Package 'bean'

July 28, 2025

Type Package

Title Data Thinning of Species Occurrences in Environmental Space

Version 0.1.2

Maintainer Paanwaris Paansri <paanwaris@vt.edu>

Description Provides a suite of tools to mitigate sampling bias in species occurrence records by thinning data in the environmental space (E-space). This process could help increase accuracy and precision in species distribution modeling (SDM, or ecological niche modeling, ENM). The package offers a data-driven protocol to determine thinning parameters. Thinning methods (stochastic and deterministic) help reduce oversampled conditions and outlier observations. The name 'bean' reflects the core principle of the method: ensuring that each "pod" (a grid cell in E-space) contains only a specified number of "beans" (occurrence points).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/paanwaris/bean

BugReports https://github.com/paanwaris/bean/issues

RoxygenNote 7.3.2

Imports
raster, terra, testthat, dismo, dplyr, ggplot2, tidyr, magrittr, rlang, stats, MASS, rJava, covr, scales

Depends R (>= 3.5)

Author Paanwaris Paansri [cre, aut] (ORCID: <https://orcid.org/0000-0001-9992-098X>)
Luis E. Escobar [ctb] (ORCID: <https://orcid.org/0000-0001-5735-2750>)

Contents

calibrate_bean 2
find_env_resolution 4
find_optimal_cap 6
fit_ellipsoid 8
plot_bean 10
prepare_bean 11
test_env_thinning 13
thin_env_center 16
thin_env_density 17

Index 19

calibrate_bean *Calibrate thinning and niche parameters using model performance*

Description

This function automates the selection of an optimal set of parameters for the entire `bean` workflow. It treats the `quantile` for resolution, the `target_percent` for thinning, and the `method` for ellipsoid fitting as model hyperparameters. It iterates through a range of these values, performs the full "prepare -> resolve -> thin -> fit" workflow, and evaluates the final model's performance using repeated k-fold cross-validation (AUC). The function identifies the combination of parameters that produces the highest mean AUC and provides statistical comparisons against a baseline model built with unthinned data.

Usage

```
calibrate_bean(
  data,
  env_vars,
  background_data,
  env_rasters,
  longitude,
  latitude,
  quantile_range = seq(0.1, 0.9, 0.1),
  method_range = c("covmat", "mve"),
  target_percent = 0.95,
  level = 0.95,
  thinning_reps = 10,
  k = 4,
  n_repeats = 10,
 maxent_args = c("linear=true", "quadratic=false", "product=false", "threshold=false",
    "hinge=false", "doclamp=true")
)
```

Arguments

`data`	A data.frame containing species occurrences and pre-scaled environmental variables, typically the output of `prepare_bean()`.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables.
`background_data`	A data frame with longitude and latitude of background points.
`env_rasters`	A RasterStack or SpatRaster object of environmental variables.
`longitude`	(character) The name of the longitude column in the data frames.
`latitude`	(character) The name of the latitude column in the data frames.
`quantile_range`	A numeric vector of quantile values to test for resolution finding.
`method_range`	A character vector of ellipsoid methods to test (e.g., "c("covmat", "mve")").
`target_percent`	(numeric) The target proportion of points to retain after density thinning.

`level`	(numeric) A single value between 0 and 1 representing the proportion of data points the ellipse is intended to encompass. Default is 0.95.
`thinning_reps`	(numeric) The number of times to repeat the stochastic thinning process for each parameter combination to get a stable performance estimate.
`k`	(numeric) The number of folds for cross-validation. Default = 4.
`n_repeats`	(numeric) The number of times to repeat the k-fold process. Default = 10.
`maxent_args`	(character) A vector of arguments for `dismo::maxent`.

