Package 'bean'

July 21, 2025

Type Package

Title Data Thinning of Species Occurrences in Environmental Space

Version 0.1.0

Maintainer Paanwaris Paansri <paanwaris@vt.edu>

Description Provides a suite of tools to address sampling bias in species occurrence records by thinning the data in environmental space. This process is a crucial pre-processing step for ecological niche modeling (ENM). The package offers a data-driven protocol to objectively determine thinning parameters and includes multiple methods for thinning (stochastic, deterministic) and for delineating the final environmental condition using robust ellipses. The name 'bean' reflects the core principle of the method, ensuring that each "pod" (a grid cell in environmental 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, testthat, dismo, dplyr, ggplot2, tidyr, magrittr, rlang, stats, MASS, rJava, covr

Depends R (>= 3.5)

Contents

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

Description

This function assumes the environmental variables in the input data have already been scaled (e.g., using 'scale()'). It calculates the distribution of all pairwise 1D distances for each variable. The distance at a specified lower quantile is returned as a suggested grid resolution for each axis.

Usage

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

Arguments

`data`	A data frame containing species occurrences and pre-scaled environmental data.
`env_vars`	A character vector of length two specifying the names of the environmental variables to use.
`quantile`	(numeric) The quantile of pairwise distances to use for the resolution. A smaller value (e.g., 0.05 or 0.1) is recommended. Default = 0.1.
`verbose`	(logical) If TRUE, prints progress messages and warnings. Default = TRUE.

Value

An object of class `bean_resolution`.

Examples

```
## Not run:
# 1. Load the example occurrence data included with the package
occ_file <- system.file("extdata", "P_maniculatus_samples.csv", package = "bean")
occ_data <- read.csv(occ_file)
# 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)

# 4. Extract the suggested resolution
grid_res <- resolution_results$suggested_resolution

## End(Not run)
```

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

Description

This function searches for optimal density caps using two criteria: 1) the cap that results in a point count closest to the target percentage, and 2) the cap that results in the closest point count AT OR ABOVE the target.

Usage

```
find_optimal_cap(
  data,
  env_vars,
  grid_resolution,
  target_percent,
  verbose = TRUE
)
```

Arguments

`data`	A data frame containing species occurrences and environmental data.
`env_vars`	A character vector of length two specifying the names of the columns to be used as the axes of the environmental space.
`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.
`target_percent`	A numeric value (0-1) for the target proportion of points to retain.
`verbose`	(logical) If TRUE, prints progress messages. Default = TRUE.

Value

An object of class `bean_optimization`.

Examples

```
## Not run:
# 1. Load and prepare the data
occ_file <- system.file("extdata", "P_maniculatus_samples.csv", package = "bean")
occ_data <- read.csv(occ_file)

# 2. Find optimal cap to retain ~80% of the data
set.seed(81) # For reproducibility
optimal_params <- find_optimal_cap(
  data = occ_data,
  env_vars = c("BIO1", "BIO12"),
  grid_resolution = 0.2, # Using an example resolution
  target_percent = 0.80
)

# 3. Print the summary and plot the results
```

```
print(optimal_params)
plot(optimal_params)

# 4. Extract the best cap
best_cap <- optimal_params$best_cap_above_target

## End(Not run)
```

`fit_ellipsoid` *Fit a bivariate environmental ellipsoid to occurrence data*

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, var1, var2, method = "covmat", level = 95)
```

Arguments

`data`	A data frame containing species occurrences and environmental data. It is highly recommended that this data be scaled.
`var1`	(character) The name of the first environmental variable column.
`var2`	(character) The name of the second environmental variable column.
`method`	(character) The method for calculating the centroid and covariance matrix. Options are "covmat" (standard covariance) and "mve" (Minimum Volume Ellipsoid, robust to outliers). Default = "covmat".
`level`	(numeric) The confidence level (e.g., 95 for 95 ellipse. This determines what proportion of the points the ellipse should contain. Default = 95.

Value

An object of class `bean_ellipsoid`.

Examples

```
## Not run:
# 1. Create some thinned data to work with
set.seed(81)
thinned_data <- data.frame(
  BIO1 = rnorm(50),
  BIO12 = rnorm(50)
)

