# Package 'enmpa'

June 12, 2025

**Title** Ecological Niche Modeling using Presence-Absence Data

**Version** 0.2.1

**Maintainer** Luis F. Arias-Giraldo <lfarias.giraldo@gmail.com>

**Date** 2025-06-12

**Description** A set of tools to perform Ecological Niche Modeling with
presence-absence data. It includes algorithms for data partitioning,
model fitting, calibration, evaluation, selection, and prediction.
Other functions help to explore signals of ecological niche using univariate
and multivariate analyses, and model features such as variable response
curves and variable importance. Unique characteristics of this package are
the ability to exclude models with concave quadratic responses, and the
option to clamp model predictions to specific variables. These tools are
implemented following principles proposed in
Cobos et al., (2022) <doi:10.17161/bi.v17i.15985>,
Cobos et al., (2019) <doi:10.7717/peerj.6281>,
and Peterson et al., (2008) <doi:10.1016/j.ecolmodel.2007.11.008>.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**URL** https://github.com/Luisagi/enmpa

**BugReports** https://github.com/Luisagi/enmpa/issues

**Imports** doSNOW, ellipse, foreach, graphics, methods, mgcv, parallel,
Rcpp, snow, stats, terra, utils, vegan

**LazyData** true

**Depends** R (>= 3.5.0)

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Author** Luis F. Arias-Giraldo [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4861-8064>),
Marlon E. Cobos [aut] (ORCID: <https://orcid.org/0000-0002-2611-1767>),
A. Townsend Peterson [ctb] (ORCID:
<https://orcid.org/0000-0003-0243-2379>)

# Contents

---

| calibration_glm | *GLM calibration with presence-absence data* |
|---|---|

---

## Description

Creates candidate models based on distinct parameter settings, evaluates models, and selects the ones that perform the best.

## Usage

```
calibration_glm(data, dependent, independent, weights = NULL,
                response_type = "l", formula_mode = "moderate",
                minvar = 1, maxvar = NULL, user_formulas = NULL,
                cv_kfolds = 5, partition_index = NULL, seed = 1,
                n_threshold = 100, selection_criterion = "TSS",
                exclude_bimodal = FALSE, tolerance = 0.01,
                out_dir = NULL, parallel = FALSE,
                n_cores = NULL, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| data | data.frame or matrix of data to be used in model calibration. Columns represent dependent and independent variables. |
| dependent | (character) name of dependent variable. |
| independent | (character) vector of name(s) of independent variable(s). |
| weights | (numeric) a vector with the weights for observations. |
| response_type | (character) a character string that must contain "l", "p", "q" or a combination of them. l = lineal, q = quadratic, p = interaction between two variables. Default = "l". |
| formula_mode | (character) a character string to indicate the strategy to create the formulas for candidate models. Options are: "light", "moderate", "intensive", or "complex". Default = "moderate". "complex" returns only the most complex formula defined in response_type. |
| minvar | (numeric) minimum number of independent variables in formulas. |
| maxvar | (numeric) maximum number of independent variables in formulas. |
| user_formulas | (character) vector with formula(s) to test. Default = NULL. |
| cv_kfolds | (numeric) number of folds to use for k-fold cross-validation exercises. Default = 5. Ignored if partition_index is defined. |
| partition_index | |
| | list of indices for cross-validation in k-fold. The default, NULL, uses the function [kfold_partition](). |
| seed | (numeric) a seed for k-fold partitioning. |
| n_threshold | (logical) number of threshold values to produce evaluation metrics. Default = 100. |
| selection_criterion | |
| | (character) criterion used to select best models, options are "TSS" and "ESS". Default = "TSS". |
| exclude_bimodal | |
| | (logical) whether to filter out models with one or more variables presenting concave responses. Default = FALSE. |
| tolerance | (numeric) value to modify the limit value of the metric used to filter models during model selection if none of the models meet initial considerations. Default = 0.01 |

|             |                                                                                      |
| ----------- | ------------------------------------------------------------------------------------ |
| out_dir     | (character) output directory name to save the main calibration results. Default = NULL. |
| parallel    | (logical) whether to run on parallel or sequential. Default = FALSE.                  |
| n_cores     | (numeric) number of cores to use. Default = number of free processors - 1.           |
| verbose     | (logical) whether to print messages and show progress bar. Default = TRUE            |

### Details

Model evaluation is done considering the ability to predict presences and absences,as well as model fitting and complexity. Model selection consists of three steps: 1) a first filter to keep the models with ROC AUC >= 0.5 (statistically significant models), 2) a second filter to maintain only models that meet the selection_criterion ("TSS": TSS >= 0.4; or "ESS": maximum Accuracy - tolerance), and 3) from those, pick the ones with delta AIC <= 2.

formula_mode options determine what strategy to iterate the predictors defined in type for creating models:

- **light**.– returns simple iterations of complex formulas.
- **moderate**.– returns a comprehensive number of iterations.
- **intensive**.– returns all possible combination. Very time-consuming for 6 or more independent variables.
- **complex**.– returns only the most complex formula.

### Value

An object of the class enmpa_calibration containing: selected models, a summary of statistics for all models, results obtained in cross-validation for all models, original data used, weights, and data-partition indices used.

### Examples

```
# Load species occurrences and environmental data.
data("enm_data", package = "enmpa")
head(enm_data)