Value

An object of class `bean_calibration`, which is a list containing:

`best_parameters`	A list with the optimal "quantile", "method", "resolution", and "cap".
`calibration_summary`	A tibble summarizing the performance for each tested combination, including statistical comparisons to the baseline.
`baseline_auc`	A numeric vector of the AUC scores from the model built on the original, unthinned data.
`best_points_in_ellipse`	The data frame of points inside the ellipse from the best performing model.
`best_points_outside_ellipse`	The data frame of points outside the ellipse from the best performing model.
`parameters`	A list containing the parameter ranges used in the calibration (e.g., `quantile_range`).

Note

This is a very computationally intensive function. For reproducible results, run `set.seed()` before calling this function.

Examples

```
## Not run:
# This is a long-running example and requires the "dismo" package to be installed.

# Load the package's example data
library(raster)
library(dismo)
occ_file <- system.file("extdata", "Peromyscus_maniculatus_prepared.csv", package = "bean")
occ_data <- read.csv(occ_file)

bio1_file <- system.file("extdata", "BIO1.tif", package = "bean")
bio12_file <- system.file("extdata", "BIO12.tif", package = "bean")
env_rasters <- raster::stack(bio1_file, bio12_file)
# Set seed for reproducibility
set.seed(123)

# Calibrate all key parameters
final_calibration <- calibrate_bean(
  data = occ_data,
  env_vars = c("BIO1", "BIO12"),
  background_data = background_df,
  env_rasters = env_rasters,
  longitude = "x",
  latitude = "y",
  quantile_range = seq(0.1, 0.9, 0.1),
  method_range = c("covmat", "mve"),
  target_percent = 0.95,
  level = 95,
  thinning_reps = 3, # Use a small number for the example
  k = 2,
  n_repeats = 2 # Use minimal settings for a runnable example
)

# Print the summary to see the best combination of parameters
print(final_calibration)

# Access the final, best-thinned data directly
final_data <- final_calibration$best_points_in_ellipse
head(final_data)

## End(Not run)
```

`find_env_resolution` *Find objective grid resolutions for each environmental axis*

Description

This function calculates an objective, data-driven grid resolution for use in environmental thinning. It operates by analyzing the distribution of pairwise 1D distances for each environmental variable separately. The distance at a specified lower quantile of this distribution is returned as a suggested grid resolution for each axis. The function allows for the creation of rectangular grid cells that are adapted to the unique scale and clustering within each environmental dimension.

Usage

```
find_env_resolution(data, env_vars, quantile = 0.1)
```

Arguments

`data`	A data.frame containing species occurrence coordinates and the environmental variables.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables to be used in the analysis.
`quantile`	(numeric) The quantile of pairwise distances to use for the precision. A smaller value (e.g., 0.05 or 0.1) is recommended to retain most of the original points and it represents the spacing between closely clustered points. Default = 0.1 (10th percentile). See Details.

Details

### The Rationale for Using a Distance Quantile (`quantile`)

Selecting an appropriate grid resolution for environmental thinning is crucial for reducing sampling bias without discarding excessive data. Rather than relying on an arbitrary, user-defined value, this function employs a data-driven heuristic to determine a characteristic scale from the data itself based on environmental filtering (Varela et al., 2014).

For each environmental variable, the function calculates all pairwise distances between the occurrence points. This creates a distribution that reflects the internal spacing and clustering of the data in that one dimension. By selecting a low quantile of this distribution (e.g., "quantile = 0.1" for the 10th percentile is recommended), we identify a distance that is representative of the spacing between closely clustered points. This approach is conceptually powerful because it focuses on the distributional edges to identify limiting factors or characteristic scales, rather than being limited to an analysis of the mean (Cade & Noon, 2003).

Using the grid resolution from the quantile approach is a robust strategy because it adapts the cell size to the inherent data structure. If points are tightly clustered in one dimension (e.g., a species has a narrow thermal tolerance), the resulting resolution will be fine. If points are spread out, the resolution will be coarser.

Value

An object of class `bean_resolution`, which is a list containing:

`suggested_resolution`	A named numeric vector of the calculated grid resolutions for each environmental variable.
`distance_distributions`	A data frame containing all pairwise distances for each variable, suitable for plotting.
`quantile`	The percentile used for the calculation.

