15468/dl.9jcjrm. [p308]

GBIF.org. GBIF Occurrence Download. 2025c. doi: 10.15468/dl.xsyrkh. [p308]

A. Guisan, R. Tingley, J. B. Baumgartner, I. Naujokaitis-Lewis, P. R. Sutcliffe, A. I. Tulloch, T. J. Regan, L. Brotons, E. McDonald-Madden, C. Mantyka-Pringle, et al. Predicting species distributions for conservation decisions. *Ecology Letters*, 16(12):1424–1435, 2013. doi: 10.1111/ele.12189. [p299]

M. Higa, Y. Yamaura, I. Koizumi, Y. Yabuhara, M. Senzaki, and S. Ono. Mapping large-scale bird distributions using occupancy models and citizen data with spatially biased sampling effort. *Diversity and Distributions*, 21(1):46–54, 2015. doi: 10.1111/ddi.12255. [p299]

R. J. Hijmans, S. Phillips, J. Leathwick, and J. Elith. *dismo: Species Distribution Modeling*, 2023. URL https://github.com/rspatial/dismo. R package version 1.3-14. [p299, 302]

A. C. Hughes, M. C. Orr, K. Ma, M. J. Costello, J. Waller, P. Provoost, Q. Yang, C. Zhu, and H. Qiao. Sampling biases shape our view of the natural world. *Ecography*, 44(9):1259–1269, 2021. doi: 10.1111/ecog.05926. [p299]

R. Inman, J. Franklin, T. Esque, and K. Nussear. Comparing sample bias correction methods for species distribution modeling using virtual species. *Ecosphere*, 12(3):e03422, 2021. doi: 10.1002/ecs2.3422. [p299]

R. Kindt, A. Abiyu, P. Borchardt, I. Dawson, S. Demissew, L. Graudal, R. Jamnadass, J.-P. Lillesø, S. Moestrup, F. Pedercini, et al. *The Climate Change Atlas for Africa of Tree Species Prioritized for Forest Landscape Restoration in Ethiopia: A Description of Methods Used to Develop the Atlas*. CIFOR, 2023. doi: 10.17528/cifor-icraf/008977. [p302]

V. Koshkina, Y. Wang, A. Gordon, R. M. Dorazio, M. White, and L. Stone. Integrated species distribution models: Combining presence-background data and site-occupancy data with imperfect detection. *Methods in Ecology and Evolution*, 8(4):420–430, 2017. doi: 10.1111/2041-210X.12738. [p299]

J. M. Lobo and M. F. Tognelli. Exploring the effects of quantity and location of pseudo-absences and sampling biases on the performance of distribution models with limited point occurrence data. *Journal for Nature Conservation*, 19(1):1–7, 2011. doi: 10.1016/j.jnc.2010.03.002. [p299]

J. Mäkinen, C. Merow, and W. Jetz. Integrated species distribution models to account for sampling biases and improve range-wide occurrence predictions. *Global Ecology and Biogeography*, 33(3): 356–370, 2024. doi: 10.1111/geb.13792. [p299]

E. Maria, E. Budiman, M. Taruk, et al. Measure distance locating nearest public facilities using Haversine and euclidean methods. In *Journal of Physics: Conference Series*, volume 1450, page 012080. IOP Publishing, 2020. doi: 10.1088/1742-6596/1450/1/012080. [p302]

J. Mestre-Tomás. *GeoThinneR: Efficient Spatial Thinning of Species Occurrences*, 2025. URL https://github.com/jmestret/GeoThinneR. R package version 2.1.0. [p300]

C. Meyer, W. Jetz, R. P. Guralnick, S. A. Fritz, and H. Kreft. Range geometry and socio-economics dominate species-level biases in occurrence information. *Global Ecology and Biogeography*, 25(10): 1181–1193, 2016. doi: 10.1111/geb.12483. [p299]

J. Miller. Species distribution modeling. *Geography Compass*, 4(6):490–509, 2010. doi: 10.1111/j.1749-8198.2010.00351.x. [p299]

V. Moudrỳ, M. Bazzichetto, R. Remelgado, R. Devillers, J. Lenoir, R. G. Mateo, J. J. Lembrechts, N. Sillero, V. Lecours, A. F. Cord, et al. Optimising occurrence data in species distribution models: Sample size, positional uncertainty, and sampling bias matter. *Ecography*, 2024(12):e07294, 2024. doi: 10.1111/ecog.07294. [p299]

S. Noori, A. Hofmann, D. Rödder, M. Husemann, and H. Rajaei. A window to the future: Effects of climate change on the distribution patterns of iranian zygaenidae and their host plants. *Biodiversity and Conservation*, 33(2):579–602, 2024. doi: 10.1007/s10531-023-02760-2. [p300]

O. Ovaskainen and N. Abrego. *Joint Species Distribution Modelling: With Applications in R*. Cambridge University Press, 2020. ISBN 9781108591720. doi: 10.1017/9781108591720. [p300]