# Calibration using linear (l), quadratic (q), products(p) responses.
cal_res <- calibration_glm(data = enm_data, dependent = "Sp",
                           independent = c("bio_1", "bio_12"),
                           response_type = "lpq", formula_mode = "moderate",
                           selection_criterion = "TSS", cv_kfolds = 3,
                           exclude_bimodal = TRUE, verbose = FALSE)
print(cal_res)
summary(cal_res)

head(cal_res$calibration_results)
head(cal_res$summary)
head(cal_res$selected)
head(cal_res$data)
```

---

cal_res                       *Example of results obtained from GLM calibration using enmpa*

---

### Description

An object of the class enmpa_calibration storing the results from GLM calibration.

### Usage

```
cal_res
```

### Format

An object of class enmpa_calibration with results from the function \link{calibration_glm}.

### Examples

```
data("cal_res", package = "enmpa")

str(cal_res)
```

---

enmpa                         *enmpa: Ecological Niche Modeling using Presence-Absence Data*

---

### Description

enmpa contains a set of tools to perform detailed Ecological Niche Modeling using presence-absence data.

### Details

It includes algorithms for data partitioning, model fitting, calibration, evaluation, selection, and prediction. Other functions help to explore model features as such variable response curves and variable importance.

### Main functions in enmpa

[calibration_glm](), [evaluation_stats](), [fit_glms](), [fit_selected](), [get_formulas](), [independent_eval1](), [kfold_partition](), [model_selection]() [model_validation](), [niche_signal](), [optimize_metrics](), [predict_glm](), [predict_selected](), [response_curve](), [resp2var](), [var_importance](), [jackknife](), [plot_jk]()

**Author(s)**

**Maintainer**: Luis F. Arias-Giraldo <lfarias.giraldo@gmail.com> (ORCID)

Authors:

- Marlon E. Cobos <manubio13@gmail.com> (ORCID)

Other contributors:

- A. Townsend Peterson <town@ku.edu> (ORCID) [contributor]

**See Also**

Useful links:

- https://github.com/Luisagi/enmpa
- Report bugs at https://github.com/Luisagi/enmpa/issues

---

enmpa_calibration                    *Constructor of S3 objects of class enmpa_calibration*

---

**Description**

Constructor of S3 objects of class enmpa_calibration

**Usage**

```
new_enmpa_calibration(selected, summary, calibration_results, data,
                      partitioned_data, weights = NULL)
```

**Arguments**

| | |
|---|---|
| selected | date.frame with information about selected models. |
| summary | data.frame a summary of statistics for all models. |
| calibration_results | |
| | data.frame with results obtained from cross-validation for all models. |
| data | data.frame or matrix with the input data used for calibration. |
| partitioned_data | |
| | a list of partition indices. |
| weights | (numeric) a vector with the weights for observations. Default = NULL. |

**Value**

An S3 object of class enmpa_calibration.

---

enmpa_fitted_models      *Constructor of S3 objects of class enmpa_fitted_models*

---

### Description

Constructor of S3 objects of class enmpa_fitted_models

### Usage

```
new_enmpa_fitted_models(glms_fitted, selected, data, weights = NULL)
```

### Arguments

| | |
|---|---|
| glms_fitted | a list of fitted GLMs. |
| selected | date.frame with information about selected models. |
| data | data.frame or matrix with the input data used for calibration. |
| weights | (numeric) a vector with the weights for observations. Default = NULL. |

### Value

An S3 object of class enmpa_fitted_models.

---

enm_data      *Example data used to run model calibration exercises*

---

### Description

A dataset containing information on presence and absence, and independent variables used to fit GLM models.

### Usage

```
enm_data
```

### Format

A data frame with 5627 rows and 3 columns.

**Sp** numeric, values of 0 = absence and 1 = presence.

**bio_1** numeric, temperature values.

**bio_12** numeric, precipitation values.

### Examples

```
data("enm_data", package = "enmpa")
head(enm_data)
```

---

evaluation_stats            *Summary of evaluation statistics for candidate models*

---

### Description

Calculate median and standard deviation of evaluation results for all candidate models considering cross-validation kfolds.

### Usage

```
evaluation_stats(evaluation_results, bimodal_toexclude = FALSE)
```

### Arguments

evaluation_results

> data.frame model evaluation results. These results are the output of the function [model_validation](model_validation).

bimodal_toexclude

> (logical) whether models in which binomial variable response curves were detected will be excluded during selection processes.

### Value

A data.frame with the mean and standard deviation for all metrics considering cross-validation kfolds.

### Examples

```
# data
data("cal_res", package = "enmpa")
all_res <- cal_res$calibration_results[, -1]

# statistics for all evaluation results
evaluation_stats(all_res, bimodal_toexclude = TRUE)
```

---

fit_selected                *Fitting selected GLMs models*

---

### Description

Functions to facilitate fitting multiple GLMs.

### Usage

```
fit_selected(glm_calibration)

fit_glms(formulas, data, weights = NULL, id = NULL)
```

## Arguments

glm_calibration

> a list resulting from [calibration_glm](calibration_glm). Models fitted are those in the slot "selected".

formulas          (character) a vector containing the formula(s) for GLM(s).

data                data.frame with the dependent and independent variables.

weights           (numeric) a vector with the weights for observations. Default = NULL.

id                  (character) id code for models fitted. Default = NULL.

## Value

A list of fitted GLMs.

For fit_selected, an enmpa fitted models object.

## Examples

```
# GLM calibration results
data(cal_res, package = "enmpa")

# Fitting selected models
sel_fit <- fit_selected(cal_res)

sel_fit

# Custom formulas
forms <- c("Sp ~ bio_1 + I(bio_1^2) + I(bio_12^2)",
           "Sp ~ bio_12 + I(bio_1^2) + I(bio_12^2)")