References

Cade, B. S., & Noon, B. R. (2003). A gentle introduction to quantile regression for ecologists. Frontiers in Ecology and the Environment, 1(8), 412–420.

Varela, S., Anderson, R. P., García-Valdés, R., & Fernández-González, F. (2014). Environmental filters reduce the effects of sampling bias and improve predictions of ecological niche models. Ecography, 37(11), 1084–1091.

Examples

```
## Not run:
# 1. Create the example occurrence records and environmental variables
occ_data <- data.frame(
 BIO1 = c(0, 1, 3, 6),
 BIO12 = c(0, 10, 30, 60),
 x = c(1, 1, 0, 0.5),
 y = c(0, 0.5, 2, 1.5),
 species = "A"
)
# 2. Find the resolution at the 10th percentile
resolutions <- find_env_resolution(
  data = occ_data,
  env_vars = c("BIO1", "BIO12"),
  quantile = 0.1
)

# 3. Print the summary and plot the results
print(resolutions)
```

```
plot(resolutions)

## End(Not run)
```

`find_optimal_cap` *Find optimal density caps based on a target thinning percentage*

Description

This function searches for an optimal density cap by evaluating two criteria: 1) the density cap that results in an occurrence point count closest to the target percentage, and 2) the density cap that results in an occurrence point count that is closest to, but not below, the target percentage.

Usage

```
find_optimal_cap(data, env_vars, grid_resolution, target_percent = 0.95)
```

Arguments

`data`	A data.frame containing species occurrence coordinates and the environmental variables.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables to be used in the analysis.
`grid_resolution`	A numeric vector of length one or two specifying the resolution(s) for the grid axes. If length one, it is used for both axes. See Details
`target_percent`	A numeric value (0-1) for the target percentage of points to retain. A value of 0.95 is recommended to remove 5 percent of the most densely clustered points while retaining most of the data. Default = 0.95.

Details

### Defining Environmental Grid Axes

The "grid_resolution" parameter is fundamental to how this function operates. It defines the dimensions of the grid cells in the environmental space, which are used to calculate occurrence density.

A single numeric value (e.g., "grid_resolution = 0.2") will create square grid cells, applying the same resolution to both environmental axes. A numeric vector of two values (e.g., "grid_resolution = c(0.2, 0.5)") will create rectangular grid cells, applying a different resolution to each respective axis.

This function's flexibility allows you to tailor the thinning process to the specific characteristics of your environmental data.

Value

An object of class `bean_optimization`, which is a list containing:

`best_cap_closest`	The integer density cap value that results in a final count of occurrence points nearest to the target count.

`retained_points_closest`	The number of occurrence points retained after thinning with the density cap that provides the closest result.
`best_cap_above_target`	The smallest integer density cap value that retains a number of occurrence points greater than or equal to the target occurrence point count. This is often the most practical value to use.
`retained_points_above_target`	The number of occurrence points retained after thinning with the best density cap that is at or above the target.
`search_results`	A data.frame (as a tibble) containing the thinning results for every density cap value tested.
`parameters`	A list of key input parameters used in the optimization, such as the calculated target occurrence point count.

Note

It is highly recommended to first use the `find_env_resolution` function to determine an objective, data-driven resolution. This allows you to tailor the thinning process to the specific characteristics of your environmental data.

See Also

`find_env_resolution`

Examples

```
## Not run:
# 1. Create the example occurrence records and environmental variables
occ_data <- data.frame(
 BIO1 = c(0, 1, 3, 6),
 BIO12 = c(0, 10, 30, 60),
 x = c(1, 1, 0, 0.5),
 y = c(0, 0.5, 2, 1.5),
 species = "A"
)

# 2. Find optimal cap to retain ~95% of the data
set.seed(81) # For reproducibility
optimal_params <- find_optimal_cap(
  data = occ_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = c(0.1, 0.2), # Using an example resolution
  target_percent = 0.95
)

# 3. Print the summary and plot the results
print(optimal_params)
plot(optimal_params)

## End(Not run)
```

`fit_ellipsoid`	*Outlier removal*

Description

This function calculates a bivariate ellipse that encompasses a specified proportion of the data points in a 2D environmental space. It can use either a standard covariance matrix or a robust Minimum Volume Ellipsoid. The function also correctly identifies which of the input points fall within and outside the calculated ellipse boundary, preserving all original columns.