O. Ovaskainen, G. Tikhonov, A. Norberg, F. Guillaume Blanchet, L. Duan, D. Dunson, T. Roslin, and N. Abrego. How to make more out of community data? a conceptual framework and its implementation as models and software. *Ecology Letters*, 20(5):561–576, 2017. doi: 10.1111/ele.12757. [p300]

R. Panigrahy. An improved algorithm finding nearest neighbor using kd-trees. In *Latin American Symposium on Theoretical Informatics*, pages 387–398. Springer, 2008. doi: 10.1007/978-3-540-78773-0_34. [p304]

J. L. Pearce and M. S. Boyce. Modelling distribution and abundance with presence-only data. *Journal of Applied Ecology*, 43(3):405–412, 2006. doi: 10.1111/j.1365-2664.2005.01112.x. [p299]

M. G. Pennino, I. Paradinas, J. B. Illian, F. Muñoz, J. M. Bellido, A. López-Quílez, and D. Conesa. Accounting for preferential sampling in species distribution models. *Ecology and Evolution*, 9(1): 653–663, 2019. doi: 10.1002/ece3.4789. [p299]

S. J. Phillips, M. Dudík, J. Elith, C. H. Graham, A. Lehmann, J. Leathwick, and S. Ferrier. Sample selection bias and presence-only distribution models: Implications for background and pseudo-absence data. *Ecological Applications*, 19(1):181–197, 2009. doi: 10.1890/07-2153.1. [p299]

L. J. Pollock, R. Tingley, W. K. Morris, N. Golding, R. B. O'Hara, K. M. Parris, P. A. Vesk, and M. A. McCarthy. Understanding co-occurrence by modelling species simultaneously with a joint species distribution model (JSDM). *Methods in Ecology and Evolution*, 5(5):397–406, 2014. doi: 10.1111/2041-210X.12180. [p300]

H. R. Pulliam. On the relationship between niche and distribution. *Ecology Letters*, 3(4):349–361, 2000. doi: 10.1046/j.1461-0248.2000.00143.x. [p299]

P. Ram and K. Sinha. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1378–1388, 2019. doi: 10.1145/3292500.3330875. [p303, 304, 305]

E. G. Simmonds, S. G. Jarvis, P. A. Henrys, N. J. Isaac, and R. B. O'Hara. Is more data always better? a simulation study of benefits and limitations of integrated distribution models. *Ecography*, 43(10): 1413–1422, 2020. doi: 10.1111/ecog.05146. [p299]

A. B. Smith, S. J. Murphy, D. Henderson, and K. D. Erickson. Including imprecisely georeferenced specimens improves accuracy of species distribution models and estimates of niche breadth. *Global Ecology and Biogeography*, 32(3):342–355, 2023. doi: 10.1111/geb.13628. [p299, 304]

M. Soley-Guardia, A. C. Carnaval, and R. P. Anderson. Sufficient versus optimal climatic stability during the late quaternary: Using environmental quality to guide phylogeographic inferences in a neotropical montane system. *Journal of Mammalogy*, 100(6):1783–1807, 2019. doi: 10.1093/jmammal/gyz162. [p309]

V. A. Steen, M. W. Tingley, P. W. Paton, and C. S. Elphick. Spatial thinning and class balancing: Key choices lead to variation in the performance of species distribution models with citizen science data. *Methods in Ecology and Evolution*, 12(2):216–226, 2021. doi: 10.1111/2041-210X.13525. [p299, 300]

S. Varela, R. P. Anderson, R. García-Valdés, and F. Fernández-González. Environmental filters reduce the effects of sampling bias and improve predictions of ecological niche models. *Ecography*, 37(11): 1084–1091, 2014. doi: 10.1111/j.1600-0587.2013.00441.x. [p299]

S. D. Veloz. Spatially autocorrelated sampling falsely inflates measures of accuracy for presence-only niche models. *Journal of Biogeography*, 36(12):2290–2299, 2009. doi: 10.1111/j.1365-2699.2009.02174.x. [p299, 309]

J. Vollering, R. Halvorsen, I. Auestad, and K. Rydgren. Bunching up the background betters bias in species distribution models. *Ecography*, 42(10):1717–1727, 2019. doi: 10.1111/ecog.04503. [p299]

M. S. Wisz, R. Hijmans, J. Li, A. T. Peterson, C. Graham, A. Guisan, and NCEAS Predicting Species Distributions Working Group. Effects of sample size on the performance of species distribution models. *Diversity and Distributions*, 14(5):763–773, 2008. doi: 10.1111/j.1472-4642.2008.00482.x. [p299]

H. Yu, T. Wang, A. Skidmore, M. Heurich, and C. Bässler. How future climate and tree distribution changes shape the biodiversity of macrofungi across europe. *Diversity and Distributions*, 29(5): 666–682, 2023. doi: 10.1111/ddi.13688. [p300]

Jorge Mestre-Tomás
Renewable Marine Resources Department, Institut de Ciències del Mar (ICM-CSIC), Barcelona, SpainDepartment of Applied Statistics and Operational Research, and Quality, Universitat Politècnica de València (UPV), Valencia, Spain

https://github.com/jmestret
ORCiD: 0000-0002-8983-3417
jorgemestre@icm.csic.es