# Fitting models
fits <- fit_glms(forms, data = cal_res$data)

fits$ModelID_1
```

---

get_formulas                 *Get GLM formulas according to defined response types*

---

## Description

Generate GLM formulas for independent variables predicting a dependent variable, taking into account response types required. All possible combinations of variables can be created using arguments of the function.

**Usage**

```
get_formulas(dependent, independent, type = "l", mode = "moderate",
             minvar = 1, maxvar = NULL)

get_formulas_main(dependent, independent, type = "l",
                  complex = FALSE, minvar = 1, maxvar = NULL)

aux_var_comb(var_names, minvar = 2, maxvar = NULL)

aux_string_comb(string)
```

**Arguments**

| | |
|---|---|
| dependent | (character) name of dependent variable. |
| independent | (character) vector of name(s) of independent variable(s). |
| type | (character) a character string that must contain "l", "p", "q" or a combination of them. l = lineal, q = quadratic, p = interaction between two variables. Default = "l". |
| mode | (character) (character) a character string to indicate the strategy to create the formulas for candidate models. Options are: "light", "moderate", "intensive", or "complex". Default = "moderate". |
| minvar | (numeric) minimum number of independent variables in formulas. |
| maxvar | (numeric) maximum number of independent variables in formulas. |
| complex | (logical) whether to return the most complex formula. |
| var_names | sames as independent. |
| string | same as type. |

**Details**

mode options determine what strategy to iterate the predictors defined in type for creating models:

- **light**.– returns simple iterations of complex formulas.

- **moderate**.– returns a comprehensive number of iterations.

- **intensive**.– returns all possible combination. Very time-consuming for 6 or more independent variables.

- **complex**.– returns only the most complex formula.

**Value**

A character vector containing the resulting formula(s).

### Examples

```
# example variables
dep <- "sp"
ind <- c("temp", "rain", "slope")

# The most complex formula according to "type"
get_formulas(dep, ind, type = "lqp", mode = "complex")

# mode = 'light', combinations according to type
get_formulas(dep, ind, type = "lqp", mode = "light")

# mode = 'light', combinations according to type
get_formulas(dep, ind, type = "lqp", mode = "intensive")
```

---

independent_eval1 *Evaluate final models using independent data*

---

### Description

Final evaluation steps for model predictions using an independent dataset (not used in model calibration).

### Usage

```
independent_eval1(prediction, threshold, test_prediction = NULL,
                  lon_lat = NULL)

independent_eval01(prediction, observation, lon_lat = NULL)
```

### Arguments

| | |
|---|---|
| prediction | (numeric) vector or SpatRaster object. If numeric, predicted values in independent data (for [independent_eval01](#)), or the entire area of prediction (for [independent_eval1](#)). If SpatRaster prediction over the area of interest. |
| threshold | (numeric) the lowest predicted probability value for an occurrence point. This value must be defined for presences-only data. Default = NULL. |
| test_prediction | |
| | (numeric) vector of predictions for independent data. Default = NULL. |
| lon_lat | matrix or data.frame of coordinates (longitude and latitude, in that order) of independent data. Points must be located within the valid area of prediction. For [independent_eval01](#) they must correspond with values in observation. |
| observation | (numeric) vector of observed (known) values of presence or absence to test against prediction (if numeric) or values of prediction in lon_lat. |

### Value

A data.frame or list containing evaluation results.

## Examples

```
# Independent test data based on coordinates (lon/lat WGS 84) from presence
# and absences records
data("test", package = "enmpa")
head(test)

# Loading a model prediction
pred <- terra::rast(system.file("extdata", "proj_out_wmean.tif",
                                package = "enmpa"))
terra::plot(pred)

# Evaluation using presence-absence data
independent_eval01(prediction = pred, observation = test$Sp,
                   lon_lat = test[, 2:3])

# Evaluation using presence-only data
test_p_only <- test[test$Sp == 1, ]
th_maxTSS   <- 0.1274123            # threshold based on the maxTSS

independent_eval1(prediction = pred, threshold = th_maxTSS,
                  lon_lat = test_p_only[, 2:3])
```

---

jackknife | *Jackkniffe test for variable contribution*

---

## Description

The Jackknife function providing a detailed reflection of the impact of each variable on the overall model, considering four difference measures: ROC-AUC, TSS, AICc, and Deviance.

## Usage

```
jackknife(data, dependent, independent, user_formula = NULL, cv = 3,
          response_type = "l", weights = NULL)
```

## Arguments

| | |
|---|---|
| data | data.frame or matrix of data to be used in model calibration. Columns represent dependent and independent variables. |
| dependent | (character) name of dependent variable. |
| independent | (character) vector of name(s) of independent variable(s). |
| user_formula | (character) custom formula to test. Default = NULL. |
| cv | (numeric) number of folds to use for k-fold cross-validation exercises. Default = 3. |
| response_type | (character) a character string that must contain "l", "p", "q" or a combination of them. l = lineal, q = quadratic, p = interaction between two variables. Default = "l". |
| weights | (numeric) a vector with the weights for observations. |

**Value**

list including model performance metrics (ROC-AUC, TSS, AICc, and deviance) for the complete model, model performance when excluding a specific predictor, and the independent contribution of that predictor to the model.

**Examples**

```
# Load data
data("enm_data", package = "enmpa")

jk <- jackknife(data = enm_data,
                dependent = "Sp",
                independent = c("bio_1", "bio_12"),
                user_formula = NULL,
                cv = 3, response_type = "lpq")
jk