Usage

```
fit_ellipsoid(data, env_vars, method = "covmat", level = 0.95)
```

Arguments

`data`	A data.frame containing species occurrence coordinates and the environmental variables.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables to be used in the analysis.
`method`	(character) The method for calculating the centroid and covariance matrix. Options are "covmat" (for a standard covariance matrix) and "mve" (Minimum Volume Ellipsoid, robust to outliers). Default = "covmat". See Details
`level`	(numeric) A single value between 0 and 1 representing the proportion of data points the ellipse is intended to encompass. Default is 0.95.

Details

This function provides two distinct statistical approaches for defining the center and shape of the ellipse, selectable via the `method` parameter. The size of the ellipse is controlled by the `level` parameter, which defines a statistical confidence interval.

## Method

The `method` argument determines how the centroid and covariance matrix (shape and orientation) of the data cloud are calculated.

- "covmat" (Default): This is the classical approach, which uses the standard sample mean and sample covariance matrix calculated from all provided data points. While optimal for cleanly distributed, multivariate normal data, this method is highly sensitive to outliers. A single anomalous data point can significantly skew the mean and inflate the covariance matrix, resulting in an ellipse that poorly represents the central tendency of the data (Rousseeuw & Leroy, 2003).

- "mve": This option uses the Minimum Volume Ellipsoid (MVE) estimator, a robust statistical method designed to resist the influence of outliers (Rousseeuw et al., 1984, 1985). Instead of using all data points, the MVE algorithm finds the ellipsoid with the smallest possible volume that contains a specified subset of the data (at least h = (n_points + n_variables + 1)/2 points) (Cobos et al., 2024). By focusing on the most concentrated "core" of the data, the MVE method effectively ignores outliers, providing a more reliable estimate of the data's true center and scatter when contamination is present (Van Aelst & Rousseeuw, 2009).

## Confidence Level

The `level` parameter specifies the confidence level for the ellipse, representing the percentage of the data that the ellipse is intended to encompass (Cobos et al., 2024). It determines the size of the ellipse by defining a statistical boundary based on Mahalanobis distances from the centroid.

Assuming the data follows a multivariate normal distribution, the boundary of the ellipse corresponds to a quantile of the chi-squared (χ^2) distribution (Van Aelst & Rousseeuw, 2009). For example, a `level` of 95 (the default) constructs an ellipse whose boundary is defined by the set of points having a squared Mahalanobis distance equal to the 0.95 quantile of the χ^2 distribution with 2 degrees of freedom (for a 2D analysis). Points with a smaller Mahalanobis distance are inside the ellipse, while those with a larger distance are outside.

A higher `level` (e.g., 99) will result in a larger ellipse, while a lower `level` (e.g., 90) will produce a smaller, more conservative ellipse.

Value

An object of class `bean_ellipsoid`, which is a list containing:

`niche_ellipse`	A data.frame of points defining the perimeter of the calculated ellipse.
`centroid`	A named vector representing the center of the ellipse.
`covariance_matrix`	The 2x2 covariance matrix used to define the ellipse's shape.
`all_points_used`	The input data frame, filtered to include only complete, finite observations.
`points_in_ellipse`	A data.frame containing the subset of rows from `all_points_used` that fall inside the ellipse boundary.
`points_outside_ellipse`	A data.frame containing the subset of rows from `all_points_used` that fall outside the ellipse boundary.
`inside_indices`	A numeric vector of the row indices (from `all_points_used`) of the points inside the ellipse.
`parameters`	A list of the key parameters used, including `level` and `method`.