# 2. Fit a 95% ellipse using the robust covariance matrix method
niche_ellipse <- fit_ellipsoid(
  data = thinned_data,
  var1 = "BIO1",
```

```
  var2 = "BIO12",
  method = "covmat",
  level = 95
)

# 3. Print the summary and plot the result
print(niche_ellipse)
plot(niche_ellipse)

## End(Not run)
```

`plot.bean_ellipsoid` *Plot bean_ellipsoid results*

Description

Creates a diagnostic plot from the output of `fit_ellipsoid`.

Usage

```
## S3 method for class 'bean_ellipsoid'
plot(x, ...)
```

Arguments

`x`	An object of class `bean_ellipsoid`.
`...`	Additional arguments (not used).

Value

A ggplot object.

`plot.bean_evaluation` *Plot bean_evaluation results*

Description

Creates a diagnostic plot from the output of `test_model_auc`.

Usage

```
## S3 method for class 'bean_evaluation'
plot(x, ...)
```

Arguments

`x`	An object of class `bean_evaluation`.
`...`	Additional arguments (not used).

Value

A ggplot object.

plot.bean_optimization *Plot bean_optimization results*

Description

Creates a diagnostic plot from the output of `find_optimal_cap`.

Usage

```
## S3 method for class 'bean_optimization'
plot(x, ...)
```

Arguments

`x`	An object of class `bean_optimization`.
`...`	Additional arguments (not used).

Value

A ggplot object.

plot.bean_resolution *Plot bean_resolution results*

Description

Creates a diagnostic plot from the output of `find_env_resolution`.

Usage

```
## S3 method for class 'bean_resolution'
plot(x, ...)
```

Arguments

`x`	An object of class `bean_resolution`.
`...`	Additional arguments (not used).

Value

A ggplot object.

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

Description

This function provides a robust framework for evaluating a Maxent model's performance using repeated k-fold cross-validation. It calculates the Area Under the Curve (AUC) for each run, providing a distribution of scores to assess model stability and predictive accuracy.

Usage

```
test_model_auc(
  presence_data,
  background_data,
  env_rasters,
  longitude,
  latitude,
  k = 4,
  n_repeats = 10,
 maxent_args = c("linear=true", "quadratic=true", "product=false", "threshold=false",
    "hinge=false", "doclamp=true"),
  verbose = 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. See '?dismo::maxent' for details.
`verbose`	(logical) If TRUE, prints progress messages. Default = TRUE.

Value

An object of class `bean_evaluation`.

Examples

```
## Not run:
# This is a long-running example and requires Maxent to be installed.

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

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

# 2. Create background points
set.seed(1)
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_model_auc(
  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=true")
)

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

## End(Not run)
```

---

thin_env_center *Thin occurrence data to grid cell centers (deterministic)*

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, verbose = TRUE)
```

Arguments

data	A data frame containing species occurrences and environmental data.
env_vars	A character vector of length two specifying the names of the environmental variables to use.
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.
verbose	(logical) If TRUE, prints progress messages. Default = TRUE.

Value

An object of class `bean_thinned_center`.

Examples

```
## Not run:
# 1. Load and prepare the data
occ_file <- system.file("extdata", "P_maniculatus_samples.csv", package = "bean")
occ_data <- read.csv(occ_file)

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

# 3. Print the summary
print(thinned_center_obj)

# 4. Access the new centroid points
thinned_centers_df <- thinned_center_obj$thinned_points

## End(Not run)
```

`thin_env_density` *Thin occurrence data based on environmental density (stochastic)*

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.

Usage

```
thin_env_density(data, env_vars, grid_resolution, max_per_cell, verbose = TRUE)
```

Arguments

data	A data frame containing species occurrences and environmental data.
env_vars	A character vector of length two specifying the names of the environmental variables to use.
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.
max_per_cell	An integer specifying the maximum number of points to retain per grid cell.
verbose	(logical) If TRUE, prints progress messages. Default = TRUE.

Value

An object of class `bean_thinned_density`.

Examples

```
## Not run:
# 1. Load and prepare the data
occ_file <- system.file("extdata", "P_maniculatus_samples.csv", package = "bean")
occ_data <- read.csv(occ_file)

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

# 3. Print the summary
print(thinned_obj)

# 4. Access the thinned data frame
thinned_df <- thinned_obj$thinned_data

## End(Not run)
```

Index