# plot JK's results
plot_jk(jk, metric = "TSS")
plot_jk(jk, metric = "ROC_AUC")
plot_jk(jk, metric = "AIC")
plot_jk(jk, metric = "Residual_deviance")
```

---

kfold_partition            *K-fold data partitioning*

---

**Description**

Creates indices to partition available data into k equal-sized subsets or folds, maintaining the global proportion of presence-absences in each fold.

**Usage**

```
kfold_partition(data, dependent, k = 2, seed = 1)
```

**Arguments**

| | |
|---|---|
| data | data.frame or matrix containing at least two columns. |
| dependent | (character) name of column that contains the presence-absence records (1-0). |
| k | (numeric) the number of groups that the given data is to be split into. |
| seed | (numeric) integer value to specify an initial seed. Default = 1. |

**Value**

A list of vectors with the indices of rows corresponding to each fold.

## Examples

```
# example data
data <- data.frame(species = c(rep(0, 80), rep (1,20)),
                   variable1 = rnorm(100),
                   variable2 = rpois(100, 2))

# create partition indices
kfolds <- kfold_partition(data, dependent = "species", k = 2)

# data for partition 1
data[kfolds$Fold_1, ]
```

---

model_selection            *Selection of best candidate models considering various criteria*

---

### Description

Applies a series of criteria to select best candidate models.

### Usage

```
model_selection(evaluation_stats, criterion = "TSS", exclude_bimodal = FALSE,
                tolerance = 0.01)
```

### Arguments

evaluation_stats

    data.frame with the statistics of model evaluation results. These results are the output of the function [evaluation_stats](#).

criterion            (character) metric used as the predictive criterion for model selection.

exclude_bimodal

    (logical) whether to exclude models in which binomial variable response curves were detected.

tolerance            (numeric)

### Value

A data.frame with one or more selected models.

### Examples

```
# data
data("cal_res", package = "enmpa")
eval_stats <- cal_res$summary[, -1]

# selecting best model
selected_mod <- model_selection(eval_stats, exclude_bimodal = TRUE)
```

---

| | |
|---|---|
| model_validation | *Model validation options* |

---

### Description

Model evaluation using entire set of data and a k-fold cross validation approach. Models are assessed based on discrimination power (ROC-AUC), classification ability (accuracy, sensitivity, specificity, TSS, etc.), and the balance between fitting and complexity (AIC).

### Usage

```
model_validation(formula, data, family = binomial(link = "logit"),
                 weights = NULL, cv = FALSE, partition_index = NULL,
                 k = NULL, dependent = NULL, n_threshold = 100,
                 keep_coefficients = FALSE, seed = 1)
```

### Arguments

| | |
|---|---|
| formula | (character) expression to be used as a glm formula. |
| data | data.frame with dependent and independent variables. |
| family | a family object for models used by functions such as glm. Default = binomial(link = "logit"). |
| weights | (numeric) vector with weights for observations. Default = NULL. |
| cv | (logical) whether to use a k-fold cross validation for evaluation. Default = FALSE. |
| partition_index | list of indices for cross validation in k-fold. Obtained with the function [kfold_partition](). Default = NULL. |
| k | (numeric) number of folds for a new k-fold index preparation. Ignored if partition_index is defined or if cv = FALSE. Default = NULL. |
| dependent | (character) name of dependent variable. Ignore if cv = FALSE. Default = NULL. |
| n_threshold | (numeric) number of threshold values to be used for ROC. Default = 100. |
| keep_coefficients | (logical) whether to keep model coefficients. Default = FALSE. |
| seed | (numeric) a seed number. Default = 1. |

### Value

A data.frame with results from evaluation.

## Examples

```
# Load species occurrences and environmental data.
data("enm_data", package = "enmpa")
head(enm_data)

# Custom formula
form <- c("Sp ~ bio_1 + I(bio_1^2) + I(bio_12^2)")

# Model evaluation using the entire set of records
model_validation(form, data = enm_data)

# Model evaluation using a k-fold cross-validation (k = 3)
model_validation(form, data = enm_data, cv = TRUE, k = 3, dependent = "Sp")
```

---

niche_signal                    *Niche Signal detection using one or multiple variables*

---

## Description

Identifies whether a signal of niche can be detected using one or multiple variables. This is an implementation of the methods developed by Cobos & Peterson (2022) doi:10.17161/bi.v17i.15985 that focuses on identifying niche signals in presence-absence data.

## Usage

```
niche_signal(data, condition, variables, method = "univariate",
             permanova_method = "mahalanobis", iterations = 1000,
             set_seed = 1, verbose = TRUE, ...)

niche_signal_univariate(data, condition, variable, iterations = 1000,
                        set_seed = 1, verbose = TRUE)

niche_signal_permanova(data, condition, variables, permutations = 999,
                       permanova_method = "mahalanobis", verbose = TRUE, ...)
```

## Arguments

| | |
|---|---|
| data | matrix or data.frame containing at least the following information: a column representing condition (positive = 1 or negative = 0), and one or more columns representing environmental variables. |
| condition | (character) name of the column with numeric information about detection (positive = 1 or negative = 0). |
| variables | (character) vector of one or more names of columns to be used as environmental variables. If method = "univariate", only one variable is used; for method = "permanova", multiple variables can be used. |
| method | (character) name of the method to be used for niche comparison. Default = "univariate". |

permanova_method

(character) name of the dissimilarity index to be used as method in adonis2. See all options in vegdist. Default = "mahalanobis".

iterations        (numeric) number of iterations to be used in analysis. Default = 1000. If method = "permanova", permutations = iterations - 1.

set_seed          (numeric) integer value to specify a initial seed. Default = 1.

verbose           (logical) whether or not to print messages about the process. Default = TRUE.

...               other arguments to be passed to adonis2.

variable          (character) name of the column containing data to be used as environmental variable.

permutations      number of permutations to be performed.

### Value

A list with results from analysis depending on method.

### Examples