References

Rousseeuw, P. J., & Leroy, A. M. (2003). Robust regression and outlier detection. John wiley & sons.

Van Aelst, S., & Rousseeuw, P. (2009). Minimum volume ellipsoid. Wiley Interdisciplinary Reviews: Computational Statistics, 1(1), 71-82.

Cobos, M.E., Osorio-Olvera, L., Soberón, J., Peterson, A.T., Barve, V. & Barve, N. (2024) ellipsenm: ecological niche's characterizations using ellipsoids. <https://github.com/marlonecobos/ellipsenm>

Rousseeuw, P. J. (1984). Least median of squares regression. Journal of the American statistical association, 79(388), 871-880.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. Mathematical statistics and applications, 8(283-297), 37.

Examples

```
## Not run:
# 1. Create environmental data with a cluster and an outlier
set.seed(81)
```

```
env_data <- data.frame(
  BIO1 = c(rnorm(50, mean = 10, sd = 1), 30),
  BIO12 = c(rnorm(50, mean = 20, sd = 2), 50)
)

# 2. Fit a 95% ellipse using the standard covariance method
fit <- fit_ellipsoid(
  data = env_data,
  env_vars = c("BIO1", "BIO12"),
  method = "covmat",
  level = 0.95
)

# 3. Print the summary and plot the results
print(fit)
plot(fit)

## End(Not run)
```

`plot_bean` *Visualize environmental thinning results*

Description

This function creates a comparison plot showing the original, unthinned occurrence points overlaid with the results of an environmental thinning process. It draws the environmental grid to visualize how points were either sampled (stochastic) or consolidated to cell centers (deterministic).

Usage

```
plot_bean(original_data, thinned_object, env_vars, grid_resolution)
```

Arguments

`original_data`	A data.frame of the prepared, unthinned occurrence points.
`thinned_object`	The output object from either `thin_env_density()` or `thin_env_center()`. The function will automatically detect the thinning type.
`env_vars`	A character vector of length two specifying the names of the environmental variables to plot on the x and y axes.
`grid_resolution`	A numeric vector of length two specifying the grid resolution used for thinning. This is required to draw the grid correctly.

Value

A ggplot object.

Examples

```
## Not run:
# Assume 'prepared_data' and 'grid_res' are available from previous steps.

# --- Example 1: Visualizing Stochastic Thinning ---
set.seed(123)
thinned_stochastic <- thin_env_density(
  data = prepared_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = grid_res,
  max_per_cell = 1
)

plot_bean(
  original_data = prepared_data,
  thinned_object = thinned_stochastic,
  grid_resolution = grid_res,
  env_vars = c("BIO1", "BIO12")
)

# --- Example 2: Visualizing Deterministic Thinning ---
thinned_deterministic <- thin_env_center(
  data = prepared_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = grid_res
)

plot_bean(
  original_data = prepared_data,
  thinned_object = thinned_deterministic,
  grid_resolution = grid_res,
  env_vars = c("BIO1", "BIO12")
)

## End(Not run)
```

`prepare_bean` *Prepare data for environmental thinning*

Description

This function serves as a pre-processing step to clean and prepare species occurrence data. It performs three key actions: 1. Removes records with missing longitude or latitude values. 2. Extracts environmental data from raster layers that are already scaled for each occurrence point. 3. Removes records that fall outside the raster extent or have missing environmental data. The final output is a clean data frame where the environmental variables have a mean of 0 and a standard deviation of 1.

Usage

```
prepare_bean(data, env_rasters, longitude, latitude, scale = TRUE)
```

Arguments

`data`	A data.frame of species occurrences records, including columns for longitude and latitude.
`env_rasters`	A SpatRaster (from `terra` package) or RasterStack (from `raster` package) object of environmental variables.
`longitude`	(character) The name of the longitude column in `data`.
`latitude`	(character) The name of the latitude column in `data`.
`scale`	(logical) If "TRUE" (default), the environmental variables in `env_rasters` will be scaled before extraction. Set to "FALSE" if your rasters are already on a comparable scale. See Details

Details

### The Importance of Scaling Environmental Variables (`scale`)

Environmental variables used in ecological modeling, such as temperature and precipitation, often have vastly different units (e.g., °C vs. mm) and numerical ranges. When using methods based on distance calculations—such as the Euclidean and Mahalanobis distance these differences in scale can unintentionally bias the analysis. Variables with larger numerical ranges will dominate the distance metric, while those with smaller ranges will have a negligible influence (Qiao et al., 2016).

Standardization is the standard procedure to address this issue. By setting "scale = TRUE", the function transforms each variable to have a mean of 0 and a standard deviation of 1 (i.e., it calculates z-scores) (Baddeley et al., 2016). This process makes the variables equal variance. As a result, each variable contributes equally to the analysis, ensuring that the resulting resolutions are based on the relative distribution of data points within each environmental dimension, not their arbitrary original units (Beaugrand, 2024; Kléparski et al., 2021).

Value

A data.frame containing the cleaned and scaled occurrence data, with the following columns:

`Original Columns`
All columns from the input `data` are preserved for the valid records.

`Environmental Variables`
New columns, named after the layers in `env_rasters`, containing the extracted and scaled environmental data.

References

Baddeley, A., Rubak, E. and Turner, R. (2016). Spatial point patterns: methodology and applications with R. CRC press.

Beaugrand, G. (2024). An ecological niche model that considers local relationships among variables: The Environmental String Model. Ecosphere, 15(10), e70015.

Kléparski, L., Beaugrand, G. and Edwards, M. (2021). Plankton biogeography in the North Atlantic Ocean and its adjacent seas: Species assemblages and environmental signatures. Ecology and Evolution, 11(10), 5135-5149.

Qiao, H., Peterson, A. T., Campbell, L. P., Soberón, J., Ji, L. and Escobar, L. E. (2016). NicheA: creating virtual species and ecological niches in multivariate environmental scenarios. Ecography, 39(8), 805-813.

Examples

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

# 1. Load sample data
bio1_file <- system.file("extdata", "BIO1.tif", package = "bean")
bio12_file <- system.file("extdata", "BIO12.tif", package = "bean")
env_rasters <- terra::rast(c(bio1_file, bio12_file))

occ_file <- system.file("extdata", "Peromyscus_maniculatus_original.csv", package = "bean")
data <- read.csv(occ_file)

# 2. Run the preparation function
prepared_data <- prepare_bean(
  data = data,
  env_rasters = env_rasters,
  longitude = "x",
  latitude = "y",
  scale = TRUE
)

# 3. View the clean, scaled data
head(prepared_data)
summary(prepared_data)

## End(Not run)
```

`test_env_thinning` *Perform repeated k-fold cross-validation for a Maxent model*

Description

This function provides a metric for evaluating the effect of occurrences thinning in environmental space using a Maxent model (see Details). Performance is tested using repeated k-fold cross-validation. The test calculates the Area Under the Curve (AUC) for each fold and repetition, providing a distribution of scores to assess model stability and predictive accuracy.

Usage