```
# Load species occurrences and environmental data.
data("enm_data", package = "enmpa")
head(enm_data)

# Detection of niche signal using an univariate non-parametric test
sn_bio1 <- niche_signal(data = enm_data, variables = "bio_1",
                        condition = "Sp", method = "univariate")
sn_bio1

sn_bio12 <- niche_signal(data = enm_data, variables = "bio_12",
                         condition = "Sp", method = "univariate")
sn_bio12
```

---

optimize_metrics        *Find threshold values to produce three optimal metrics*

---

### Description

The metrics true skill statistic (TSS), sensitivity, specificity are explored by comparing actual vs predicted values to find threshold values that produce sensitivity = specificity, maximum TSS, and a sensitivity value of 0.9.

### Usage

```
optimize_metrics(actual, predicted, n_threshold = 100)
```

## Arguments

| | |
|---|---|
| `actual` | (numeric) vector of actual values (0, 1) to be compared to `predicted` values after applying a threshold. |
| `predicted` | (numeric) vector of predicted probability values to be thresholded and compared to `actual`. |
| `n_threshold` | (numeric) number of threshold values to be used. Default = 100. |

## Value

A list containing a data.frame with the resulting metrics for all threshold values tested, and a second data.frame with the results for the threshold values that produce sensitivity = specificity (ESS), maximum TSS (maxTSS), and a sensitivity value of 0.9 (SEN90).

## Examples

```
# example data
act <- c(rep(1, 20), rep(0, 80))
pred <- c(runif(20, min = 0.4, max = 0.7), runif(80, min = 0, max = 0.5))

# run example
om <- optimize_metrics(actual = act, predicted = pred)
om$optimized
```

---

| `plot_importance` | *Plot variable importance* |
|---|---|

---

## Description

Visualization of the results obtained with the function [var_importance](var_importance).

## Usage

```
plot_importance(x, xlab = NULL, ylab = "Relative contribution",
                main = "Variable importance", extra_info = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `x` | data.frame output from [var_importance](var_importance). |
| `xlab` | (character) a label for the x axis. |
| `ylab` | (character) a label for the y axis. |
| `main` | (character) main title for the plot. |
| `extra_info` | (logical) when results are from more than one model, it adds information about the number of models using each predictor and the mean contribution found. |
| `...` | additional arguments passed to [barplot](barplot) or [boxplot](boxplot). |

## Value

A plot

## Examples

```
# Load species occurrences and environmental data.
data("enm_data", package = "enmpa")

# Custom formulas
forms <- c("Sp ~ bio_1 + I(bio_1^2) + I(bio_12^2)",
           "Sp ~ bio_12 + I(bio_1^2) + I(bio_12^2)")

# Fitting models
fits <- fit_glms(forms, data = enm_data)

# Variable importance for single models
vi_1 <- var_importance(fits$ModelID_1)
plot_importance(x = vi_1)

vi_2 <- var_importance(fits$ModelID_2)
plot_importance(x = vi_2)

# Variable importance for multiple models
vi_c <- var_importance(fits)
plot_importance(x = vi_c)
```

---

plot_jk                          *Jackkniffe plot for variable contribution*

---

## Description

The jackknife figure shows the impact of each variable on the full model, providing detailed information about the function and significance of each variable. Light blue indicates the impact on the model if the variable is not included, while dark blue indicates the independent contribution of the variable to the model.

## Usage

```
plot_jk(x, metric = "ROC_AUC", legend = TRUE,
        colors = c("cyan", "blue", "red"),
        xlab = NULL, main = NULL)
```

## Arguments

| | |
|---|---|
| x | list output from [jackknife](#). |
| metric | (character) model metric to plot. Default = "ROC_AUC". |
| legend | (logical) whether to add legend. Default = TRUE. |
| colors | (character) vector of colors. Default = c("cyan", "blue", "red"). |

| xlab | (character) a label for the x axis. |
|---|---|
| main | (character) main title for the plot. |

---

plot_niche_signal  *Plot Niche Signal results*

---

## Description

Plots to interpret results from niche_signal tests (Cobos & Peterson (2022) doi:10.17161/bi.v17i.15985).