```
test_env_thinning(
  presence_data,
  background_data,
  env_rasters,
  longitude,
  latitude,
  k = 4,
  n_repeats = 10,
 maxent_args = c("linear=true", "quadratic=false", "product=false", "threshold=false",
   "hinge=false", "doclamp=true")
)
```

Arguments

`presence_data`	A data.frame with at least two columns for longitude and latitude of presence points.
`background_data`	A data.frame with at least two columns for longitude and latitude of background (or pseudo-absence) points.
`env_rasters`	A RasterStack or SpatRaster object of environmental variables.
`longitude`	(character) The name of the longitude column in the data.frames.
`latitude`	(character) The name of the latitude column in the data.frames.
`k`	(numeric) The number of folds for cross-validation. Default = 4.
`n_repeats`	(numeric) The number of times to repeat the k-fold process. Default = 10.
`maxent_args`	(character) A vector of arguments to be passed to the "dismo::maxent" function. The default enables only linear features. See "?dismo::maxent" for details.

Details

This function serves as a wrapper for the `dismo::maxent` function, automating the process of model evaluation through repeated k-fold cross-validation. This approach provides a robust assessment of a model's predictive performance and stability, which is particularly useful when comparing models built with different data inputs (ten Caten et al., 2023).

### Maxent (`dismo`)

`test_env_thinning` relies on the `dismo` package, which provides an R interface to the standalone Maxent Java application (Hijmans et al., 2024). Maxent is a presence-background (or presence-only) machine learning algorithm that estimates a species' potential distribution based on the principle of maximum entropy (Phillips et al., 2006). It contrasts the environmental conditions at known presence locations (`presence_data`) with the environmental conditions available across the broader study area, as represented by the `background_data`. The `maxent_args` parameter allows for direct control over model complexity, such as specifying different feature classes (e.g., linear, quadratic, hinge) and adjusting the regularization multiplier to prevent overfitting (Phillips et al., 2006; Warren and Seifert, 2011).

### Repeated K-Fold Cross-Validation

To provide a rigorous evaluation, this function implements a repeated k-fold cross-validation procedure. The process is as follows: 1. The `presence_data` is randomly partitioned into `k` equal-sized subsets (folds). 2. The model is trained using `k-1` folds and then evaluated on the single held-out fold. 3. This is repeated `k` times, with each fold serving as the test set exactly once. 4. The entire k-fold process is then repeated `n_repeats` times, each time with a new random partitioning of the data.

### Model Evaluation using AUC

The function calculates the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) plot for each test fold. AUC is a threshold-independent metric that measures the model's ability to discriminate between a random presence site and a random background site (Phillips et al., 2006). An AUC value of 1.0 indicates perfect discrimination, while a value of 0.5 indicates performance no better than random (Hijmans et al., 2024).

Value

An object of class `bean_evaluation`, which is a list containing:

`summary`	A data.frame with summary statistics (mean, sd, etc.) of the AUC scores.

`all_auc_scores`	A numeric vector of all AUC scores from every fold and repetition.
`parameters`	A list of the key parameters used in the evaluation.

Note

For reproducible results, run "set.seed()" before calling this function.

References

Hijmans R, Phillips S, Leathwick J, Elith J (2024). dismo: Species Distribution Modeling. R package version 1.3-16, <https://rspatial.org/raster/sdm/>.

Phillips, S. J., Anderson, R. P., & Schapire, R. E. (2006). Maximum entropy modeling of species geographic distributions. Ecological modelling, 190(3-4), 231-259.

Ten Caten, C., & Dallas, T. (2023). Thinning occurrence points does not improve species distribution model performance. Ecosphere, 14(12), e4703.

Valavi, R., Elith, J., Lahoz-Monfort, J. J., & Guillera-Arroita, G. (2018). blockCV: An r package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models. Biorxiv, 357798.

Warren, D. L., & Seifert, S. N. (2011). Ecological niche modeling in Maxent: the importance of model complexity and the performance of model selection criteria. Ecological applications, 21(2), 335-342.

Examples

```
## Not run:
# This is a long-running example and requires the "dismo" package to be installed.

# 1. Load the package's example data
library(raster)
library(dismo)
occ_file <- system.file("extdata", "Peromyscus_maniculatus_prepared.csv", package = "bean")
occ_data <- read.csv(occ_file)

bio1_file <- system.file("extdata", "BIO1.tif", package = "bean")
bio12_file <- system.file("extdata", "BIO12.tif", package = "bean")
env_rasters <- raster::stack(bio1_file, bio12_file)

# 2. Create background points
set.seed(81)
background_pts <- dismo::randomPoints(env_rasters, 1000)
background_df <- as.data.frame(background_pts)
colnames(background_df) <- c("x", "y")

# 3. Run the evaluation with minimal settings for the example
auc_results <- test_env_thinning(
  presence_data = occ_data,
  background_data = background_df,
  env_rasters = env_rasters,
  longitude = "x",
  latitude = "y",
  k = 2,
  n_repeats = 2, # Use a small number for the example
  maxent_args = c("linear=true",
  "quadratic=true",
  "product=false",
  "threshold=false",
  "hinge=false",
  "doclamp=false")
)

# 4. Print the summary and plot the distribution of AUC scores
print(auc_results)
plot(auc_results)

## End(Not run)
```

`thin_env_center` *Deterministic centroid*

Description

This function thins species occurrence records by finding all occupied cells in a 2D environmental grid and returning a single new point at the exact center of each of those cells. This is a deterministic method.

Usage

```
thin_env_center(data, env_vars, grid_resolution)
```

Arguments

`data`	A data.frame containing species occurrence coordinates and the environmental variables.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables to be used in the analysis.
`grid_resolution`	A numeric vector of length one or two specifying the resolution(s) for the grid axes. If length one, it is used for both axes. See the Details section of `find_env_resolution` for a full explanation

Value

An object of class `bean_thinned_center`. which is a list containing:

`thinned_points`	A data.frame with two columns representing the new points at the center of each occupied environmental grid cell.
`n_original`	An integer representing the number of complete occurrence records in the input data.
`n_thinned`	An integer representing the number of unique grid cells that were occupied, which is also the number of points returned.
`parameters`	A list of the key parameters used, such as whether scaling was applied.

See Also

`find_env_resolution`

Examples

```
## Not run:
# 1. Create environmental data
set.seed(81)
env_data <- data.frame(
BIO1 = c(0.1, 0.2, 1.1, 1.2, 1.3),
BIO12 = c(0.1, 0.2, 2.1, 2.2, 2.3)
)

# 2. Thin the data to grid cell centers
thinned_center_obj <- thin_env_center(
  data = env_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = c(0.1, 0.2)
)

# 3. Print the summary
print(thinned_center_obj)

## End(Not run)
```

`thin_env_density` *Stochastic centroid*

Description

This function thins species occurrence records in a 2D environmental space by randomly sampling a specified number of points from each occupied grid cell. This is a stochastic method.

Usage

```
thin_env_density(data, env_vars, grid_resolution, max_per_cell)
```

Arguments

`data`	A data.frame containing species occurrence coordinates and the environmental variables.
`env_vars`	A character vector specifying the column names in data that represent the environmental variables to be used in the analysis.
`grid_resolution`	A numeric vector of length one or two specifying the resolution(s) for the grid axes. If length one, it is used for both axes. See the Details section of `find_env_resolution` for a full explanation
`max_per_cell`	An integer specifying the maximum number of points to retain per grid cell. See the Details section of `find_optimal_cap` for a full explanation

Value

An object of class `bean_thinned_density`, which is a list containing:

`thinned_data`	A data.frame containing the occurrence records that were retained after the thinning process.

`n_original`	An integer representing the number of complete occurrence records in the input data before thinning.
`n_thinned`	An integer representing the number of occurrence records remaining after thinning.

See Also

`find_optimal_cap`, `find_env_resolution`

Examples

```
## Not run:
# 1. Create environmental data
env_data <- data.frame(
BIO1 = c(rep(0.1, 5), rep(1.1, 3)),
BIO12 = c(rep(0.2, 5), rep(2.2, 3)),
species = "A"
)

# 2. Thin the data to a max of 1 point per cell
set.seed(81) # For reproducible results
thinned_obj <- thin_env_density(
  data = env_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = c(0.2, 0.2),
  max_per_cell = 1
)

# 3. Print the summary
print(thinned_obj)

## End(Not run)
```

Index

calibrate_bean, 2

find_env_resolution, 4, *7*, *16–18*
find_optimal_cap, 6, *17, 18*
fit_ellipsoid, 8

plot_bean, 10
prepare_bean, 11

test_env_thinning, 13, *14*
thin_env_center, 16
thin_env_density, 17