## Usage

```
plot_niche_signal(niche_signal_list, statistic = "mean",
                  variables = NULL, ellipses = FALSE, level = 0.99,
                  breaks = "Sturges", main = "", xlab = NULL, ylab = NULL,
                  h_col = "lightgray", h_cex = 0.8, lty = 2, lwd = 1,
                  l_col = c("blue", "black"), e_col = c("black", "red"),
                  pch = 19, pt_cex = c(1.3, 0.8), pt_col = c("black", "red"),
                  ...)

plot_niche_signal_univariate(niche_signal_univariate_list, statistic = "mean",
                             breaks = "Sturges", main = "", xlab = NULL,
                             ylab = "Frequency", h_col = "lightgray",
                             h_cex = 0.8, lty = 2, lwd = 1,
                             l_col = c("blue", "black"), ...)

plot_niche_signal_permanova(niche_signal_permanova_list, variables = NULL,
                            ellipses = FALSE, level = 0.99, main = "",
                            xlab = NULL, ylab = NULL,
                            e_col = c("black", "red"), lty = 2, lwd = 1,
                            pch = 19, pt_cex = c(1.3, 0.8),
                            pt_col = c("black", "red"), ...)
```

## Arguments

| niche_signal_list | |
|---|---|
| | list of results from niche_signal. |
| statistic | (character) name of the statistic for which results will be explored when results come for univariate analysis. Default = "mean". Options are: "mean", "median", "SD", and "range". |
| variables | (character) name of variables to used in plots when results come from analysis using the permanova method. The default, NULL, uses the first two variables. |
| ellipses | (logical) whether to use ellipses to represent all and positive data when results come from PERMANOVA. The default, FALSE, plots points instead. |

| | |
|---|---|
| level | (numeric) value from 0 to 1 representing the limit of the ellipse to be plotted. Default = 0.99. |
| breaks | breaks in the histogram as in [hist](#). Default = "Sturges". |
| main | (character) title for plot. Default = "". |
| xlab | (character) x axis label. Default = NULL. For results from PERMANOVA, appropriate variable names are used. |
| ylab | (character) y axis label. Default = NULL. For univariate results, the default turn into "Frequency". For results from PERMANOVA, appropriate variable names are used. |
| h_col | a color to be used to fill the bars of histograms. Default = "lightgray". |
| h_cex | (numeric) value by which plotting text and symbols should be magnified relative to the default in histograms. Default = 0.8. |
| lty | (numeric) line type. See options in [par](#). Default = 2. A vector of length = 2 can be used, in order, all and positive. |
| lwd | (numeric) line width. See options in [par](#). Default = 1. A vector of length = 2 can be used, in order, all and positive. |
| l_col | line color for observed value of positives and confidence intervals. Default = c("blue", "black"). |
| e_col | color of ellipse lines for all and positive data. Default = c("black", "red"). |
| pch | point type. See options in [points](#). Default = 19. A vector of length = 2 can be used, in order, all and positive. |
| pt_cex | (numeric) value by which points will be magnified. Values for all and positive points are recommended. Default = c(1.3, 0.8). |
| pt_col | color for points. Values for all and positive points are recommended. Default = c("black", "red"). |
| ... | other plotting arguments to be used. |
| niche_signal_univariate_list | |
| | list of results from niche_signal_univariate. |
| niche_signal_permanova_list | |
| | list of results from niche_signal_permanova. |

### Value

A plot.

### Examples

```
# Load species occurrences and environmental data.
data("enm_data", package = "enmpa")
head(enm_data)

# Detection of niche signal using an univariate non-parametric test
sn_bio1 <- niche_signal(data = enm_data,
                        variables = "bio_1",
                        condition = "Sp",
```

```
                                  method = "univariate")

  plot_niche_signal(sn_bio1, variables = "bio_1")

  sn_bio12 <- niche_signal(data = enm_data,
                           variables = "bio_12",
                           condition = "Sp",
                           method = "univariate")

  plot_niche_signal(sn_bio12, variables = "bio_12")
```

---

predict_glm                 *Extension of glm predict to generate predictions of different types*

---

### Description

Obtains predictions from a fitted generalized linear model objects. It also allows the clamping option to restrict extrapolation in areas outside the calibration area.

### Usage

```
predict_glm(
  model,
  newdata,
  data = NULL,
  extrapolation_type = "E",
  restricted_vars = NULL,
  type = "response"
)
```

### Arguments

model                 a glm object.

newdata               a data.frame or matrix with the new data to project the predictions.

data                  data.frame or matrix of data used in the model calibration step. Default = NULL.

extrapolation_type
                      (character) to indicate extrapolation type of model. Models can be transferred with three options: free extrapolation ('E'), extrapolation with clamping ('EC'), and no extrapolation ('NE'). Default = 'E'.

restricted_vars
                      (character) a vector containing the names of the variables that will undergo clamping or no extrapolation. For clamping, these variables are set to minimum and maximum values established for the max and min values within calibration values. For no extrapolation, the variables outside calibration limits became NA. If no specific names are provided, the value is set to NULL by default, indicating that clamping (EC) or no extrapolation (NE) will be applied to all variables. Ignore if extrapolation_type = 'E'.

| type | (character) the type of prediction required. For a default binomial model the default predictions are of log-odds (probabilities on logit scale). The default, "response", returns predicted probabilities. |
| --- | --- |

### Value

A `SpatRaster` object or a vector with predictions.

### Examples

```
# Load fitted model
data("sel_fit", package = "enmpa")

# Load raster layers to be projected
env_vars <- terra::rast(system.file("extdata", "vars.tif", package = "enmpa"))

# Prediction
pred <- predict_glm(sel_fit$glms_fitted$ModelID_7, newdata = env_vars,
                    data = sel_fit$data)
terra::plot(pred)
```

---

predict_selected            *Predictions for the models selected after calibration*

---

### Description

Wrapper function that facilitates the prediction of those models selected as the most robust. In addition, it allows the calculation of consensus models, when more than one model are selected.

### Usage

```
predict_selected(fitted, newdata, extrapolation_type = "E",
                 restricted_vars = NULL, type = "response", consensus = TRUE)
```

### Arguments

| fitted | an enmpa-classfitted models object obtained using the functions [fit_selected](#). |
| --- | --- |
| newdata | a `SpatRaster`, data.frame, or matrix with the new data on which to predict. |
| extrapolation_type | |
| | (character) to indicate extrapolation type of model. Models can be transferred with three options: free extrapolation ('E'), extrapolation with clamping ('EC'), and no extrapolation ('NE'). Default = 'E'. |
| restricted_vars | |
| | (character) a vector containing the names of the variables that will undergo clamping or no extrapolation. For clamping, these variables are set to minimum and maximum values established for the max and min values within calibration values. For no extrapolation, the variables outside calibration limits became NA. |

If no specific names are provided, the value is set to NULL by default, indicating that clamping (EC) or no extrapolation (NE) will be applied to all variables. Ignore if extrapolation_type = 'E'.

type             (character) the type of prediction required. For a default binomial model the default predictions are of log-odds (probabilities on logit scale). The default, "response", returns predicted probabilities.

consensus        (logical) valid if newdata is a SpatRaster, whether to produce consensus results obtained by combining the predictions from the collection of selected models. By default consensuses are calculated using the mean, median, variance, and weighted average using the AIC weights. Default = TRUE.

### Value

A list with predictions of selected models on the newdata and fitted selected model(s). Consensus predictions are added if multiple selected models exits and if newdata is a SpatRaster object.

### Examples

```
# Load a fitted selected model
data(sel_fit, package = "enmpa")

# Load raster layers to be projected
env_vars <- terra::rast(system.file("extdata", "vars.tif", package = "enmpa"))

# Predictions (only one selected mode, no consensus required)
preds <- predict_selected(sel_fit, newdata = env_vars, consensus = FALSE)

# Plot prediction
terra::plot(preds$predictions)
```

---

print                         *Print a short version of elements in 'calibration' and 'fitted models'*
                              *objects*

---

### Description

Print a short version of elements in 'calibration' and 'fitted models' objects

### Usage

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

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

## Arguments

x            object of enmpa_fitted_models or enmpa_calibration

...          additional arguments affecting the summary produced. Ignored in these functions.

---

proc_enm                *Partial ROC calculation*

---

## Description

proc applies partial ROC tests to model predictions.

## Usage

```
proc_enm(test_prediction, prediction, threshold = 5, sample_percentage = 50,
         iterations = 500)
```

## Arguments

test_prediction

         (numeric) vector of model predictions for testing data.

prediction       SpatRaster or numeric vector of model predictions to be evaluated.

threshold        (numeric) value from 0 to 100 to represent the percentage of potential error (E) that the data could have due to any source of uncertainty. Default = 5.

sample_percentage

         (numeric) percentage of testing data to be used in each bootstrapped process for calculating the partial ROC. Default = 50.

iterations        (numeric) number of bootstrap iterations to be performed; default = 500.

## Details

Partial ROC is calculated following Peterson et al. (2008) [doi:10.1016/j.ecolmodel.2007.11.008](doi:10.1016/j.ecolmodel.2007.11.008).

## Value

A list with the summary of the results and a data.frame containing the AUC values and AUC ratios calculated for all iterations.

## Examples

```
# Loading a model prediction
pred <- terra::rast(system.file("extdata", "proj_out_wmean.tif",
                                package = "enmpa"))

# Simulated data
test <- runif(100, min = 0.3, max = 0.8)

# partial ROC calculation
pr <- proc_enm(test, pred, threshold = 5, sample_percentage = 50,
               iterations = 500)
```

---

resp2var                    *Two-Way interaction response plot*

---

## Description

A view of the species probability into a two-dimensional environmental space.

## Usage

```
resp2var(model, variable1 , variable2, modelID = NULL, data = NULL, n = 1000,
         new_data = NULL, extrapolate = FALSE, add_bar = TRUE,
         add_limits = FALSE, color.palette = NULL, xlab = NULL, ylab = NULL,
         ...)
```

## Arguments

| | |
|---|---|
| model | an object of class `glm` or `enmpa_fitted_models`. |
| variable1 | (character) name of the variable to be plotted in x axis. |
| variable2 | (character) name of the variable to be plotted in y axis. |
| modelID | (character) name of the ModelID if inputed `model` is in the `enmpa_fitted_models` object. Default = NULL. |
| data | data.frame or matrix of data to be used in model calibration. Default = NULL. |
| n | (numeric) an integer guiding the number of breaks. Default = 100 |
| new_data | a `SpatRaster`, data.frame, or matrix of variables representing the range of variable values in an area of interest. Default = NULL. |
| extrapolate | (logical) whether to allow extrapolation to study the behavior of the response outside the calibration limits. Ignored if `new_data` is defined. Default = TRUE. |
| add_bar | (logical) whether to add bar legend. Default = TRUE. |
| add_limits | (logical) whether to add calibration limits if `extrapolate` = TRUE. Default = FALSE. |
| color.palette | (function) a color palette function to be used to assign colors in the plot. Default = function(n) rev(hcl.colors(n, "terrain")). |

| xlab | (character) a label for the x axis. The default, NULL, uses the name defined in variable1. |
|---|---|
| ylab | (character) a label for the y axis. The default, NULL, uses the name defined in variable2. |
| ... | additional arguments passed to [image](). |

#### Details

The function calculates probabilities by focusing on each combination of the two supplied environmental variable while keeping all other variables constant at their mean values.

#### Value

A plot with the response interaction of two environmental dimensions for variable1 and variable2, and don't return anything.

#### Examples

```
# Load a fitted selected model
data(sel_fit, package = "enmpa")

# Two-Way interaction response plot in the calibration limits

resp2var(sel_fit, variable1 = "bio_1", variable2 = "bio_12", xlab = "BIO-1",
ylab = "BIO-12", modelID = "ModelID_7")

# Two-Way interaction response plot allowing extrapolation
resp2var(sel_fit, variable1 = "bio_1", variable2 = "bio_12", xlab = "BIO-1",
ylab = "BIO-12", modelID = "ModelID_7", extrapolate = TRUE)
```

---

response_curve                *Variable response curves for GLMs*

---

#### Description

A view of variable responses in models. Responses based on single or multiple models can be provided.

#### Usage

```
response_curve(fitted, variable, data = NULL, modelID = NULL,  n = 100,
                new_data = NULL,  extrapolate = TRUE, xlab = NULL,
                ylab = "Probability", col = "red", ...)
```

## Arguments

| | |
|---|---|
| `fitted` | an object of class glm, a list of GLMs obtained using the function [fit_glms](#), or an object enmpa_fitted_models from [fit_selected](#) . |
| `variable` | (character) name of the variables to be plotted. |
| `data` | data.frame or matrix of data used in the model calibration step. Default = NULL. |
| `modelID` | (character) vector of ModelID(s) to be considered when the fitted models is an `enmpa_fitted_object`. By default all models are included. Default = NULL. |
| `n` | (numeric) an integer guiding the number of breaks. Default = 100 |
| `new_data` | a `SpatRaster`, data.frame, or matrix of variables representing the range of variable values in an area of interest. Default = NULL. It must be defined in case the model entered does not explicitly include a data component. |
| `extrapolate` | (logical) whether to allow extrapolation to study the behavior of the response outside the calibration limits. Ignored if `new_data` is defined. Default = TRUE. |
| `xlab` | (character) a label for the x axis. The default, NULL, uses the name defined in `variable`. |
| `ylab` | (character) a label for the y axis. Default = "Probability". |
| `col` | (character) color for lines. Default = "red". |
| `...` | additional arguments passed to [plot](#). |

## Details

The function calculates these probabilities by focusing on a single environmental variable while keeping all other variables constant at their mean values.

When responses for multiple models are to be plotted, the mean and confidence intervals for the set of responses are calculated using a GAM.

## Value

A plot with the response curve for a `variable`.

## Examples

```
# Load a fitted selected model
data(sel_fit, package = "enmpa")

# Response curve for single models
response_curve(sel_fit$ModelID_7, variable = "bio_1")

# Response curve when model(s) are in a list (only one model in this one)
response_curve(sel_fit, variable = "bio_12")
```

---

sel_fit                                    *Example of selected models fitted*

---

### Description

An object of the class enmpa_fitted_models containing fitted selected model(s) and the information from model evaluation for such model(s).

### Usage

```
sel_fit
```

### Format

A list with four elements.

**glms_fitted** a fitted glm (ModelID_7).

**selected** a data.frame with results from evaluation of ModelID_7

**data** a data.frame containing information on presence and absence, and independent variables used to fit GLM models.

**weights** a vector with the weights for observations.

### Examples

```
data("sel_fit", package = "enmpa")
```

---

summary                        *Summary of 'calibration' and 'fitted models'*

---

### Description

Summary of 'calibration' and 'fitted models'

### Usage

```
## S3 method for class 'enmpa_calibration'
summary(object, ...)

## S3 method for class 'enmpa_fitted_models'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | of class enmpa_calibration or enmpa_fitted_models |
| ... | additional arguments affecting the summary produced. Ignored in these functions. |

**Value**

A printed summary.

---

test                              *Example data used to test models*

---

**Description**

A dataset containing information on presence and absence, and independent variables used to fit GLM models.

**Usage**

```
test
```

**Format**

A data frame with 100 rows and 3 columns.

**Sp**  numeric, values of 0 = absence and 1 = presence.

**lon**  numeric, longitude values.

**lat**  numeric, latitude values.

**Examples**

```
data("test", package = "enmpa")
head(test)
```

---

var_importance                   *Variable importance for GLMs*

---

**Description**

Calculates the relative importance of predictor variables based on the concept of explained deviance. This is achieved by fitting a GLMs multiple times, each time leaving out a different predictor variable to observe its impact on the model's performance.

**Usage**

```
var_importance(fitted, modelID = NULL, data = NULL)
```

## Arguments

| | |
|---|---|
| `fitted` | an object of class glm, a list of GLMs obtained using the function [fit_glms](#), or an object enmpa_fitted_models from [fit_selected](#) . |
| `modelID` | (character) vector of ModelID(s) to be considered when the `fitted` input is of the class enmpa_fitted_models. By default all models are included. Default = NULL. |
| `data` | data.frame or matrix of data used in the model calibration step. It must be defined in case the model entered does not explicitly include a data component. Default = NULL. |

## Details

The process begins by fitting the full GLM model, which includes all predictor variables. Subsequently, separate GLM models are fitted, excluding one variable at a time to assess the influence of its absence on the model's performance. By systematically evaluating the effect of removing each predictor variable, the function provides valuable insights into their individual contributions to the model's overall performance and explanatory power.

## Value

A data.frame containing the relative contribution of each variable. An identification for distinct models is added if `fitted` contains multiple models.

## Examples

```
# Load a fitted selected model
data(sel_fit, package = "enmpa")

# Variable importance for single models
var_importance(sel_fit, modelID = "ModelID_7")

# Variable importance for multiple models (only one model in this list)
var_importance(sel_fit)
```

# Index